МИКРОСХЕМА ИНТЕГРАЛЬНАЯ

1867ВН016

Руководство пользователя

2019

Содержание

1 Введение

В настоящее время цифровая обработка сигналов (ЦОС) охватывает широкий спектр практических приложений. В качестве примеров можно назвать цифровую фильтрацию, кодирование речи, обработку изображений, быстрое преобразование Фурье (БПФ), телекоммуникацию и цифровую звукотехнику. Как в этих, так и в других областях, цифровая обработка сигналов имеет общие особенности:

- большой объём вычислений;
- работа в реальном времени;
- обработка оцифрованных данных;
- гибкость систем ЦОС.

Развитие техники интегральных схем позволило создавать более быстрые микропроцессоры и микро-ЭВМ. В результате во многих областях, связанных с цифровой обработкой сигналов, матричные процессоры стали вытесняться разрядно-модульными микропроцессорными секциями, а затем однокристальными микро-ЭВМ. Цена цифровых процессоров обработки сигналов (ПЦОС) резко снизилась, открыв дорогу для их самого широкого внедрения. Рост производительности ИС расширил область применения ПЦОС, по традиции ограничивающуюся телефонной связью, и ПЦОС стали применяться в графических системах и системах обработки изображений, а затем и в звукотехнике. Последним достижением техники ЦОС стали однокристальные цифровые процессоры сигналов, примером которых служат процессоры серии TMS320.

На текущий момент отечественной промышленностью освоен серийный выпуск широкой номенклатуры 16-разрядных ИС ПЦОС с фиксированной запятой: 1867ВМ1 (функциональный аналог TMS320C10), 1867ВМ2 (функциональный аналог TMS320C25), 1867ВЦ2Т (функциональный аналог TMS320C50), 1867ВЦ4Т (функциональный аналог TMS320C542), 1867ВЦ5Т (функциональный аналог TMS320F240).

Однако, необходимость решения сложных задач системного уровня, для которых характерно значительное увеличение динамического диапазона и высокая производительность, обусловила проектирование и освоение серийного производства 32-разрядных ИС ПЦОС, способных оперировать с данными в формате как с фиксированной, так и с плавающей запятой, например, ИС 1867ВЦ6Ф (функциональный аналог DSP TMS320C30), ИС 1867ВЦ3Ф (функциональный аналог TMS320C40).

Дальнейшим развитием семейства 32-разрядных ПЦОС стала разработанная в рамках ОКР «Обработка-И6» ИС 1867ВН016 – радиационно-стойкая «система в корпусе» с двухъядерным высокопроизводительным процессором цифровой обработки сигналов. ИС 1867ВН016 представляет собой однородную вычислительную систему: два одинаковых процессорных ядра, каждое из которых полностью реализует архитектуру, систему команд и способы адресации ПЦОС с плавающей и фиксированной запятой 1867ВЦ6Ф (функциональный аналог DSP TMS320C30 фирмы Texas Instruments) и его радиационно-стойкого аналога ИС 1867ВЦ11Ф.

Настоящий документ содержит общее описание «системы в корпусе», функциональное назначение выводов ИС, а также конспективное описание процессорного ядра с модифицированными (по сравнению с ИС 1867ВЦ6Ф из-за необходимости организации межъядерного взаимодействия и добавления совместно используемых периферийных устройств) системой прерываний, регистровым файлом и картой памяти. Основное внимание в данном документе уделяется максимально подробному описанию функционирования каждого из совместно используемых (разделяемых) периферийных устройств, совместно используемой внутрикристальной памяти и организации внешнего интерфейса ИС 1867ВН016. Этот документ предназначен, как для разработчиков, уже имеющих опыт проектирования аппаратуры на базе ИС 1867ВЦ6Ф (TMS320C3x), ИС 1867ВЦ11Ф, так и для программистов, разрабатывавших программный код на уровне ассемблера или с использованием высокоуровневых компиляторов для этих микросхем.

Описания разделяемых периферийных устройств и интерфейсов «системы в корпусе» ИС 1867ВН016: MIL-STD-1553B, UART, ARINC-429, GPIO, USB 2.0, TechBus и ПЧК – представлены в разделах 13, 14, 15, 16, 17, 18 и 19 соответственно.

Неотъемлемой частью документа КФДЛ.431299.034ТО является приложение А «Система в корпусе», процессорное ядро», в котором приведено подробное описание архитектуры процессорного ядра и взаимодействия его основных блоков и регистров, системы внутриядерных шин и организации обмена по ним, а также детальный разбор форматов данных, способов их адресации, набора инструкций на языке ассемблера и всех возможных вариантов управления последовательностью исполнения программного кода. Кроме этого, в приложении А рассматривается функционирование неразделяемых (индивидуально ассоциированных с каждым из ядер) периферийных устройств.

2 Функциональные характеристики и электрические параметры ИС 1867ВН016

ИС 1867ВН016 – это состоящая из трёх кристаллов высокопроизводительная двухъядерная «система в корпусе». На главном кристалле ИС реализованы два 32-разрядных процессорных ядра цифровой обработки сигналов с плавающей запятой, входной тактовой частотой до 80 МГц и общим объёмом адресуемой внутренней и внешней памяти до 8 Мбайт на каждое ядро. Процессорные ядра выполнены на базе высокопроизводительного микропроцессора 1867ВЦ6Ф (архитектура, система команд и способы адресации DSP TMS320C30).

В каждом ядре реализован программный загрузчик (bootloader). Высокоскоростной межъядерный обмен данными осуществляется через реализованное на главном кристалле «системы в корпусе» совместное двухпортовое 32-разрядное ОЗУ (далее – IDPRAM) объёмом 16К слов.

С каждым из процессорных ядер ИС 1867ВН016 ассоциированы реализованные в виде отдельных кристаллов индивидуальные однопортовые 32-разрядные ОЗУ (далее – INTSRAM) объёмом 512К слов каждое.

Набор периферийных устройств, расположенных на главном кристалле «системы в корпусе» ИС 1867ВН016, включает:

- четыре контроллера интерфейса ГОСТ Р 52070-2003 (MIL-STD-1553B) со скоростью передачи/приёма 1 Мбит/с, резервированием и буферным ОЗУ 8К × 32 бит (по 2К × 32 бит на каждый контроллер);
- восемь контроллеров интерфейса ГОСТ 18977-79 (ARINC-429) со скоростью передачи-приёма 50/250 Кбит/с каждый (восемь универсальных приёмопередающих модулей с переключением скорости и наличием буферного ОЗУ 1К × 32 бит (по 128 × 32 бит на каждый контроллер);
- два контроллера UART (NS16550A) со скоростью передачи/приёма – 50 – 3 000 000 бит/с, с 8-/32-разрядным режимом работы;
- один контроллер USB 2.0 с режимами передачи/приёма High Speed /Full Speed со скоростью 12/480 Мбит/с и буферным ОЗУ 2К × 32 бит;
- одно специализированное пользовательское устройство, обеспечивающее работу по заданному алгоритму преобразования частота-код;
- одна технологическая шина с изменяемой скоростью работы;
- 32 программируемые линии ввода-вывода (порты ввода-вывода общего назначения);
- Два отладочных порта (по одному на каждое ядро).

Все вышеперечисленные периферийные устройства (за исключением индивидуальных для каждого ядра отладочных портов) являются разделяемыми и могут подключаться к любому из процессорных ядер в любое время через коммутатор и, соответственно, управляться любым ядром.

Такая конфигурация «системы в корпусе» ИС 1867ВН016 обеспечивает возможность реализации эффективной мультипроцессорной обработки данных.

2.1 Технические характеристики функциональных элементов ИС 1867ВН016

В таблице 2.1 приведены функциональные характеристики процессорного ядра ИС 1867ВН016, в таблице 2.2, 2.3 – характеристики ОЗУ, а в таблицах 2.4 – 2.8 – характеристики внутрикристальных периферийных устройств.

Таблица 2.1 – Функциональные характеристики процессорного ядра микросхемы

Наименование параметра, единица измерения	Значение параметра
Архитектура	1867ВЦ6Ф (TMS320C30)
Система команд процессорных ядер	1867ВЦ6Ф (TMS320C30)
Разрядность АЛУ, бит: - плавающая запятая (ПЗ) - фиксированная запятая (ФЗ)	 40 32
Разрядность умножителя, бит: - ПЗ - ФЗ	 32 × 32 = 40 24 × 24 = 32
Объём ПЗУ, бит	4К × 32
Объём ОЗУ, бит	2К × 32
Объём кэш-ОЗУ, бит	64 × 32
Объём адресуемой памяти, бит	16М × 32 + 16К × 32
Количество 32-разрядных таймеров	2
Количество последовательных портов	2
Сопроцессор ПДП	1
Тактовая частота процессора, не более, МГц	80/40*
Производительность: - MFLOPS, не менее - MIPS, не менее	 80 40
Начальный загрузчик	есть
Порт отладки	есть
* В зависимости от состояния входа CLKMD: CLKMD = 1 – не более 80 МГц/ CLKMD = 0 – не более 40 МГц (тестовый режим).	

Таблица 2.2 – Функциональные характеристики двухпортового ОЗУ

Наименование параметра, единица измерения	Значение параметра
Объём ОЗУ, бит	16К × 32
Количество портов записи	2
Количество портов чтения	2
Доступ к ОЗУ	из каждого процессорного ядра

Таблица 2.3 – Функциональные характеристики ОЗУ для каждого ядра

Наименование параметра, единица измерения	Значение параметра
Объём ОЗУ, бит	512К × 32

Таблица 2.4 – Функциональные характеристики интерфейса MIL-STD-1553B

Наименование параметра, единица измерения	Значение параметра
Архитектура и функции	в соответствии с ГОСТ Р 52070-2003
Режимы работы	контроллер шины; оконечное устройство; монитор
Приёмопередатчики	внешние
Скорость передачи/приёма, Мбит/с	1
Объём буферного ОЗУ, бит	4 × (2К × 32)*
* Для каждого из четырёх каналов индивидуальное ОЗУ (2К × 32) бит, суммарно – (8К × 32) бит.	

Таблица 2.5 – Функциональные характеристики контроллера интерфейса UART

Наименование параметра, единица измерения	Значение параметра
Архитектура (регистровый уровень)	NS16550A
Скорость передачи/приёма, бит/с	50 – 3 000 000
Режим работы	8-/32-разрядный
Режим передачи/приёма	асинхронный
Контроль чётности/нечётности	есть
Управление контролем по чётности	есть
Длина передаваемых/принимаемых данных, бит	5/6/7/8
Количество стоп-бит	1/1,5/2

Таблица 2.6 – Функциональные характеристики контроллера интерфейса USB 2.0

Наименование параметра, единица измерения	Значение параметра
Количество конечных точек (End Points), не менее	4
Размер буферного ОЗУ, бит	4K × 32
Скорость передачи/приёма, Мбит/с	12/480
Режимы работы	Full Speed/High Speed
Формирование сигнала прерывания	есть
Интерфейс физического уровня	UTMI+

Таблица 2.7 – Функциональные характеристики контроллера интерфейса ARINC-429

Наименование параметра, единица измерения	Значение параметра
Архитектура и функции	в соответствии с ГОСТ 18977-79
Скорость передачи/приёма, Кбит/с	50/250
Количество приёмопередатчиков	8
Буферное ОЗУ, бит	1K × 32

Таблица 2.8 – Функциональные характеристики устройств ввода-вывода общего назначения

Наименование параметра, единица измерения	Значение параметра
Количество входов/выходов общего назначения, не менее	32

Таблица 2.9 – Функциональные характеристики преобразователя частота-код

Наименование параметра, единица измерения	Значение параметра
Общее количество входов преобразования	12
Максимальное количество одновременно работающих входов преобразования	6
Максимальная частота преобразуемого сигнала, МГц	16
Разрядность результата преобразования (положительные и отрицательные значения в дополнительном коде), бит	13

2.2 Электрические параметры и режимы эксплуатации ИС 1867ВН016

В таблице 2.10 приведены электрические параметры ИС при приёмке и поставке, а в таблице 2.11 – нормы предельно допустимых и предельных режимов эксплуатации ИС.

Таблица 2.10 – Электрические параметры микросхем при приёмке и поставке

Наименование параметра, единица измерения, режим измерения	Буквенное обозначение параметра	Норма параметра		Температура среды, ºС
		не менее	не более	
1 Выходное напряжение низкого уровня, В,[1] U_{CC1} = 1,62 В, U_{CC2} = 3,0 В, U_{CC3} = 3,0 В, U_{CC4} = 3,0 В, U_{CC5} = 3,0 В, U_{CC6} = 3,0 В, U_{CC7} = 3,0 В, U_{CC8} = 3,0 В, I_{OL} = 2,0 мА	U_{OL}	–	0,4	–60 ± 3 25 ± 10 85 ± 3
2 Выходное напряжение высокого уровня, В,[1] U_{CC1} = 1,62 В, U_{CC2} = 3,0 В, U_{CC3} = 3,0 В, U_{CC4} = 3,0 В, U_{CC5} = 3,0 В, U_{CC6} = 3,0 В, U_{CC7} = 3,0 В, U_{CC8} = 3,0 В, I_{OH} = – 0,3 мА	U_{OH}	2,4	–	
3 Ток утечки низкого уровня на входах, мкА,[2] U_{CC1} = 1,98 В, U_{CC2} = 3,6 В, U_{CC3} = 3,6 В, U_{CC4} = 3,6 В, U_{CC5} = 3,6 В, U_{CC6} = 3,6 В, U_{CC7} = 3,6 В, U_{CC8} = 3,6 В, U_{IL} = 0,45 В	I_{ILL}	–30	–	
4 Ток утечки высокого уровня на входах, мкА,[2] U_{CC1} = 1,98 В, U_{CC2} = 3,6 В, U_{CC3} = 3,6 В, U_{CC4} = 3,6 В, U_{CC5} = 3,6 В, U_{CC6} = 3,6 В, U_{CC7} = 3,6 В, U_{CC8} = 3,6 В, U_{IH} = 3,6 В	I_{ILH}	–	30	
5 Динамический ток потребления ядра, мА,[3] U_{CC1} = 1,98 В, U_{CC2} = 3,6 В, U_{CC3} = 3,6 В, U_{CC4} = 3,6 В, U_{CC5} = 3,6 В, U_{CC6} = 3,6 В, U_{CC7} = 3,6 В, U_{CC8} = 3,6 В, U_{IH} = 3,6 В, f_{CI} = 80 МГц	I_{OCC1}	–	1 200	
6 Функциональный контроль U_{CC1} = (1,62; 1,98) В, U_{CC2} = (3,0; 3,6) В, U_{CC3} = (3,0; 3,6) В, U_{CC4} = (3,0; 3,6) В, U_{CC5} = (3,0; 3,6) В, U_{CC6} = (3,0; 3,6) В, U_{CC7} = (3,0; 3,6) В, U_{CC8} = (3,0; 3,6) В, f_{CI} = 80 МГц	ФК	–	–	

[1] Параметры U_{OL}, U_{OH} измеряются по выводам A3, A6, A11 – A13, A20, A21, A23, AA3 – AA7, AA9, AA11, AA12, AA15, AA16, AA20 – AA22, AA24, AB1, AB4 – AB8, AB10, AB14 – AB17, AB27, AC1, AC3 – AC7, AC9 – AC11, AC17, AC18, AC24, AC25, AC27, AD2 – AD12, AD15 – AD18, AD23, AD24, AD26, AD27, AE1 – AE3, AE5, AE8, AE12, AE16, AE18 – AE20, AE23 – AE26, AF1, AF2, AF4, AF6, AF8, AF17 – AF20, AF22 –AF27, AG2, AG3, AG5, AG8, AG13, AG16, AG17, AG20, AG22, AG25, AG26, B2 – B4, B7, B12, B16, B23, B27, C1, C3 – C5, C12, C13, C21, C25 – C27, D2 – D11, D13, D16, D17, D21, D23, D25, D26, E1, E4 – E7, E9, E10, E13 – E15, E21 – E23, F2, F4 – F16, F18, F20, G4 – G9, G11 – G13, G15 – G17, G19, H1 – H4, H6, H7, H9, H10, H13, J4 –J6, J8, J21, J23, J24, K3 – K6, K8, K20, K22 – K24, L1, L4, L5, L7, L8, L21 – L23, L26, M3, M4, M6, M24, M27, N1, N3 – N6, N25, N27, P3, P4, P6, P25, R3 – R6, R22 – R25, T3, T4, T6, T24, T25, U2, U4 – U8, V3 – V6, V8, V22, V26, W2 – W5, W20, W21, W23, Y1, Y3, Y4, Y6 – Y13, Y15, Y17, Y21, Y22, Y24 – Y26.

[2] Параметры I_{ILL}, I_{ILH} измеряются по выводам A3, A6, A7, A11 – A13, A26, AA3 – AA7, AA15, AA16, AA23, AA24, AB1, AB4, AB13, AB15, AB16, AB20 – AB23, AB27, AC1, AC3, AC4, AC13, AC18, AC19, AC21 – AC25, AC27, AD2 – AD4, AD13, AD15 – AD24, AD26, AD27, AE1 – AE3, AE5, AE10, AE13, AE16, AE18 – AE20, AE23 – AE26, AF1, AF2, AF4, AF17 – AF20, AF22 – AF27, AG2, AG3, AG11, AG16, AG17, AG20, AG22, AG25, AG26, B2 – B4, B12, B24, B26, C1, C3 – C5, C12, C13, C16, C24, D2 – D4, D12, D13, D19, D20, D24, E1, E4, E11, E13, E17 – E19, E24 – E26, F2, F4, F13 – F17, F19, F24, F27, G4 – G6, G13, G15, G16, G21, G23, G24, G27, H1 – H4, H6, H7, H13, H19 – H22, H24 – H27, J4 – J6, J8, J20, J25, K3 – K6, K8, K25, K26, L1, L4, L5, L7, L8, L24, M3, M4, M6, M7, M21, N1, N3 – N5, N7, N20, N22 – N24, P3, P4, P23, R3 –R5, R7, R20, R27, T3, T4, T25, T26, U2, U4 – U8, U20, U21, V3 – V6, V8, V20, V26, W2 – W5, Y1, Y3, Y4, Y6 – Y8, Y14, Y15, Y24 – Y26.

Параметры I_{ILL}, I_{ILH} при температуре (−60 ± 3) ºС не измеряются, а гарантируются нормами при температуре (25 ± 10) ºС.

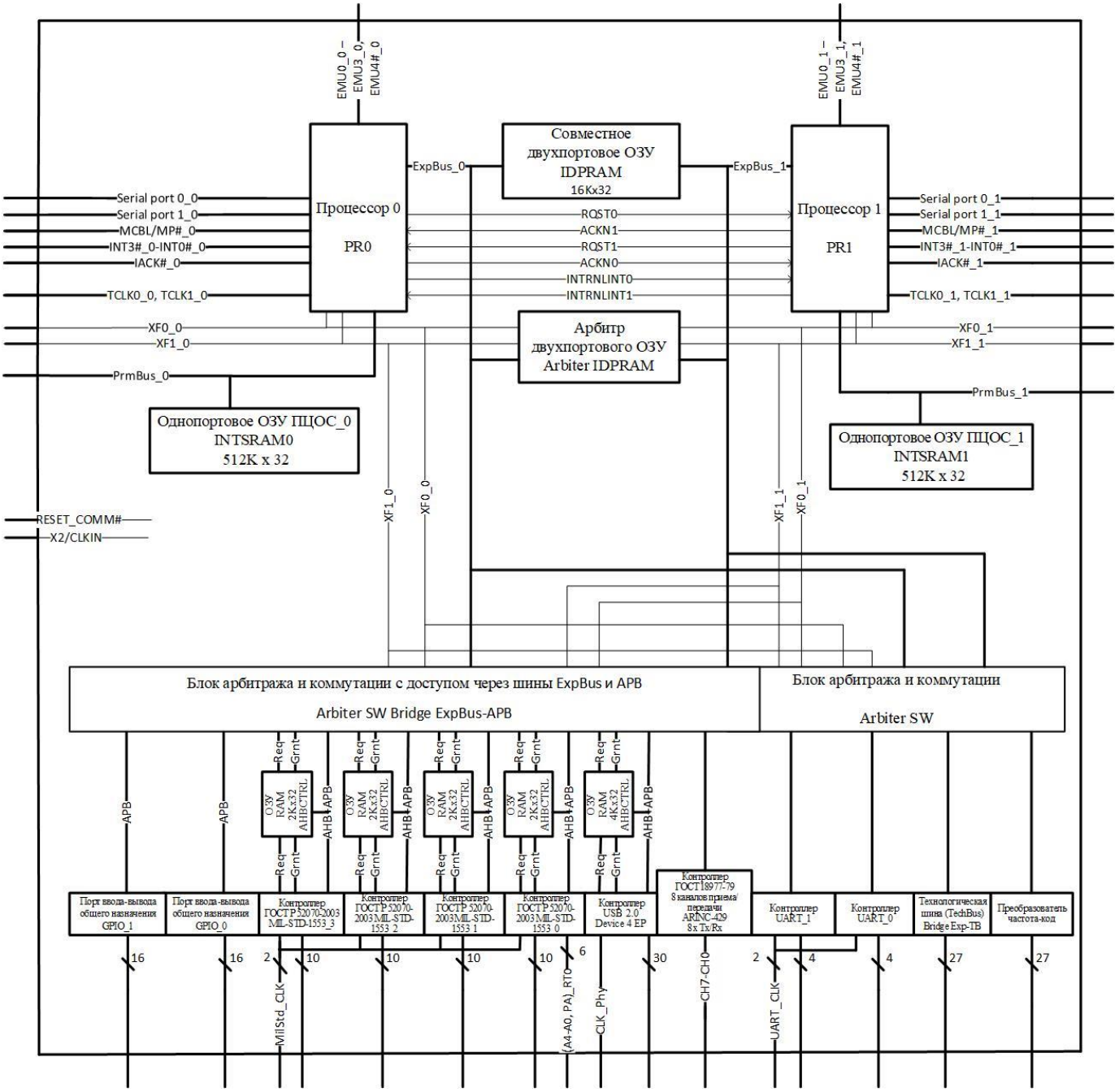
[3] Параметр I_{OCC1} измеряется при отключенных от нагрузок выходах и подключенных к уровням U_{IL} и U_{IH} входах.

Таблица 2.11 – Нормы предельно допустимых и предельных режимов эксплуатации ИС

Наименование параметра режима, единица измерения	Буквенное обозначение параметра	Предельно допустимый режим		Предельный режим	
		не менее	не более	не менее	не более
1 Напряжение питания по выводам VCC_CL, В	U_{CC1}	1,62	1,98	–0,2	2,2
2 Напряжение питания по выводам VCC_DR, В	U_{CC2}	3,0	3,6	–0,3	4,0
3 Напряжение питания по выводам VDDA33, В	U_{CC3}	3,0	3,6	–0,3	4,0
4 Напряжение питания по выводам VDDE33, В	U_{CC4}	3,0	3,6	–0,3	4,0
5 Напряжение питания по выводу VDD33_osc_CPU, В	U_{CC5}	3,0	3,6	–0,3	4,0
6 Напряжение питания по выводу VDD33_osc_Mstd, В	U_{CC6}	3,0	3,6	–0,3	4,0
7 Напряжение питания по выводу VDD33_osc_UART, В	U_{CC7}	3,0	3,6	–0,3	4,0
8 Напряжение питания по выводу VDD33_osc_FI, В	U_{CC8}	3,0	3,6	–0,3	4,0
9 Входное напряжение низкого уровня, В	U_{IL}	0	0,6	–0,3	–
10 Входное напряжение высокого уровня, В	U_{IH}	0,7 U_{CC2}	U_{CC2}	–	U_{CC2} + 0,3
11 Выходной ток низкого уровня, мА	I_{OL}	–	2,0	–	–
12 Выходной ток высокого уровня, мА	I_{OH}	–0,3	–	–	–
13 Ёмкость нагрузки, пФ	C_L	–	40	–	–
14 Частота следования импульсов тактового сигнала, МГц	f_{CI}	–	80	–	–
Примечание – Время работы в одном из предельных режимов должно быть не более 5 с.					

3 Структурная схема ИС

Структурная схема «системы в корпусе» ИС 1867ВН016 приведена на рисунке 3.1.



Принятые условные обозначения основных шин и сигналов на структурной схеме:

PrmBus_0 – первичный интерфейс процессора 0 предназначен для подключения внутреннего однопортового ОЗУ INTSRAM0 и внешних запоминающих устройств;

PrmBus_1 – первичный интерфейс процессора 1 предназначен для подключения внутреннего однопортового ОЗУ INTSRAM1 и внешних запоминающих устройств;

ExpBus_0 – расширенный интерфейс процессора 0 предназначен для подключения внутреннего двухпортового ОЗУ IDPRAM и разделяемых периферийных устройств;

ExpBus_1 – расширенный интерфейс процессора 1 предназначен для подключения внутреннего двухпортового ОЗУ IDPRAM и разделяемых периферийных устройств;

APB, AHB-APB – шины подключения разделяемых периферийных устройств;

Serial port 0_0 – интерфейс последовательного порта 0 процессора 0;

Serial port 1_0 – интерфейс последовательного порта 1 процессора 0;

Serial port 0_1 – интерфейс последовательного порта 0 процессора 1;

Serial port 1_1 – интерфейс последовательного порта 1 процессора 1;

Рисунок 3.1, лист 1 – Структурная схема микросхемы 1867ВН016

RESET_COMM#	– сигнал общего сброса;
X2/CLKIN	– общий тактовый сигнал;
MCBL/MP#_0	– сигнал выбора режима микрокомпьютер/микропроцессор процессора 0;
MCBL/MP#_1	– сигнал выбора режима микрокомпьютер/микропроцессор процессора 1;
INT3#_0-INT0#_0	– сигналы внешних прерываний процессора 0;
INT3#_1-INT0#_1	– сигналы внешних прерываний процессора 1;
IACK#_0	– сигнал подтверждения прерываний процессора 0;
IACK#_1	– сигнал подтверждения прерываний процессора 1;
TCLK0_0, TCLK1_0	– сигналы таймеров 0 и 1 процессора 0;
TCLK0_1, TCLK1_1	– сигналы таймеров 0 и 1 процессора 1;
XF0_0, XF1_0,	– сигналы внешних флагов 0 и 1 процессора 0;
XF0_1, XF1_1	– сигналы внешних флагов 0 и 1 процессора 1;
Req	– сигналы запроса обработки событий;
Grnt	– сигналы готовности обработки событий;
MilStd_CLK	– тактовый сигнал модулей MIL-STD-1553_(0, 1, 2, 3);
CLK_Phy	– тактовый сигнал модуля USB 2.0;
UART_CLK	– тактовый сигнал модулей UART_(0, 1);
(A4-A0, PA)_RT0	– сигналы адреса удаленного устройства MIL-STD-1553_0;
CH7-CH0	– сигналы приемопередатчиков контроллера ARINC-429;
RQST0	– сигнал запроса на обслуживание процессора 0;
RQST1	– сигнал запроса на обслуживание процессора 1;
ACKN0	– сигнал подтверждения обслуживания процессора 0;
ACKN1	– сигнал подтверждения обслуживания процессора 1;
INTRNLINT0	– сигнал внутреннего прерывания процессора 0;
INTRNLINT1	– сигнал внутреннего прерывания процессора 1;
EMU0_0-EMU3_0	– сигналы внутрипроцессорного эмулятора процессора 0;
EMU0_1-EMU3_1	– сигналы внутрипроцессорного эмулятора процессора 1;
EMU4#_0	– сигнал перевода выводов в высокоимпедансное состояние процессора 0;
EMU4#_1	– сигнал перевода выводов в высокоимпедансное состояние процессора 1.

Рисунок 3.1, лист 2

«Система в корпусе» ИС 1867ВН016 является однородной двухъядерной вычислительной системой, состоящей из блоков, связываемых различными шинами и сигналами, см. рисунок 3.1.

Далее, в разделе 4 «Организация памяти, регистровый файл ядра ИС и система прерываний» приведены модифицированные (по сравнению с ИС 1867ВЦ6Ф) карты памяти, расширенная система прерываний процессорного ядра, включая межъядерные прерывания, а также описаны загрузчик пользовательской программы (bootloader) и вновь добавленный регистр номера процессора (PNR). Все изменения и дополнения, для наглядности, выделены в тексте этого раздела бледно-зелёным фоном (см. пример далее).

Пример – Информация о дополнениях в процессорном ядре

Все последующие разделы, начиная с пятого, виртуально полностью выделены бледно-зелёным фоном, поскольку представляют собой детальное описание только вновь добавленных разделяемых (совместно используемых обоими процессорными ядрами «системы в корпусе» ИС 1867ВН016) периферийных устройств, интерфейсов и двухпортовой памяти.

4 Организация памяти, регистровый файл ядра ИС и система прерываний

4.1 Модифицированные карты памяти процессорного ядра ИС 1867ВН016

Процессор 1867ВЦ6Ф (TMS320C30), прототип ядра ИС 1867ВН016, поддерживает общий объём адресуемой памяти 16M × 32 бит. За исключением внутрикристальных ресурсов: двух внутренних блоков ОЗУ (RAM Block 0 и RAM Block 1 объёмом 1K × 32 бит каждый), внутреннего масочного ПЗУ объёмом 4K × 32 бит (доступно только в микрокомпьютерном режиме!), а также области размером 6K слов, отведённые под картированные в памяти регистры периферийных устройств (последовательные порты, таймеры, ПДП, управление внешним интерфейсом), остальная часть адресного пространства предназначена для подключения внешних запоминающих устройств, которые могут обслуживаться двумя интерфейсами – основным (Primary Bus) и расширенным (Expansion Bus).

Оба внешних интерфейса ИС-прототипа имеют собственные двунаправленные 32-разрядные шины данных; собственные адресные шины (24-разрядная для Primary Bus и 13-разрядная для Expansion Bus); собственные стробы (STRB# для Primary Bus, MSTRB# и IOSTRB# для Expansion Bus); собственные сигналы, индицирующие направление обмена данными R/W#, XR/W# (чтение или запись) и собственные сигналы готовности внешнего устройства к передаче/приёму информации.

Primary Bus обслуживает два диапазона адресов: от 00_0000h (00_1000h) в зависимости от режима микропроцессор (микрокомпьютер) до 7F_FFFFh и от 80_A000h до FF_FFFFh. Expansion Bus обслуживает диапазон 80_0000h – 80_1FFFh, активируя строб MSTRB# и диапазон 80_4000h – 80_5FFFh, активируя строб IOSTRB#. Области адресов от 80_2000h до 80_3FFFh и от 80_6000h до 80_7FFFh зарезервированы. Все картированные в памяти периферийные регистры размещены в диапазоне адресов от 80_8000h до 80_97FFh. Независимо от режима микропроцессор/микрокомпьютер внутренние блоки ОЗУ 0 и ОЗУ 1 размещены в областях с адресами от 80_9800h до 80_9BFFh и от 80_9C00h до 80_9FFFh, соответственно. Вся вышеприведённая информация структурирована в таблицах 4.1, 4.2 (см. невыделенные строки).

Отдельно взятое ядро ИС 1867ВН016, в целом, повторяет карту адресного пространства прототипа – в тех же диапазонах расположены все внутренние запоминающие устройства (RAM, ROM) и картированные в памяти регистры «стандартных» (реализованных в ИС 1867ВЦ6Ф) периферийных устройств. Но есть несколько отличий в картах памяти, как для отдельно взятых ядер ИС 1867ВН016 (по сравнению с ИС 1867ВЦ6Ф), так и для «системы в корпусе» в целом. Все эти отличия выделены в таблицах 4.1, 4.2 бледно-зелёным фоном.

Одно из отличий заключается в том, что в каждом из двух ядер ИС 1867ВН016 реализован начальный загрузчик (bootloader), из-за чего векторы прерываний при работе в микрокомпьютерном режиме ремэпированы из нулевых адресов в другую область (см. подраздел 4.3 «Загрузчик пользовательской программы и регистр номера процессора»).

Другое отличие – выделение в индивидуальной карте памяти каждого ядра области, «предназначенной» для подключения внешнего постоянного запоминающего устройства (ПЗУ). При обращении в диапазон адресов 08_0000h – 0F_FFFFh (неразделяемая часть памяти) активируются сигналы CEROM#_*n* и OEROM#_*n*, где *n* – номер ядра (0 или 1). Эта опция позволяет минимизировать аппаратные затраты на организацию внешнего интерфейса при подключении внешнего(их) ПЗУ к ИС 1867ВН016. В этот диапазон адресов можно подключать и обычное ОЗУ, т. к. при обращении в эту область сигналы STRB#_*n* и R/W#_*n*, наряду с CEROM#_*n* и OEROM#_*n*, также остаются активны.

Если выполняется обращение к внутренним, индивидуальным для каждого ядра ОЗУ RAM Block 0 (диапазон адресов: 80_9800h – 80_9BFFh) и RAM Block 1 (80_9C00h – 80_9FFFh), внешний интерфейс неактивен (сигнал(ы) STRB#_*n* не формируются). При обращении к индивидуальным ОЗУ INTSRAM*n*, которые с точки зрения пользователя «системы на кристалле» также являются внутренними, сигналы внешнего интерфейса STRB#_*n* и R/W#_*n* остаются активны. Поэтому, если к ИС 1867ВН016 подключено какое-либо внешнее запоминающее устройство, при проектировании аппаратуры должна быть предусмотрена дополнительная внешняя (по отношению к «системе в корпусе») дешифрация адресной шины, чтобы исключить возможность активации внешней памяти при обращении в диапазон адресов, отведённый под INTSRAM*n*.

Практически всё адресное пространство «системы в корпусе» ИС 1867ВН016, выделенное под расширенный интерфейс (за исключением регистров PNR, IIR и BLCR), является разделяемым и используется для организации межъядерного обмена и взаимодействия. Каждое из ядер через эту область адресует совместно используемое ОЗУ – IDPRAM и картированные в памяти регистры разделяемых периферийных устройств. Таким образом, в ИС 1867ВН016 Expansion Bus является внутренним интерфейсом «системы в корпусе». При этом независимо от значения, записанного в регистр управления Expansion Bus (EBCR), доступ к IDPRAM осуществляется за один машинный цикл. Для обеспечения корректного доступа к расположенным в адресном диапазоне Expansion Bus регистрам разделяемых периферийных устройств, каждое из которых вырабатывает свой сигнал готовности к обмену, поле SWW регистра EBCR (адрес 80_8060h) необходимо установить в 00b (ожидание сигнала готовности). Описание регистра EBCR приведено в таблице А.7.2 приложения А.

Поскольку при выборе микрокомпьютерного режима в двухъядерной ИС 1867ВН016 активируется начальный загрузчик (bootloader), входы управления выбором режима микрокомпьютер/микропроцессор именуются «MCBL/MP#_*n*», где *n* – номер ядра (в отличие от «MC/MP#» для одноядерного процессора-прототипа 1867ВЦ6Ф). Карта памяти каждого из ядер «системы в корпусе» зависит от того, в каком режиме оно работает: микропроцессорном (при низком уровне «MCBL/MP#_*n*») или микрокомпьютерном (при высоком уровне «MCBL/MP#_*n*»). Карты памяти для этих режимов приведены в таблицах 4.1 и 4.2.

Таблица 4.1 – Карта памяти процессорного ядра (n = 0, 1) в режиме MCBL/MP#_*n* = 1

Адрес	Обозначение области памяти
Неразделяемая область памяти	
00_0000h – 00_0FFFh	Внутреннее ПЗУ (4096 × 32) бит: вектор сброса; ремэпинг векторов прерываний; начальный загрузчик
00_1000h – 07_FFFFh	Внешняя память (508K × 32) бит /STRB#/
08_0000h – 0F_FFFFh	Внешняя память (512K × 32) бит /STRB#/; возможность подключения внешнего ПЗУ /CEROM#, OEROM#/
10_0000h – 7F_FFFFh	Внешняя память (7168K × 32) бит /STRB#/
Разделяемая область памяти	
80_0000h – 80_3FFFh	IDPRAM (16K × 32) бит
80_4000h – 80_401Fh	ARINC-429 (25 × 32) бит, RSV (7 × 32) бит
80_4020h – 80_403Fh	UART_0 (10 × 32) бит, RSV (22 × 32) бит
80_4040h – 80_405Fh	UART_1 (10 × 32) бит, RSV (22 × 32) бит
80_4060h – 80_40EFh	USB 2.0 (34 × 32) бит, RSV(132 × 32) бит
80_40F0h – 80_41EFh	MIL-STD-1553_0 (34 × 32) бит, RSV (222 × 32) бит
80_41F0h – 80_42EFh	MIL-STD-1553_1 (34 × 32) бит, RSV (222 × 32) бит
80_42F0h – 80_43EFh	MIL-STD-1553_2 (34 × 32) бит, RSV (222 × 32) бит
80_43F0h – 80_44EFh	MIL-STD-1553_3 (34 × 32) бит, RSV (222 × 32) бит
80_44F0h – 80_453Fh	GPIO_0 (38 × 32) бит, RSV (42 × 32) бит
80_4540h – 80_457Fh	GPIO_1 (38 × 32) бит, RSV (26 × 32) бит
80_4580h – 80_4599h	CMMTR (26 × 32) бит
80_459Ah – 80_4612h	RSV (121 × 32) бит
80_4613h – 80_4616h	TechBus (4 × 32) бит
80_4617h – 80_467Fh	RSV (105 × 32) бит
80_4680h – 80_468Dh	PCHK (14 × 32) бит
80_468Eh – 80_46FFh	RSV (114 × 32) бит
Неразделяемая область памяти	
80_4700h	Processor Number Register (PNR)
80_4701h	Internal Interrupts Register (IIR)
80_4702h	Регистр кода первоначальной загрузки (BLCR)
80_4703h – 80_4FFFh	RSV (2301 × 32) бит
Разделяемая область памяти	
80_5000h – 80_57FFh	MIL-STD-1553_0 RAM (2K × 32) бит
80_5800h – 80_5FFFh	MIL-STD-1553_1 RAM (2K × 32) бит
80_6000h – 80_6FFFh	USB 2.0 RAM (4K × 32) бит
80_7000h – 80_77FFh	MIL-STD-1553_2 RAM (2K × 32) бит
80_7800h – 80_7FFFh	MIL-STD-1553_3 RAM (2K × 32) бит
Неразделяемая область памяти	
80_8000h – 80_97FFh	Peripheral Bus Memory-Mapped Registers (6K × 32) бит
80_9800h – 80_9BFFh	RAM Block 0 (1K × 32) бит
80_9C00h – 80_9FFFh	RAM Block 1 (1K × 32) бит
80_A000h – F7_FFFFh	Внешняя память (7640K × 32) бит /STRB#/, в том числе векторы прерываний по адресам 80_A001h – 80_A03Fh
F8_0000h – FF_FFFFh	INTSRAM (512K × 32) бит, /STRB#/

Примечание – Принятые условные обозначения и термины:
- ядру 0 соответствует вывод MCBL/MP#_0; ядру 1 соответствует вывод MCBL/MP#_1;
- неразделяемая область – область памяти, доступная только одному из ядер;
- разделяемая область – область памяти, которая доступна обоим ядрам;
- IDPRAM – внутреннее, совместно используемое, двухпортовое ОЗУ;
- INTSRAM – внутреннее, индивидуальное для каждого ядра ОЗУ (512K × 32) бит;
- RSV – зарезервированные области (ячейки).

Таблица 4.2 – Карта памяти процессорного ядра (n = 0, 1) в режиме MCBL/MP#_*n* = 0

Адрес	Обозначение области памяти
Неразделяемая область памяти	
00_0000h – 07_FFFFh	Внешняя память (512K × 32) бит /STRB#/, в том числе вектор сброса и векторы прерываний по адресам 00_0000h – 00_003Fh
08_0000h – 0F_FFFFh	Внешняя память (512K × 32) бит /STRB#/; возможность подключения внешнего ПЗУ /CEROM#, OEROM#/
10_0000h – 7F_FFFFh	Внешняя память (7168K × 32) бит /STRB#/
Разделяемая область памяти	
80_0000h – 80_3FFFh	IDPRAM (16K × 32) бит
80_4000h – 80_401Fh	ARINC-429 (25 × 32) бит, RSV (7 × 32) бит
80_4020h – 80_403Fh	UART_0 (10 × 32) бит, RSV (22 × 32) бит
80_4040h – 80_405Fh	UART_1 (10 × 32) бит, RSV (22 × 32) бит
80_4060h – 80_40EFh	USB 2.0 (34 × 32) бит, RSV (132 × 32) бит
80_40F0h – 80_41EFh	MIL-STD-1553_0 (34 × 32) бит, RSV (222 × 32) бит
80_41F0h – 80_42EFh	MIL-STD-1553_1 (34 × 32) бит, RSV (222 × 32) бит
80_42F0h – 80_43EFh	MIL-STD-1553_2 (34 × 32) бит, RSV (222 × 32) бит
80_43F0h – 80_44EFh	MIL-STD-1553_3 (34 × 32) бит, RSV (222 × 32) бит
80_44F0h – 80_453Fh	GPIO_0 (38 × 32) бит, RSV (42 × 32) бит
80_4540h – 80_457Fh	GPIO_1 (38 × 32) бит, RSV (26 × 32) бит
80_4580h – 80_4599h	CMMTR (26 × 32) бит
80_459Ah – 80_4612h	RSV (121 × 32) бит
80_4613h – 80_4616h	TechBus (4 × 32) бит
80_4617h – 80_467Fh	RSV (105 × 32) бит
80_4680h – 80_468Dh	PCHK (14 × 32) бит
80_468Eh – 80_46FFh	RSV (114 × 32) бит
Неразделяемая область памяти	
80_4700h	Processor Number Register (PNR)
80_4701h	Internal Interrupts Register (IIR)
80_4702h	Регистр кода первоначальной загрузки (BLCR)
80_4703h – 80_4FFFh	RSV (2301 × 32) бит
Разделяемая область памяти	
80_5000h – 80_57FFh	MIL-STD-1553_0 RAM (2K × 32) бит
80_5800h – 80_5FFFh	MIL-STD-1553_1 RAM (2K × 32) бит
80_6000h – 80_6FFFh	USB 2.0 RAM (4K × 32) бит
80_7000h – 80_77FFh	MIL-STD-1553_2 RAM (2K × 32) бит
80_7800h – 80_7FFFh	MIL-STD-1553_3 RAM (2K × 32) бит
Неразделяемая область памяти	
80_8000h – 80_97FFh	Peripheral Bus Memory-Mapped Registers (6K × 32) бит
80_9800h – 80_9BFFh	RAM Block 0 (1K × 32) бит
80_9C00h – 80_9FFFh	RAM Block 1 (1K × 32) бит
80_A000h – F7_FFFFh	Внешняя память (7640K × 32) бит /STRB#/
F8_0000h – FF_FFFFh	INTSRAM (512K × 32) бит, /STRB#/
Примечание – Принятые условные обозначения и термины: - ядру 0 соответствует вывод MCBL/MP#_0; ядру 1 соответствует вывод MCBL/MP#_1; - неразделяемая область – область памяти, доступная только одному из ядер; - разделяемая область – область памяти, которая доступна обоим ядрам; - IDPRAM – внутреннее, совместно используемое, двухпортовое ОЗУ; - INTSRAM – внутреннее, индивидуальное для каждого ядра ОЗУ (512K × 32) бит; - RSV – зарезервированные области (ячейки).	

Детальная карта памяти процессорных ядер 0 и 1 приведена в таблице 4.3.

Таблица 4.3 – Регистры неразделяемых и разделяемых периферийных устройств

Адрес	Мнемоника	Периферийное устройство
1	2	3
Неразделяемые периферийные устройства		
ПДП (DMA)		
80_8000h	DMACTRL	Регистр глобального управления ПДП
80_8004h	DMASRC	Регистр адреса источника ПДП
80_8006h	DMADST	Регистр адреса приёмника ПДП
80_8008h	DMATXCNTR	Регистр счётчика пересылок ПДП
80_8009h – 80_801Fh	RSV	Зарезервировано
Таймеры		
80_8020h	TCTRL0	Регистр глобального управления таймером 0
80_8024h	TCNTR0	Регистр счётчика таймера 0
80_8028h	TPRD0	Регистр периода таймера 0
80_8030h	TCTRL1	Регистр глобального управления таймером 1
80_8034h	TCNTR1	Регистр счётчика таймера 1
80_8038h	TPRD1	Регистр периода таймера 1
Последовательный порт 0		
80_8040h	SPGBCTRL0	Регистр глобального управления последовательным портом 0
80_8042h	SPDXCTRL0	Регистр управления FSX/DX/CLKX последовательного порта 0
80_8043h	SPDRCTRL0	Регистр управления FSR/DR/CLKR последовательного порта 0
80_8044h	SPTCTRL0	Регистр управления R/X-таймера последовательного порта 0
80_8045h	SPTCNTR0	Регистр счётчика R/X-таймера последовательного порта 0
80_8046h	SPTPRD0	Регистр периода R/X-таймера последовательного порта 0
80_8048h	SPTR0	Регистр передаваемых данных последовательного порта 0
80_804Ch	SPRCV0	Регистр принимаемых данных последовательного порта 0
80_804Dh – 80_804Fh	RSV	Зарезервировано
Последовательный порт 1		
80_8050h	SPGBCTRL1	Регистр глобального управления последовательным портом 1
80_8052h	SPDXCTRL1	Регистр управления FSX/DX/CLKX последовательного порта 1
80_8053h	SPDRCTRL1	Регистр управления FSR/DR/CLKR последовательного порта 1
80_8054h	SPTCTRL1	Регистр управления R/X-таймера последовательного порта 1
80_8055h	SPTCNTR1	Регистр счётчика R/X-таймера последовательного порта 1
80_8056h	SPTPRD1	Регистр периода R/X-таймера последовательного порта 1
80_8058h	SPTR1	Регистр передаваемых данных последовательного порта 1
80_805Ch	SPRCV1	Регистр принимаемых данных последовательного порта 1
80_805Dh – 80_805Fh	RSV	Зарезервировано
Регистры управления интерфейсами		
80_8060h	EBCR	Регистр управления интерфейсом Expansion Bus
80_8064h	PBCR	Регистр управления интерфейсом Primary Bus
Регистры номера процессора и управления прерываниями		
80_4700h	PNR	Регистр номера процессора
80_4701h	IIR	Регистр управления линиями прерывания смежного процессора RQST, ACKN и INTRNLINT
80_4702h	BLCR	Регистр кода начальной загрузки
80_4703h – 80_3FFFh	RSV	Зарезервировано

Продолжение таблицы 4.3

1	2	3
Разделяемые периферийные устройства		
ARINC-429		
80_4000h	AR429RXTXCON0	Регистр управления приёмником/передатчиком 0
80_4001h	AR429RXTXCON1	Регистр управления приёмником/передатчиком 1
80_4002h	AR429RXTXCON2	Регистр управления приёмником/передатчиком 2
80_4003h	AR429RXTXCON3	Регистр управления приёмником/передатчиком 3
80_4004h	AR429RXTXCON4	Регистр управления приёмником/передатчиком 4
80_4005h	AR429RXTXCON5	Регистр управления приёмником/передатчиком 5
80_4006h	AR429RXTXCON6	Регистр управления приёмником/передатчиком 6
80_4007h	AR429RXTXCON7	Регистр управления приёмником/передатчиком 7
80_4008h	AR429RXTXSTAT0	Регистр состояния приёмника/передатчика 0
80_4009h	AR429RXTXSTAT1	Регистр состояния приёмника/передатчика 1
80_400Ah	AR429RXTXSTAT2	Регистр состояния приёмника/передатчика 2
80_400Bh	AR429RXTXSTAT3	Регистр состояния приёмника/передатчика 3
80_400Ch	AR429RXTXSTAT4	Регистр состояния приёмника/передатчика 4
80_400Dh	AR429RXTXSTAT5	Регистр состояния приёмника/передатчика 5
80_400Eh	AR429RXTXSTAT6	Регистр состояния приёмника/передатчика 6
80_400Fh	AR429RXTXSTAT7	Регистр состояния приёмника/передатчика 7
80_4010h	AR429RXLABELFIFO	Регистр меток и номера FIFO
80_4011h	AR429RXTXDATA0	32 разряда данных приёмника/передатчика 0
80_4012h	AR429RXTXDATA1	32 разряда данных приёмника/передатчика 1
80_4013h	AR429RXTXDATA2	32 разряда данных приёмника/передатчика 2
80_4014h	AR429RXTXDATA3	32 разряда данных приёмника/передатчика 3
80_4015h	AR429RXTXDATA4	32 разряда данных приёмника/передатчика 4
80_4016h	AR429RXTXDATA5	32 разряда данных приёмника/передатчика 5
80_4017h	AR429RXTXDATA6	32 разряда данных приёмника/передатчика 6
80_4018h	AR429RXTXDATA7	32 разряда данных приёмника/передатчика 7
80_4019h – 80_401Fh	RSV	Зарезервировано

Продолжение таблицы 4.3

1	2	3
UART_0		
80_4020h	RB0	Выход FIFO приёмника, только чтение
80_4020h	THR0	Вход FIFO передатчика, только запись
80_4020h	DLB10	DLB10 (LSB), Младший байт регистра делителя
80_4021h	IER0	Регистр разрешения (маски) прерываний
80_4021h	DLB20	DLB20 (MSB), Старший байт регистра делителя
80_4022h	II0	Регистр идентификации прерываний, только чтение
80_4022h	FC0	Регистр управления FIFO, только запись
80_4023h	LCR0	Регистр управления линией
80_4024h	MCR0	Регистр управления модемом, только запись
80_4025h	LSR0	Регистр состояния линии, только чтение
80_4026h	MSR0	Регистр статуса модема, только чтение
80_4027h	RSV	Зарезервировано
80_4028h	DBG10	Debug10, Первый отладочный регистр, только чтение
80_4029h – 80_402Bh	RSV	Зарезервировано
80_402Ch	DBG20	Debug20, Второй отладочный регистр, только чтение
80_402Dh – 80_4032h	RSV	Зарезервировано
80_4033h	SELCLKUART0	Регистр выбора источника тактирования UART_0
80_4034h – 80_403Fh	RSV	Зарезервировано
UART_1		
80_4040h	RB1	Выход FIFO приёмника, только чтение
80_4040h	THR1	Вход FIFO передатчика, только запись
80_4040h	DLB11	DLB11 (LSB), Младший байт регистра делителя
80_4041h	IER1	Регистр разрешения (маски) прерываний
80_4041h	DLB21	DLB21 (MSB), Старший байт регистра делителя
80_4042h	II1	Регистр идентификации прерываний, только чтение
80_4042h	FC1	Регистр управления FIFO, только запись
80_4043h	LCR1	Регистр управления линией
80_4044h	MCR1	Регистр управления модемом, только запись
80_4045h	LSR1	Регистр состояния линии, только чтение
80_4046h	MSR1	Регистр статуса модема, только чтение
80_4047h	RSV	Зарезервировано
80_4048h	DBG11	Debug11, Первый отладочный регистр, только чтение
80_4049h – 80_404Bh	RSV	Зарезервировано
80_404Ch	DBG21	Debug21, Второй отладочный регистр, только чтение
80_404Dh – 80_4052h	RSV	Зарезервировано
80_4053h	SELCLKUART1	Регистр выбора источника тактирования UART_1
80_4054h – 80_405Fh	RSV	Зарезервировано

Продолжение таблицы 4.3

1	2	3
USB 2.0		
Конечная точка OUT EP0		
80_4060h	CTRLOUTEP0	Регистр управления конечной точкой 0 OUT EP0
80_4064h	DMACTRLOUTEP0	Регистр управления DMA конечной точки 0 OUT EP0
80_4068h	DMADAOUTEP0	Дескриптор адреса DMA конечной точки 0 OUT EP0
80_406Ch	STOUTEP0	Статусный регистр конечной точки 0 OUT EP0
Конечная точка OUT EP1		
80_4070h	CTRLOUTEP1	Регистр управления конечной точкой 1 OUT EP1
80_4074h	DMACTRLOUTEP1	Регистр управления DMA конечной точки 1 OUT EP1
80_4078h	DMADAOUTEP1	Дескриптор адреса DMA конечной точки 1 OUT EP1
80_407Ch	STOUTEP1	Статусный регистр конечной точки 1 OUT EP1
Конечная точка OUT EP2		
80_4080h	CTRLOUTEP2	Регистр управления конечной точкой 2 OUT EP2
80_4084h	DMACTRLOUTEP2	Регистр управления DMA конечной точки 2 OUT EP2
80_4088h	DMADAOUTEP2	Дескриптор адреса DMA конечной точки 2 OUT EP2
80_408Ch	STOUTEP2	Статусный регистр конечной точки 2 OUT EP2
Конечная точка OUT EP3		
80_4090h	CTRLOUTEP3	Регистр управления конечной точкой 3 OUT EP3
80_4094h	DMACTRLOUTEP3	Регистр управления DMA конечной точки 3 OUT EP3
80_4098h	DMADAOUTEP3	Дескриптор адреса DMA конечной точки 3 OUT EP3
80_409Ch	STOUTEP3	Статусный регистр конечной точки 3 OUT EP3
Конечная точка IN EP0		
80_40A0h	CTRLINEP0	Регистр управления конечной точкой 0 IN EP0
80_40A4h	CTRLDMAINEP0	Регистр управления DMA конечной точки 0 IN EP0
80_40A8h	DMADAINEP0	Дескриптор адреса DMA конечной точки 0 IN EP0
80_40ACh	STINEP0	Статусный регистр конечной точки 0 IN EP0
Конечная точка IN EP1		
80_40B0h	CTRLINEP1	Регистр управления конечной точкой 1 IN EP1
80_40B4h	CTRLDMAINEP1	Регистр управления DMA конечной точки 1 IN EP1
80_40B8h	DMADAINEP1	Дескриптор адреса DMA конечной точки 1 IN EP1
80_40BCh	STINEP1	Статусный регистр конечной точки 1 IN EP1
Конечная точка IN EP2		
80_40C0h	CTRLINEP2	Регистр управления конечной точкой 2 IN EP2
80_40C4h	CTRLDMAINEP2	Регистр управления DMA конечной точки 2 IN EP2
80_40C8h	DMADAINEP2	Дескриптор адреса DMA конечной точки 2 IN EP2
80_40CCh	STINEP2	Статусный регистр конечной точки 2 IN EP2
Конечная точка IN EP3		
80_40D0h	CTRLINEP3	Регистр управления конечной точкой 3 IN EP3
80_40D4h	CTRLDMAINEP3	Регистр управления DMA конечной точки 3 IN EP3
80_40D8h	DMADAINEP3	Дескриптор адреса DMA конечной точки 3 IN EP3
80_40DCh	STINEP3	Статусный регистр конечной точки 3 IN EP3
Общие регистры DC контроллера		
80_40E0h	GLBLCTRLDC	Регистр управления DC контроллера
80_40E4h	GLBLSTDC	Регистр состояния DC контроллера
80_40E5h – 80_40EFh	RSV	Зарезервировано
80_6000h – 80_6FFFh	USBRAM	ОЗУ USB контроллера (4K × 32) бит

Продолжение таблицы 4.3

1	2	3
MIL-STD-1553_0		
80_40F0h	IRQ0	Регистр прерываний
80_40F4h	IRQE0	Регистр разрешения прерываний
80_40F5h – 80_40FFh	RSV	Зарезервировано
80_4100h	HCNFGR0	Регистр аппаратной конфигурации
80_4104h	SELCLKMNCH0	Регистр выбора источника тактирования
80_4105h – 80_412Fh	RSV	Зарезервировано
КШ (контроллер шины)		
80_4130h	BCST&CNFG0	Регистр состояния и управления КШ
80_4134h	BCACT0	Регистр действий КШ
80_4138h	BCTRLNP0	Указатель следующего адреса списка передач регулярного расписания КШ
80_413Ch	DCALNP0	Указатель следующего адреса асинхронного списка передач КШ
80_4140h	BCTMR0	Регистр таймера КШ
80_4144h	BCTMRWUP0	Регистр таймера пробуждения КШ
80_4148h	BCTRIRQP0	Регистр позиции в кольце прерываний передачи КШ
80_414Ch	BCRTBSW0	Регистр замены шины УУ КШ
80_414Dh – 80_4157h	RSV	Зарезервировано
80_4158h	BCTRLCRNTSLTP0	Указатель дескриптора текущего слота времени КШ
80_415Ch	BCALCRNTSLTP0	Указатель дескриптора асинхронного списка текущего слота времени КШ
80_415Dh – 80_416Fh	RSV	Зарезервировано
УУ (удалённое устройство)		
80_4170h	RTSTR0	Регистр состояния УУ
80_4174h	RTCNFGR0	Регистр конфигурации УУ
80_4178h	RTBSTR0	Регистр состояния шины УУ
80_417Ch	RTSTWR0	Регистр слов состояния УУ
80_4180h	RTSYNR0	Регистр синхронизации УУ
80_4184h	RTSUBATBL0	Регистр базового адреса таблицы подадресов УУ
80_4188h	RTMDCNTRL0	Регистр управления кодами режимов УУ
80_4189h – 80_4193h	RSV	Зарезервировано
80_4194h	RTTIMECNTRLR0	Регистр управления тегом времени УУ
80_4195h – 80_419Bh	RSV	Зарезервировано
80_419Ch	RTEVLOGMSK0	Регистр маски журнала событий УУ
80_41A0h	RTEVLOGP0	Регистр позиции в журнале событий УУ
80_41A4h	RTEVLOGINRP0	Регистр позиции в журнале прерываний УУ
80_41A5h – 80_41AFh	RSV	Зарезервировано
МШ (монитор шины)		
80_41B0h	BMSTS0	Регистр состояния МШ
80_41B4h	BMCNTRL0	Регистр управления МШ
80_41B8h	BMRTAF0	Регистр фильтра адресов УУ МШ
80_41BCh	BMRTSUBAF0	Регистр фильтра подадресов УУ МШ
80_41C0h	BMRTMF0	Регистр фильтра кодов режимов УУ МШ
80_41C4h	BMLOGBST0	Регистр начала буфера журнала МШ
80_41C8h	BMLOGBE0	Регистр конца буфера журнала МШ
80_41CCh	BMLOGBP0	Регистр позиции буфера журнала МШ
80_41D0h	BMTIMECNTRL0	Регистр управления тегом времени МШ
80_41D1h – 80_41EFh	RSV	Зарезервировано
80_5000h – 80_57FFh	MNCH_RAM0	ОЗУ объёмом (2К × 32) бит

Продолжение таблицы 4.3

1	2	3
MIL-STD-1553_1		
80_41F0h	IRQ1	Регистр прерываний
80_41F4h	IRQE1	Регистр разрешения прерываний
80_41F5h – 80_40FFh	RSV	Зарезервировано
80_4200h	HCNFGR1	Регистр аппаратной конфигурации
80_4204h	SELCLKMNCH1	Регистр выбора источника тактирования
80_4205h – 80_422Fh	RSV	Зарезервировано
КШ (контроллер шины)		
80_4230h	BCST&CNFG1	Регистр состояния и управления КШ
80_4234h	BCACT1	Регистр действий КШ
80_4238h	BCTRLNP1	Указатель следующего адреса списка передач регулярного расписания КШ
80_423Ch	DCALNP1	Указатель следующего адреса асинхронного списка передач КШ
80_4240h	BCTMR1	Регистр таймера КШ
80_4244h	BCTMRWUP1	Регистр таймера пробуждения КШ
80_4248h	BCTRIRQP1	Регистр позиции в кольце прерываний передачи КШ
80_424Ch	BCRTBSW1	Регистр замены шины УУ КШ
80_424Dh – 80_4257h	RSV	Зарезервировано
80_4258h	BCTRLCRNTSLTP1	Указатель дескриптора текущего слота времени КШ
80_425Ch	BCALCRNTSLTP1	Указатель дескриптора асинхронного списка текущего слота времени КШ
80_425Dh – 80_426Fh	RSV	Зарезервировано
УУ (удалённое устройство)		
80_4270h	RTSTR1	Регистр состояния УУ
80_4274h	RTCNFGR1	Регистр конфигурации УУ
80_4278h	RTBSTR1	Регистр состояния шины УУ
80_427Ch	RTSTWR1	Регистр слов состояния УУ
80_4280h	RTSYNR1	Регистр синхронизации УУ
80_4284h	RTSUBATBL1	Регистр базового адреса таблицы подадресов УУ
80_4288h	RTMDCNTRL1	Регистр управления кодами режимов УУ
80_4289h – 80_4293h	RSV	Зарезервировано
80_4294h	RTTIMECNTRLR1	Регистр управления тегом времени УУ
80_4295h – 80_429Bh	RSV	Зарезервировано
80_429Ch	RTEVLOGMSK1	Регистр маски журнала событий УУ
80_42A0h	RTEVLOGP1	Регистр позиции в журнале событий УУ
80_42A4h	RTEVLOGINRP1	Регистр позиции в журнале прерываний УУ
80_42A5h – 80_42AFh	RSV	Зарезервировано
МШ (монитор шины)		
80_42B0h	BMSTS1	Регистр состояния МШ
80_42B4h	BMCNTRL1	Регистр управления МШ
80_42B8h	BMRTAF1	Регистр фильтра адресов УУ МШ
80_42BCh	BMRTSUBAF1	Регистр фильтра подадресов УУ МШ
80_42C0h	BMRTMF1	Регистр фильтра кодов режимов УУ МШ
80_42C4h	BMLOGBST1	Регистр начала буфера журнала МШ
80_42C8h	BMLOGBE1	Регистр конца буфера журнала МШ
80_42CCh	BMLOGBP1	Регистр позиции буфера журнала МШ
80_42D0h	BMTIMECNTRL1	Регистр управления тегом времени МШ
80_42D1h – 80_42EFh	RSV	Зарезервировано
80_5800h – 80_5FFFh	MNCH_RAM1	ОЗУ объёмом (2К × 32) бит

Продолжение таблицы 4.3

1	2	3
MIL-STD-1553_2		
80_42F0h	IRQ2	Регистр прерываний
80_42F4h	IRQE2	Регистр разрешения прерываний
80_42F5h – 80_42FFh	RSV	Зарезервировано
80_4300h	HCNFGR2	Регистр аппаратной конфигурации
80_4304h	SELCLKMNCH2	Регистр выбора источника тактирования
80_4305h – 80_432Fh	RSV	Зарезервировано
КШ (контроллер шины)		
80_4330h	BCST&CNFG2	Регистр состояния и управления КШ
80_4334h	BCACT2	Регистр действий КШ
80_4338h	BCTRLNP2	Указатель следующего адреса списка передач регулярного расписания КШ
80_433Ch	DCALNP2	Указатель следующего адреса асинхронного списка передач КШ
80_4340h	BCTMR2	Регистр таймера КШ
80_4344h	BCTMRWUP2	Регистр таймера пробуждения КШ
80_4348h	BCTRIRQP2	Регистр позиции в кольце прерываний передачи КШ
80_434Ch	BCRTBSW2	Регистр замены шины УУ КШ
80_434Dh – 80_4357h	RSV	Зарезервировано
80_4358h	BCTRLCRNTSLTP2	Указатель дескриптора текущего слота времени КШ
80_435Ch	BCALCRNTSLTP2	Указатель дескриптора асинхронного списка текущего слота времени КШ
80_435Dh – 80_436Fh	RSV	Зарезервировано
УУ (удалённое устройство)		
80_4370h	RTSTR2	Регистр состояния УУ
80_4374h	RTCNFGR2	Регистр конфигурации УУ
80_4378h	RTBSTR2	Регистр состояния шины УУ
80_437Ch	RTSTWR2	Регистр слов состояния УУ
80_4380h	RTSYNR2	Регистр синхронизации УУ
80_4384h	RTSUBATBL2	Регистр базового адреса таблицы подадресов УУ
80_4388h	RTMDCNTRL2	Регистр управления кодами режимов УУ
80_4389h – 80_4393h	RSV	Зарезервировано
80_4394h	RTTIMECNTRLR2	Регистр управления тегом времени УУ
80_4395h – 80_439Bh	RSV	Зарезервировано
80_439Ch	RTEVLOGMSK2	Регистр маски журнала событий УУ
80_43A0h	RTEVLOGP2	Регистр позиции в журнале событий УУ
80_43A4h	RTEVLOGINRP2	Регистр позиции в журнале прерываний УУ
80_43A5h – 80_43AFh	RSV	Зарезервировано
МШ (монитор шины)		
80_43B0h	BMSTS2	Регистр состояния МШ
80_43B4h	BMCNTRL2	Регистр управления МШ
80_43B8h	BMRTAF2	Регистр фильтра адресов УУ МШ
80_43BCh	BMRTSUBAF2	Регистр фильтра подадресов УУ МШ
80_43C0h	BMRTMF2	Регистр фильтра кодов режимов УУ МШ
80_43C4h	BMLOGBST2	Регистр начала буфера журнала МШ
80_43C8h	BMLOGBE2	Регистр конца буфера журнала МШ
80_43CCh	BMLOGBP2	Регистр позиции буфера журнала МШ
80_43D0h	BMTIMECNTRL2	Регистр управления тегом времени МШ
80_43D1h – 80_43EFh	RSV	Зарезервировано
80_7000h – 80_77FFh	MNCH_RAM2	ОЗУ объёмом (2K × 32) бит

Продолжение таблицы 4.3

1	2	3
MIL-STD-1553_3		
80_43F0h	IRQ3	Регистр прерываний
80_43F4h	IRQE3	Регистр разрешения прерываний
80_43F5h – 80_43FFh	RSV	Зарезервировано
80_4400h	HCNFGR3	Регистр аппаратной конфигурации
80_4404h	SELCLKMNCH3	Регистр выбора источника тактирования
80_4405h – 80_442Fh	RSV	Зарезервировано
КШ (контроллер шины)		
80_4430h	BCST&CNFG3	Регистр состояния и управления КШ
80_4434h	BCACT3	Регистр действий КШ
80_4438h	BCTRLNP3	Указатель следующего адреса списка передач регулярного расписания КШ
80_443Ch	DCALNP3	Указатель следующего адреса асинхронного списка передач КШ
80_4440h	BCTMR3	Регистр таймера КШ
80_4444h	BCTMRWUP3	Регистр таймера пробуждения КШ
80_4448h	BCTRIRQP3	Регистр позиции в кольце прерываний передачи КШ
80_444Ch	BCRTBSW3	Регистр замены шины УУ КШ
80_444Dh – 80_4457h	RSV	Зарезервировано
80_4458h	BCTRLCRNTSLTP3	Указатель дескриптора текущего слота времени КШ
80_445Ch	BCALCRNTSLTP3	Указатель дескриптора асинхронного списка текущего слота времени КШ
80_445Dh – 80_446Fh	RSV	Зарезервировано
УУ (удалённое устройство)		
80_4470h	RTSTR3	Регистр состояния УУ
80_4474h	RTCNFGR3	Регистр конфигурации УУ
80_4478h	RTBSTR3	Регистр состояния шины УУ
80_447Ch	RTSTWR3	Регистр слов состояния УУ
80_4480h	RTSYNR3	Регистр синхронизации УУ
80_4484h	RTSUBATBL3	Регистр базового адреса таблицы подадресов УУ
80_4488h	RTMDCNTRL3	Регистр управления кодами режимов УУ
80_4489h – 80_4493h	RSV	Зарезервировано
80_4494h	RTTIMECNTRLR3	Регистр управления тегом времени УУ
80_4495h – 80_419Bh	RSV	Зарезервировано
80_449Ch	RTEVLOGMSK3	Регистр маски журнала событий УУ
80_44A0h	RTEVLOGP3	Регистр позиции в журнале событий УУ
80_44A4h	RTEVLOGINRP3	Регистр позиции в журнале прерываний УУ
80_44A5h – 80_44AFh	RSV	Зарезервировано
МШ (монитор шины)		
80_44B0h	BMSTS3	Регистр состояния МШ
80_44B4h	BMCNTRL3	Регистр управления МШ
80_44B8h	BMRTAF3	Регистр фильтра адресов УУ МШ
80_44BCh	BMRTSUBAF3	Регистр фильтра подадресов УУ МШ
80_44C0h	BMRTMF3	Регистр фильтра кодов режимов УУ МШ
80_44C4h	BMLOGBST3	Регистр начала буфера журнала МШ
80_44C8h	BMLOGBE3	Регистр конца буфера журнала МШ
80_44CCh	BMLOGBP3	Регистр позиции буфера журнала МШ
80_44D0h	BMTIMECNTRL3	Регистр управления тегом времени МШ
80_44D1h – 80_44EFh	RSV	Зарезервировано
80_7800h – 80_7FFFh	MNCH_RAM3	ОЗУ объёмом (2К × 32) бит

Продолжение таблицы 4.3

1	2	3
GPIO_0		
80_44F0h	IOPDATINR0	Регистр входных данных порта ввода-вывода GPIO_0
80_44F1h	IOPDATOUTR0	Регистр выходных данных порта ввода-вывода GPIO_0
80_44F2h	IOPDIRR0	Регистр направления порта ввода-вывода GPIO_0
80_44F3h	INTMSKR0	Регистр маски прерываний GPIO_0
80_44F4h	INTPOLR0	Регистр полярности прерываний GPIO_0
80_44F5h	INTEDGR0	Регистр фронта прерываний GPIO_0
80_44F6h	BPSR0	Регистр обхода GPIO_0; не используется
80_44F7h	CPBLR0	Регистр возможностей GPIO_0; не используется
80_44F8h – 80_44FFh	RSV	Зарезервировано
80_4500h	INTFR0	Регистр флагов прерываний GPIO_0
80_4501h – 80_451Fh	RSV	Зарезервировано
80_4520h – 80_453Ch	MAPR0	Регистр(ы) карты прерываний GPIO_0; зарезервировано
80_453Dh – 80_453Fh	RSV	Зарезервировано
GPIO_1		
80_4540h	IOPDATINR1	Регистр входных данных порта ввода-вывода GPIO_1
80_4541h	IOPDATOUTR1	Регистр выходных данных порта ввода-вывода GPIO_1
80_4542h	IOPDIRR1	Регистр направления порта ввода-вывода GPIO_1
80_4543h	INTMSKR1	Регистр маски прерываний GPIO_1
80_4544h	INTPOLR1	Регистр полярности прерываний GPIO_1
80_4545h	INTEDGR1	Регистр фронта прерываний GPIO_1
80_4546h	BPSR1	Регистр обхода GPIO_1; не используется
80_4547h	CPBLR1	Регистр возможностей GPIO_1; не используется
80_4548h – 80_454Fh	RSV	Зарезервировано
80_4550h	INTFR1	Регистр флагов прерываний GPIO_1
80_4551h – 80_455Fh	RSV	Зарезервировано
80_4560h – 80_457Ch	MAPR1	Регистр(ы) карты прерываний GPIO_1; зарезервировано
80_457Dh – 80_457Fh	RSV	Зарезервировано
Коммутатор		
80_4580h	RS_GPIO0	Регистр семафора GPIO_0
80_4581h	RC_GPIO0	Регистр коммутации GPIO_0
80_4582h	RS_GPIO1	Регистр семафора GPIO_1
80_4583h	RC_GPIO1	Регистр коммутации GPIO_1
80_4584h	RS_MNCH0	Регистр семафора MIL-STD-1553_0
80_4585h	RC_MNCH0	Регистр коммутации MIL-STD-1553_0
80_4586h	RS_MNCH1	Регистр семафора MIL-STD-1553_1
80_4587h	RC_MNCH1	Регистр коммутации MIL-STD-1553_1
80_4588h	RS_MNCH2	Регистр семафора MIL-STD-1553_2
80_4589h	RC_MNCH2	Регистр коммутации MIL-STD-1553_2
80_458Ah	RS_MNCH3	Регистр семафора MIL-STD-1553_3
80_458Bh	RC_MNCH3	Регистр коммутации MIL-STD-1553_3
80_458Ch	RS_USB	Регистр семафора USB 2.0
80_458Dh	RC_USB	Регистр коммутации USB 2.0
80_458Eh	RS_ARINC429	Регистр семафора ARINC-429
80_458Fh	RC_ARINC429	Регистр коммутации ARINC-429
80_4590h	RS_UART0	Регистр семафора UART_0
80_4591h	RC_UART0	Регистр коммутации UART_0

Окончание таблицы 4.3

1	2	3
80_4592h	RS_UART1	Регистр семафора UART_1
80_4593h	RC_UART1	Регистр коммутации UART_1
80_4594h	RS_SHMEM	Регистр семафора критической секции разделяемой памяти
80_4595h	RC_SHMEM	Регистр коммутации критической секции разделяемой памяти
80_4596h	RS_TECHBUS	Регистр семафора TechBus
80_4597h	RC_TECHBUS	Регистр коммутации TechBus
80_4598h	RS_FNC	Регистр семафора ПЧК
80_4599h	RC_FNC	Регистр коммутации ПЧК
80_459Ah – 80_4612h	RSV	Зарезервировано
TechBus		
80_4613h	PRMR	Регистр параметров записи/чтения по шине TechBus
80_4614h	ADDR	Регистр адреса, передаваемого на TechBus
80_4614h	DSWR	Регистр данных чтения/записи по шине TechBus
80_4616h	RK	Регистр команд на шине TechBus
80_4617h – 80_467Fh	RSV	Зарезервировано
ПЧК		
80_4680h	SUX	Суммы по UX
80_4681h	SUY	Суммы по UY
80_4682h	SUZ	Суммы по UZ
80_4683h	DUX	Дельты по UX
80_4684h	DUY	Дельты по UY
80_4685h	DUZ	Дельты по UZ
80_4686h – 80_4687h	RSV	Зарезервировано. Запись и чтение из регистров разрешена
80_4688h	DPX	Дельты по PX
80_4689h	DPY	Дельты по PY
80_468Ah	DPZ	Дельты по PZ
80_468Bh	RSV	Зарезервировано. Запись и чтение из регистра разрешена
80_468Ch	TP	Значение времени на интервале 12 мс
80_468Dh	VC	Вектор
80_468Eh – 80_46FFh	RSV	Зарезервировано
Примечание – Попытка обращения к резервным или несуществующим областям памяти может привести к непредсказуемому ходу исполнения программы.		

Регистровый файл ядра ИС 1867ВН016

Регистровый файл ядра ИС 1867ВН016 не претерпел особых изменений по сравнению с процессором-прототипом 1867ВЦ6Ф, за исключением модифицированных, в связи с расширением системы прерываний, регистров разрешения (*A*IE) и флагов (*A*IF) прерываний. Подробное описание регистров *A*IE и *A*IF приведено в подразделе 4.2 настоящего документа.

Таблица 4.4 – Регистровый файл ядра ИС 1867ВН016

Название регистра	Машинный адрес (hex)[1)]	Функциональное назначение
R0	00h	Регистр повышенной точности 0
R1	01h	Регистр повышенной точности 1
R2	02h	Регистр повышенной точности 2
R3	03h	Регистр повышенной точности 3
R4	04h	Регистр повышенной точности 4
R5	05h	Регистр повышенной точности 5
R6	06h	Регистр повышенной точности 6
R7	07h	Регистр повышенной точности 7
AR0	08h	Вспомогательный регистр 0
AR1	09h	Вспомогательный регистр 1
AR2	0Ah	Вспомогательный регистр 2
AR3	0Bh	Вспомогательный регистр 3
AR4	0Ch	Вспомогательный регистр 4
AR5	0Dh	Вспомогательный регистр 5
AR6	0Eh	Вспомогательный регистр 6
AR7	0Fh	Вспомогательный регистр 7
DP	10h	Указатель страницы данных
IR0	11h	Индексный регистр 0
IR1	12h	Индексный регистр 1
BK	13h	Регистр размера блока
SP	14h	Указатель системного стека
ST	15h	Регистр состояния
*A*IE[2)]	16h	Регистр разрешения прерываний ЦПУ/ПДП (модифицированный)
*A*IF[2)]	17h	Регистр флагов прерываний ЦПУ (модифицированный)
IOF	18h	Флаги ввода-вывода
RS	19h	Регистр адреса начала повторений
RE	1Ah	Регистр адреса конца повторений
RC	1Bh	Счётчик повторений

[1)] Машинный адрес – получаемое при ассемблировании шестнадцатеричное значение, кодирующее идентификатор регистра в соответствующем поле объектного кода команд работы с регистрами (не является физическим адресом памяти ЦПУ).

[2)] Условно обозначены расширенные (модифицированные по сравнению с ИС 1867ВЦ6Ф) регистры разрешения и флагов прерываний, соответственно:

- *A*IE – *Advanced* Interrupt Enable,
- *A*IF – *Advanced* Interrupt Flags.

При разработке программ на языке ассемблера необходимо использовать стандартные имена (аббревиатуры) для этих регистров – IE и IF соответственно.

4.2 Расширенная система прерываний ИС 1867ВН016

В таблице 4.5 приведена карта векторов сброса/прерываний ядра ИС 1867ВН016.

Таблица 4.5 – Карта векторов сброса/прерываний ядра ИС 1867ВН016

Адрес вектора	Имя	Функция
1	2	3
Векторы прерываний от внешних источников и неразделяемых (внутриядерных) периферийных устройств		
00h	RESET	Прерывание от RESET_COMM#
01h	INT0/INTTB	Прерывание от вывода (выводов) INT0#_*n*, если бит IIR.TB_INT0 = 0 Прерывание от TechBus, если бит IIR.TB_INT0 = 1
02h	INT1/INTPCHK	Прерывание от вывода (выводов) INT1#_*n*, если бит IIR.PCHK_INT1 = 0 Прерывание от ПЧК, если бит IIR.PCHK_INT1 = 1
03h	INT2	Прерывание от вывода (выводов) INT2#_*n*
04h	INT3	Прерывание от вывода (выводов) INT3#_*n*
05h	XINT0	Прерывание от последовательного порта 0, если буфер передатчика пуст
06h	RINT0	Прерывание от последовательного порта 0, если буфер приёмника полон
07h	XINT1	Прерывание от последовательного порта 1, если буфер передатчика пуст
08h	RINT1	Прерывание от последовательного порта 1, если буфер приёмника полон
09h	TINT0	Прерывание, генерируемое таймером 0
0Ah	TINT1	Прерывание, генерируемое таймером 1
0Bh	DINT	Прерывание, генерируемое контроллером DMA
Векторы прерываний разделяемых между ядрами периферийных устройств (п/у)		
0Ch	ACKN/TB	Прерывание от смежного процессора по линии ACKN, если бит IIR.TB_INT0 = 1. Прерывание от TechBus, если бит IIR.TB_INT0 = 0
0Dh	RQVST/PCHK	Прерывание от смежного процессора по линии RQVST, если бит IIR.PCHK_INT1 = 1. Прерывание от ПЧК, если бит IIR.PCHK_INT1 = 0
0Eh	INTINTR	Запрос/Подтверждение обслуживания
0Fh	MILStd1553_0	Прерывание от п/у MIL-STD-1553_0
10h	MILStd1553_1	Прерывание от п/у MIL-STD-1553_1
11h	MILStd1553_2	Прерывание от п/у MIL-STD-1553_2
12h	MILStd1553_3	Прерывание от п/у MIL-STD-1553_3
13h	USB 2.0	Прерывание от п/у USB 2.0
14h	ARINCCH0	Прерывание от канала 0 (приём/передача) п/у ARINC-429
15h	ARINCCH1	Прерывание от канала 1 (приём/передача) п/у ARINC-429
16h	ARINCCH2	Прерывание от канала 2 (приём/передача) п/у ARINC-429
17h	ARINCCH3	Прерывание от канала 3 (приём/передача) п/у ARINC-429
18h	ARINCCH4	Прерывание от канала 4 (приём/передача) п/у ARINC-429
19h	ARINCCH5	Прерывание от канала 5 (приём/передача) п/у ARINC-429
1Ah	ARINCCH6	Прерывание от канала 6 (приём/передача) п/у ARINC-429
1Bh	ARINCCH7	Прерывание от канала 7 (приём/передача) п/у ARINC-429
1Ch	UART0	Прерывание от п/у UART_0
1Dh	UART1	Прерывание от п/у UART_1
1Eh	GPIO0	Прерывание от входов/выходов общего назначения GPIO_0
1Fh	GPIO1	Прерывание от входов/выходов общего назначения GPIO_1

Окончание таблицы 4.5

1	2	3
Векторы прерываний инструкции TRAP		
20h	TRAP 0	Внутреннее прерывание, генерируемое инструкцией TRAP 0
21h	TRAP 1	Внутреннее прерывание, генерируемое инструкцией TRAP 1
22h	TRAP 2	Внутреннее прерывание, генерируемое инструкцией TRAP 2
23h	TRAP 3	Внутреннее прерывание, генерируемое инструкцией TRAP 3
24h	TRAP 4	Внутреннее прерывание, генерируемое инструкцией TRAP 4
25h	TRAP 5	Внутреннее прерывание, генерируемое инструкцией TRAP 5
26h	TRAP 6	Внутреннее прерывание, генерируемое инструкцией TRAP 6
27h	TRAP 7	Внутреннее прерывание, генерируемое инструкцией TRAP 7
28h	TRAP 8	Внутреннее прерывание, генерируемое инструкцией TRAP 8
29h	TRAP 9	Внутреннее прерывание, генерируемое инструкцией TRAP 9
2Ah	TRAP 10	Внутреннее прерывание, генерируемое инструкцией TRAP 10
2Bh	TRAP 11	Внутреннее прерывание, генерируемое инструкцией TRAP 11
2Ch	TRAP 12	Внутреннее прерывание, генерируемое инструкцией TRAP 12
2Dh	TRAP 13	Внутреннее прерывание, генерируемое инструкцией TRAP 13
2Eh	TRAP 14	Внутреннее прерывание, генерируемое инструкцией TRAP 14
2Fh	TRAP 15	Внутреннее прерывание, генерируемое инструкцией TRAP 15
30h	TRAP 16	Внутреннее прерывание, генерируемое инструкцией TRAP 16
31h	TRAP 17	Внутреннее прерывание, генерируемое инструкцией TRAP 17
32h	TRAP 18	Внутреннее прерывание, генерируемое инструкцией TRAP 18
33h	TRAP 19	Внутреннее прерывание, генерируемое инструкцией TRAP 19
34h	TRAP 20	Внутреннее прерывание, генерируемое инструкцией TRAP 20
35h	TRAP 21	Внутреннее прерывание, генерируемое инструкцией TRAP 21
36h	TRAP 22	Внутреннее прерывание, генерируемое инструкцией TRAP 22
37h	TRAP 23	Внутреннее прерывание, генерируемое инструкцией TRAP 23
38h	TRAP 24	Внутреннее прерывание, генерируемое инструкцией TRAP 24
39h	TRAP 25	Внутреннее прерывание, генерируемое инструкцией TRAP 25
3Ah	TRAP 26	Внутреннее прерывание, генерируемое инструкцией TRAP 26
3Bh	TRAP 27	Внутреннее прерывание, генерируемое инструкцией TRAP 27
3Ch	TRAP 28	Внутреннее прерывание, генерируемое инструкцией TRAP 28
3Dh	TRAP 29	Внутреннее прерывание, генерируемое инструкцией TRAP 29
3Eh	TRAP 30	Внутреннее прерывание, генерируемое инструкцией TRAP 30
3Fh	TRAP 31	Внутреннее прерывание, генерируемое инструкцией TRAP 31
40h – BFh	RSV	Зарезервировано

Адреса векторов сброса/прерываний/системных прерываний распределены от 0h до 3Fh. Векторы прерываний содержат начальный адрес подпрограммы обработки прерывания, с которого начнётся исполнение, если прерывание произошло. Например, при сбросе содержимое ячейки памяти 0h (вектор прерывания), загружается в PC, и выполнение начинается с этого адреса.

Системные прерывания 28–31 зарезервированы и не должны использоваться!

На рисунке 4.1 приведена структура, а в таблице 4.6 – описание разрядов расширенного регистра разрешения прерываний CPU/DMA (*A*IE).

31	30	29	28	27	26	25	24
RSV	RSV	EINTRNLINT (DMA)	ERQST (DMA)/ PCHK	EACKN (DMA)/ TB	EDINT (DMA)	ETINT1 (DMA)	ETINT0 (DMA)
R-0	R-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0
23	22	21	20	19	18	17	16
ERINT1 (DMA)	EXINT1 (DMA)	ERINT0 (DMA)	EXINT0 (DMA)	EINT3 (DMA)	EINT2 (DMA)	EINT1 (DMA)	EINT0 (DMA)
RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0
15	14	13	12	11	10	9	8
RSV	RSV	EINTRNLINT (CPU)	ERQST (CPU)/ PCHK	EACKN (CPU)/ TB	EDINT (CPU)	ETINT1 (CPU)	ETINT0 (CPU)
R-0	R-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0
7	6	5	4	3	2	1	0
ERINT1 (CPU)	EXINT1 (CPU)	ERINT0 (CPU)	EXINT0 (CPU)	EINT3 (CPU)	EINT2 (CPU)	EINT1 (CPU)	EINT0 (CPU)
RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0

Примечание – Принятые условные обозначения: R – бит только читается, RWC – бит доступен на чтение, запись и сбрасывается при сбросе.

Рисунок 4.1 – Структура расширенного регистра разрешения прерываний CPU/DMA (*A*IE)

Таблица 4.6 – Описание битов регистра *A*IE

Бит	Обозначение	Описание
1	2	3
31 – 30	RSV	Зарезервировано
29	EINTRNLINT (DMA)	Внутреннее прерывание DMA при поступлении сигнала INTRNLINT от смежного процессора: - EINTRNLINT (DMA) = 1 – прерывание DMA разрешено; - EINTRNLINT (DMA) = 0 – прерывание DMA запрещено.
28	ERQST (DMA)/ PCHK	Разрешение прерывания DMA при поступлении сигнала RQST/PCHK от смежного процессора: - ERQST (DMA) = 1 – прерывание DMA разрешено; - ERQST (DMA) = 0 – прерывание DMA запрещено.
27	EACKN (DMA)/ TB	Разрешение прерывания DMA при поступлении сигнала ACKN/TB от смежного процессора: - EACKN (DMA) = 1 – прерывание DMA разрешено; - EACKN (DMA) = 0 – прерывание DMA запрещено.
26 – 16	*IE*	Назначение битов соответствует назначению битов регистра IE (см. приложение А, таблица А.3.3).
15 – 14	RSV	Зарезервировано.
13	EINTRNLINT (CPU)	Внутреннее прерывание CPU при поступлении сигнала INTRNLINT от смежного процессора: - EINTRNLINT (CPU) = 1 – прерывание DMA разрешено; - EINTRNLINT (CPU) = 0 – прерывание DMA запрещено.

Окончание таблицы 4.6

1	2	3
12	ERQST (CPU)/ PCHK	Разрешение прерывания CPU при поступлении сигнала RQST/PCHK от смежного процессора: - ERQST (CPU) = 1 – прерывание CPU разрешено; - ERQST (CPU) = 0 – прерывание CPU запрещено.
11	EACKN (CPU)/ TB	Разрешение прерывания CPU при поступлении сигнала ACKN/TB от смежного процессора: - EACKN (CPU) = 1 – прерывание CPU разрешено; - EACKN (CPU) = 0 – прерывание CPU запрещено.
10 – 0	*IE*	Назначение битов соответствует назначению битов регистра IE (см. приложение А, таблица А.3.3).

На рисунке 4.2 приведена структура, а в таблице 4.7 – описание разрядов расширенного регистра флагов прерываний CPU/DMA (*A*IF).

31	30	29	28	27	26	25	24
RSV	GPIO1	GPIO0	UART1	UART0	ARINCCH7	ARINCCH6	ARINCCH5
R-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0
23	22	21	20	19	18	17	16
ARINC CH4	ARINC CH3	ARINC CH2	ARINC CH1	ARINC CH0	USB	MILStd_3	MILStd_2
RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0
15	14	13	12	11	10	9	8
MILStd_1	MILStd_0	FINTRNL INT	FRQST/ PCHK	FACKN/TB	DINT	TINT1	TINT0
RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0
7	6	5	4	3	2	1	0
RINT1	XINT1	RINT0	XINT0	INT3	INT2	INT1/PCHK	INT0/TB
RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0

Примечание – Принятые условные обозначения: R – бит только читается, RWC – бит доступен на чтение, запись и сбрасывается при сбросе.

Рисунок 4.2 – Структура расширенного регистра флагов прерываний CPU/DMA (*A*IF)

Таблица 4.7 – Описание битов регистра *A*IF

Бит	Обозначение	Описание
31	RSV	Зарезервировано
30	GPIO1	Флаг прерывания от устройства GPIO1
29	GPIO0	Флаг прерывания от устройства GPIO0
28	UART1	Флаг прерывания от устройства UART_1
27	UART0	Флаг прерывания от устройства UART_0
26	ARINCCH7	Флаг прерывания от канала 7 устройства ARINC-429
25	ARINCCH6	Флаг прерывания от канала 6 устройства ARINC-429
24	ARINCCH5	Флаг прерывания от канала 5 устройства ARINC-429
23	ARINCCH4	Флаг прерывания от канала 4 устройства ARINC-429
22	ARINCCH3	Флаг прерывания от канала 3 устройства ARINC-429
21	ARINCCH2	Флаг прерывания от канала 2 устройства ARINC-429
20	ARINCCH1	Флаг прерывания от канала 1 устройства ARINC-429
19	ARINCCH0	Флаг прерывания от канала 0 устройства ARINC-429
18	USB	Флаг прерывания от устройства USB
17	MILStd_3	Флаг прерывания от устройства MIL-STD-1553_3
16	MILStd_2	Флаг прерывания от устройства MIL-STD-1553_2
15	MILStd_1	Флаг прерывания от устройства MIL-STD-1553_1
14	MILStd_0	Флаг прерывания от устройства MIL-STD-1553_0
13	FINTRNLINT	Флаг прерывания от смежного процессора по линии INTRNLINT
12	FRQST/PCHK	Флаг прерывания от устройства ПЧК, если бит IIR.PCHK_INT1 = 1 и по линии RQST от смежного процессора, если бит IIR.PCHK_INT1 = 0
11	FACKN/TB	Флаг прерывания от устройства TechBus, если бит IIR.TB_INT0 = 1 и по линии ACKN от смежного процессора, если бит IIR.TB_INT0 = 0
10	DINT	Флаг прерывания от устройства DMA
9	TINT1	Флаг прерывания от таймера 1
8	TINT0	Флаг прерывания от таймера 0
7	RINT1	Флаг прерывания от приёмника последовательного порта 1
6	XINT1	Флаг прерывания от передатчика последовательного порта 1
5	RINT0	Флаг прерывания от приёмника последовательного порта 0
4	XINT0	Флаг прерывания от передатчика последовательного порта 0
3	INT3	Флаг прерывания от внешнего входа (входов) INT3#_*n*
2	INT2	Флаг прерывания от внешнего входа (входов) INT2#_*n*
1	INT1/PCHK	Флаг прерывания от внешнего входа (входов) INT1#_*n*, если бит IIR.PCHK_INT1 = 0. Флаг прерывания от устройства ПЧК, если бит IIR.PCHK_INT1 = 1
0	INT0/TB	Флаг прерывания от внешнего входа (входов) INT0#_*n*, если бит IIR.TB_INT0 = 0. Флаг прерывания от устройства TechBus, если бит IIR.TB_INT0 = 1

Примечание – Если флаг установлен устройством или программой в «1», то он будет находиться в этом состоянии до тех пор, пока прерывание от этого флага не обслужится процессорным ядром или этот флаг не сбросится программой в «0». После сброса флаг находится в «0».

На рисунке 4.3 приведена структура, а в таблице 4.8 – описание разрядов регистра управления внутренними прерываниями (IIR).

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
15	14	13	12	11	10	9	8
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
7	6	5	4	3	2	1	0
RSV	SROM MODE	SRAM MODE	PCHK_ INT1	TB_INT0	INTINTR	RQST	ACKN
R-X	RWC-1	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0

Примечание – Принятые условные обозначения: R – бит только читается, RWC – бит доступен на чтение, запись и сбрасывается при сбросе, X – значение не определено.

Рисунок 4.3 – Структура регистра управления линиями прерывания смежного процессора IIR /адрес **80_4701**h/

Таблица 4.8 – Описание битов регистра IIR

Бит	Обозначение	Описание
1	2	3
31–7	RSV	Зарезервировано
6	SROMMODE	Бит управления режимом работы внешним ROM: - SROMMODE = 0 – WaitState не добавляется при обращении к внешнему ROM; - SROMMODE = 1 – добавляется один WaitState при обращении к внешнему ROM. После системного сброса SROMMODE = 1.
5	SRAMMODE	Бит управления режимом работы INTSRAM: - SRAMMODE = 0 – WaitState не добавляется при обращении к INTSRAM; - SRAMMODE = 1 – добавляется один WaitState при обращении к INTSRAM. После системного сброса SRAMMODE = 0.
4	PCHK_INT1	- PCHK_INT1 = 0 – прерывание от внешнего вывода INT1# осуществляется по адресу 02h, прерывание от ПЧК осуществляется по адресу 0Dh, а прерывание по линии RQVST от смежного процессора блокируется. - PCHK_INT1 = 1 – прерывание от внешнего вывода INT1# блокируется, прерывание от ПЧК осуществляется по адресу 02h, а прерывание по лини RQVST от смежного процессора осуществляется по адресу 0Dh. После системного сброса бит PCHK_INT1 = 0.
3	TB_INT0	- TB_INT0 = 0 – прерывание от внешнего вывода INT0# осуществляется по адресу 01h, прерывание от TechBus (вывод ZPR_TB) осуществляется по адресу 0Dh, а прерывание по линии RQVST блокируется. - TB_INT0 = 1 – прерывание от внешнего вывода INT0# блокируется, прерывание от TechBus осуществляется по адресу 02h, а прерывание по лини RQVST от смежного процессора осуществляется по адресу 0Dh. После системного сброса бит TB_INT0 = 0.
2	INTINTR	Бит управления линией прерывания смежного процессора INTRLINE. Запись «1» в этот бит формирует сигнал прерывания (если оно разрешено!) с вектором 0Eh на смежный процессор по линии INTRLINE. Запись «0» в этот бит не приводит к формированию прерывания в смежном процессоре. После системного сброса бит INTINTR = 0.

Окончание таблицы 4.8

1	2	3
1	RQST	Бит управления линией RQST. Запись «1» в этот бит формирует сигнал на смежный процессор по линии RQST. Если бит IIR.PCHK_INT1 смежного процессора установлен в «1», то произойдёт прерывание смежного процессора по линии RQST и не произойдёт в противном случае. Запись «0» в этот бит не приводит к выполнению каких-либо действий. После системного сброса бит RQST = 0.
0	ACKN	Бит управления линией ACKN. Запись «1» в этот бит формирует сигнал на смежный процессор по линии ACKN. Если бит IIR.TB_INT0 смежного процессора установлен в «1», то произойдёт прерывание смежного процессора по линии ACKN и не произойдёт в противном случае. Запись «0» в этот бит не приводит к выполнению каких-либо действий. После системного сброса бит ACKN = 0.

4.3 Загрузчик пользовательской программы и регистр номера процессора

Начальный загрузчик (bootloader) для процессорных ядер 0 и 1 ИС 1867ВН016 (далее – загрузчик) 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 параллельно (через Primary Bus) или последовательно (через UART или MIL-STD-1553). Загрузчик активируется подачей сигнала аппаратного сброса (RESET_COMM#) при условии, что на входе MCBL/MP#_*n* соответствующего ядра установлен высокий уровень. При сбросе все прерывания глобально запрещаются, это позволило наделить входы внешних прерываний INT3#_*n* – INT1#_*n* альтернативной функциональностью – с их помощью выбирается один из восьми возможных параллельных или последовательных вариантов загрузки (см. рисунок 4.4 и таблицу 4.9).

Описание загрузчика

Для старта загрузчика необходимо, чтобы во время системного сброса вывод(ы) MCBL/MP#_*n* был(и) установлен(ы) в высокий уровень (режим микрокомпьютер-bootloader). Источник загрузки пользовательской программы в ту или иную область памяти (место назначения) ядер 0 и/или 1 определяется состоянием выводов INT3#_*n* – INT1#_*n* соответствующего ядра после системного сброса ИС. Само(и) место(а) назначения (одна или несколько областей оперативной памяти, которые могут быть несмежными) задаётся пользователем в загружаемом программном коде. Программа, под управлением которой осуществляется загрузка, «зашита» (маской, в процессе производства кристаллов) во внутреннем ПЗУ каждого ядра в диапазоне адресов 00_00C0h – 00_0FFFh. Стартовая ячейка 00_0000h содержит команду перехода на адрес 00_00C0h, начиная с которой и расположено собственно тело загрузчика.

В ячейках с 00_0001h по 00_003Fh (поле векторов прерываний) располагаются команды переходов (аппаратный ремэпинг) на адреса 80_A001h – 80_A03Fh соответственно. В этих адресах пользовательского кода должны быть прописаны переходы на подпрограммы обработки соответствующих прерываний.

Загрузка пользовательской программы производится поблочно. Каждый блок загружаемого кода предваряется «служебной» (не записываемой в память назначения) информацией, указывающей загрузчику на размер блока (количество 32-разрядных слов, которые будут записаны по месту назначения) и стартовый адрес назначения. Максимальный размер блока составляет 16М слов. Нулевой размер блока означает окончание загрузки данных. Загружаемый файл должен иметь структуру, которая приведена в таблице 4.10.

Пользовательская программа может быть загружена в различные области адресного пространства. Каждый блок загружаемой программы имеет свой размер и свой стартовый адрес загрузки. Загружаемая программа заканчивается словом из всех нулей (0000_0000h); далее указывается адрес размещения команды IACK, формирующей сигнал IACK#, который сигнализирует о завершении работы загрузчика.

Процессорные ядра ИС 1867ВН016 исполняют 32-разрядный программный код. Если загрузка пользовательской программы осуществляется 32-битными словами (такой вариант возможен только при параллельном способе загрузки), то каждое слово потока данных попадает в свою ячейку области назначения за один приём, бит-в-бит. Наряду с 32-битными, загрузчик позволяет оперировать с потоком из 16- или из 8-битных слов.

В 16-битном варианте (параллельном или через интерфейс MIL-STD-1553) 32-разрядное слово формируется в ячейке области назначения в два приёма: первыми в посылке передаются младшие 16 разрядов, заполняя разряды 15 – 0; затем передаётся старшее полуслово, заполняющее в ячейке области назначения разряды 31 – 16.

8-битный вариант (параллельно или через UART) формирует 32-разрядное слово назначения в четыре приёма: первый переданный байт заполняет разряды 7 – 0, второй байт – разряды 15 – 8, третий – разряды 23 – 16 и четвёртый – разряды 31 – 24 области назначения.

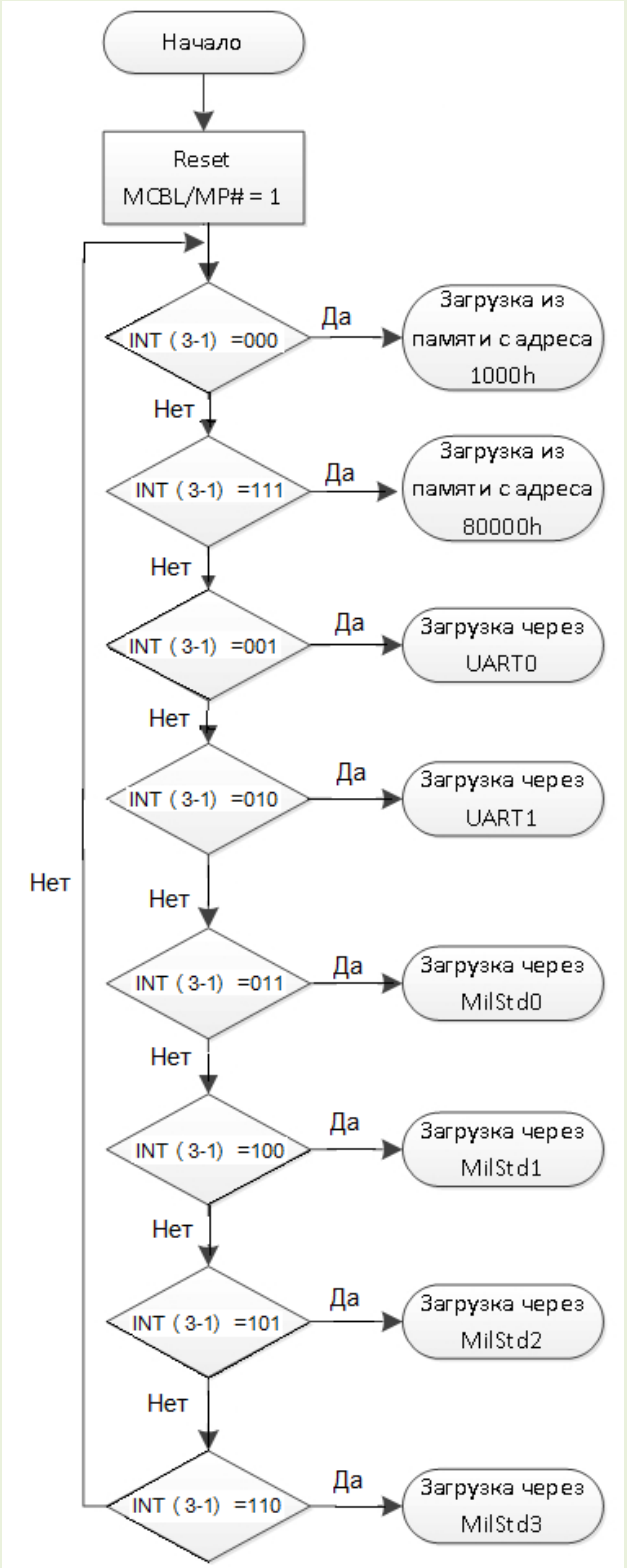


Рисунок 4.4 – Выбор режима загрузки в зависимости от состояния выводов внешних прерываний INT3#_*n* – INT1#_*n* (*n* – номер ядра)

Таблица 4.9 – Варианты загрузки для процессоров 0 и 1

Номер по порядку	INT3#_*n*	INT2#_*n*	INT1#_*n*	Источник загрузки
1	0	0	0	Загрузка с Primary Bus из адреса 00_1000h
2	0	0	1	Загрузка через UART_0. Первоначальная скорость загрузки равна 9600 бит/с
3	0	1	0	Загрузка через UART_1. Первоначальная скорость загрузки равна 9600 бит/с
4	0	1	1	Загрузка через MIL-STD-1553_0
5	1	0	0	Загрузка через MIL-STD-1553_1
6	1	0	1	Загрузка через MIL-STD-1553_2
7	1	1	0	Загрузка через MIL-STD-1553_3
8	1	1	1	Загрузка с Primary Bus из адреса 08_0000h
Примечание – *n* – номер процессорного ядра: 0, 1.				

Таблица 4.10 – Структура загружаемой пользовательской программы

Номер слова	Описание обработки слова данных
1	Ширина (разрядность) загружаемых данных: - 08h – 8-разрядные данные; - 10h – 16-разрядные данные; - 20h – 32-разрядные данные
2	Программирование регистра Expansion Bus Control Register (адрес: 80_8060h)
3	Программирование регистра Primary Bus Control Register (адрес: 80_8064h)
4	***Размер первого блока.*** ***Размер блока равен числу 32-битных слов в блоке данных.*** ***Нулевой размер блока означает окончание загрузки данных***
5	***Стартовый адрес назначения,*** ***начиная с которого будет загружаться первый блок данных***
6	**Первое слово первого блока данных**
…	**…**
n	**Последнее слово первого блока данных**
…	….
…	….
…	….
m	***Размер последнего блока.*** ***Размер блока равен числу 32-битных слов в блоке данных*** ***Нулевой размер блока означает окончание загрузки данных***
m+1	***Стартовый адрес назначения,*** ***начиная с которого будет загружаться последний блок данных***
m+2	**Первое слово последнего блока данных**
…	**…**
j	**Последнее слово последнего блока данных**
j+1	Последнее слово пользовательской программы (все нули – 0000_0000h)
j+2	Адрес ячейки памяти для инструкции IACK. Команда IACK формирует сигнал IACK#, который сигнализирует о конце загрузки
Примечания 1 **Заштрихованные строки** соответствуют одному блоку загружаемой программы. 2 Два первых слова в каждом блоке (***выделены курсивом***) несут служебную информацию и в память назначения не записываются.	

Разрядность данных, предназначенных для загрузки, указывается в 16-ричном коде в первом слове посылки (см. структуру загружаемых данных в зависимости от их разрядности: 32 бита – 20h, 16 бит – 10h и 8 бит – 8h в таблицах 4.11, 4.12 и 4.13 соответственно).

Таблица 4.11 – Пример структуры данных загрузки с разрядностью 32 бита

Адрес	Значение	Описание
1000h	0000_0020h	Ширина данных – 32 бита
1001h	0000_1058h	Программирование EBCR – регистра управления Expansion Bus (в данном примере: поле SWW = 11; поле WTCNT = 2)
1002h	0000_1058h	Программирование PBCR – регистра управления Primary Bus (в данном примере: поле SWW = 11; поле WTCNT = 2)
1003h	0000_01FFh	Размер программного блока (1FF – 512 32-битных слов в блоке данных)
1004h	0080_9C00h	Стартовый адрес назначения загружаемых данных – 80_9C00h

Таблица 4.12 – Пример структуры данных загрузки с разрядностью 16 бит

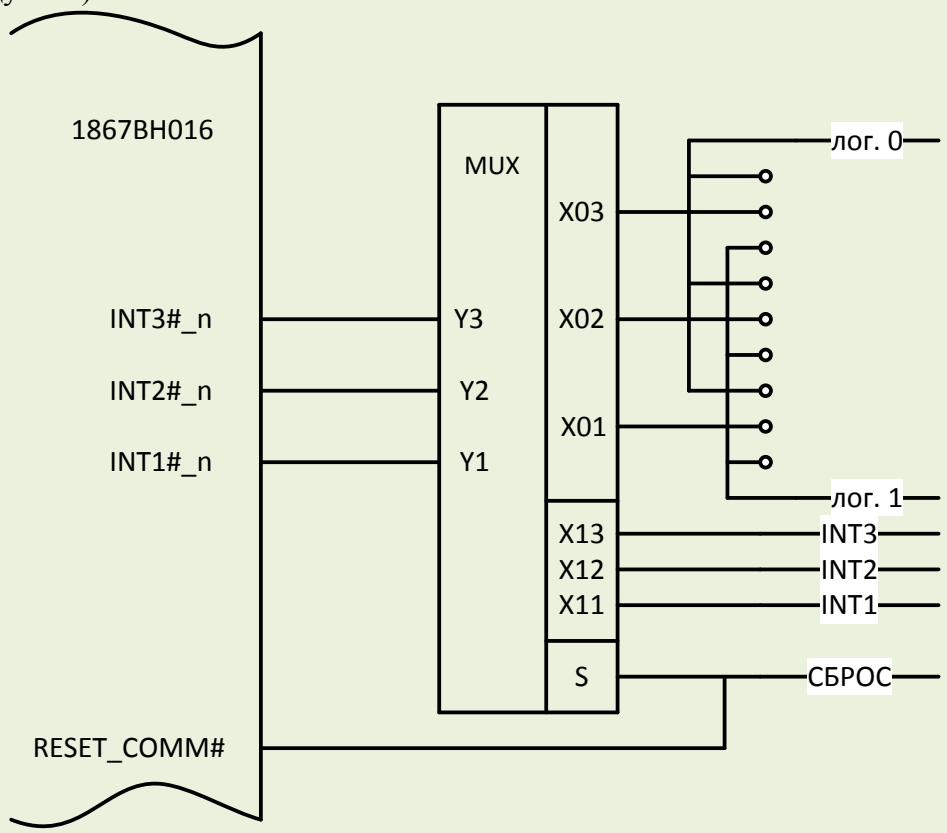
Адрес	Значение	Описание
1000h	0010h	Ширина данных – 16 бит
1001h	0000h	
1002h	1058h	Программирование EBCR – регистра управления Expansion Bus (в данном примере: поле SWW = 11; поле WTCNT = 2)
1003h	0000h	
1004h	1058h	Программирование PBCR – регистра управления Expansion Bus (в данном примере: поле SWW = 11; поле WTCNT = 2)
1005h	0000h	
1006h	01FFh	Размер программного блока (1FF – 512 32-битных слов в блоке данных)
1007h	0000h	
1008h	9C00h	Стартовый адрес назначения загружаемых данных – 80_9C00h
1009h	0080h	

Таблица 4.13 – Пример структуры данных загрузки с разрядностью 8 бит

Адрес	Значение	Описание
1000h	08h	Ширина данных – 8 бит
1001h	00h	
1002h	00h	
1003h	00h	
1004h	58h	Программирование EBCR – регистра управления Expansion Bus (в данном примере: поле SWW = 11; поле WTCNT = 2)
1005h	10h	
1006h	00h	
1007h	00h	
1008h	58h	Программирование PBCR – регистра управления Expansion Bus (в данном примере: поле SWW = 11; поле WTCNT = 2)
1009h	10h	
100Ah	00h	
100Bh	00h	
100Ch	FFh	Размер программного блока (1FF – 512 32-битных слов в блоке данных)
100Dh	01h	
100Eh	00h	
100Fh	00h	
1010h	00h	Стартовый адрес назначения загружаемых данных – 80_9C00h
1011h	9Ch	
1012h	80h	
1013h	00h	

Примечание – В соответствии с протоколами обмена имеющихся в ИС 1867ВН016 последовательных периферийных интерфейсов, данные, загружаемые через MIL-STD-1553, имеют разрядность 16 бит (см. таблицу 4.12), а через UART – 8 бит (см. таблицу 4.13).

На рисунке 4.5 приведена обобщённая схема подключения входов INT3#_*n* – INT1#_*n* для установки кода первоначальной загрузки. Значения сигналов на этих входах во время системного сброса (при переходе RESET_COMM# из низкого уровня в высокий) защёлкиваются в соответствующих разрядах регистра кода первоначальной загрузки BLCR (см. рисунок 4.5 и таблицу 4.14).



Примечание – MUX – условное обозначение трёхканального мультиплексора 2 в 1, реализуемого пользователем самостоятельно на основе предпочитаемой компонентной базы.

Рисунок 4.5 – Обобщённая схема подключения выводов INT3#_*n* – INT1#_*n* (*n* – номер ядра)

На рисунке 4.6 приведена структура регистра кода первоначальной загрузки BLCR, в таблице 4.14 – описание его разрядов.

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
15	14	13	12	11	10	9	8
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
7	6	5	4	3	2	1	0
RSV	RSV	RSV	RSV	RSV	BLC2	BLC1	BLC0
R-X	R-X	R-X	R-X	R-X	RWC-0	RWC-0	RWC-0

Примечание – Принятые условные обозначения: R – бит только читается, RWC – бит доступен на чтение, запись и сбрасывается при сбросе, X – значение не определено.

Рисунок 4.6 – Структура регистра кода первоначальной загрузки BLCR /адрес **80_4702**h/

Таблица 4.14 – Описание разрядов регистра BLCR

Биты	Обозначение	Описание
31 – 3	RSV	Зарезервировано.
2	BLC2	В этот бит защёлкивается уровень на входе INT3#_*n* во время системного сброса (при переходе RESET_COMM# из низкого уровня в высокий).
1	BLC1	В этот бит защёлкивается уровень на входе INT2#_*n* во время системного сброса (при переходе RESET_COMM# из низкого уровня в высокий).
0	BLC0	В этот бит защёлкивается уровень на входе INT1#_*n* во время системного сброса (при переходе RESET_COMM# из низкого уровня в высокий).

На рисунках 4.7, 4.8 и 4.9 приведены алгоритмы работы загрузчика, для случаев, когда источником загружаемого пользовательского кода является параллельный интерфейс (Primary Bus), последовательные интерфейсы UART и MIL-STD-1553 соответственно.

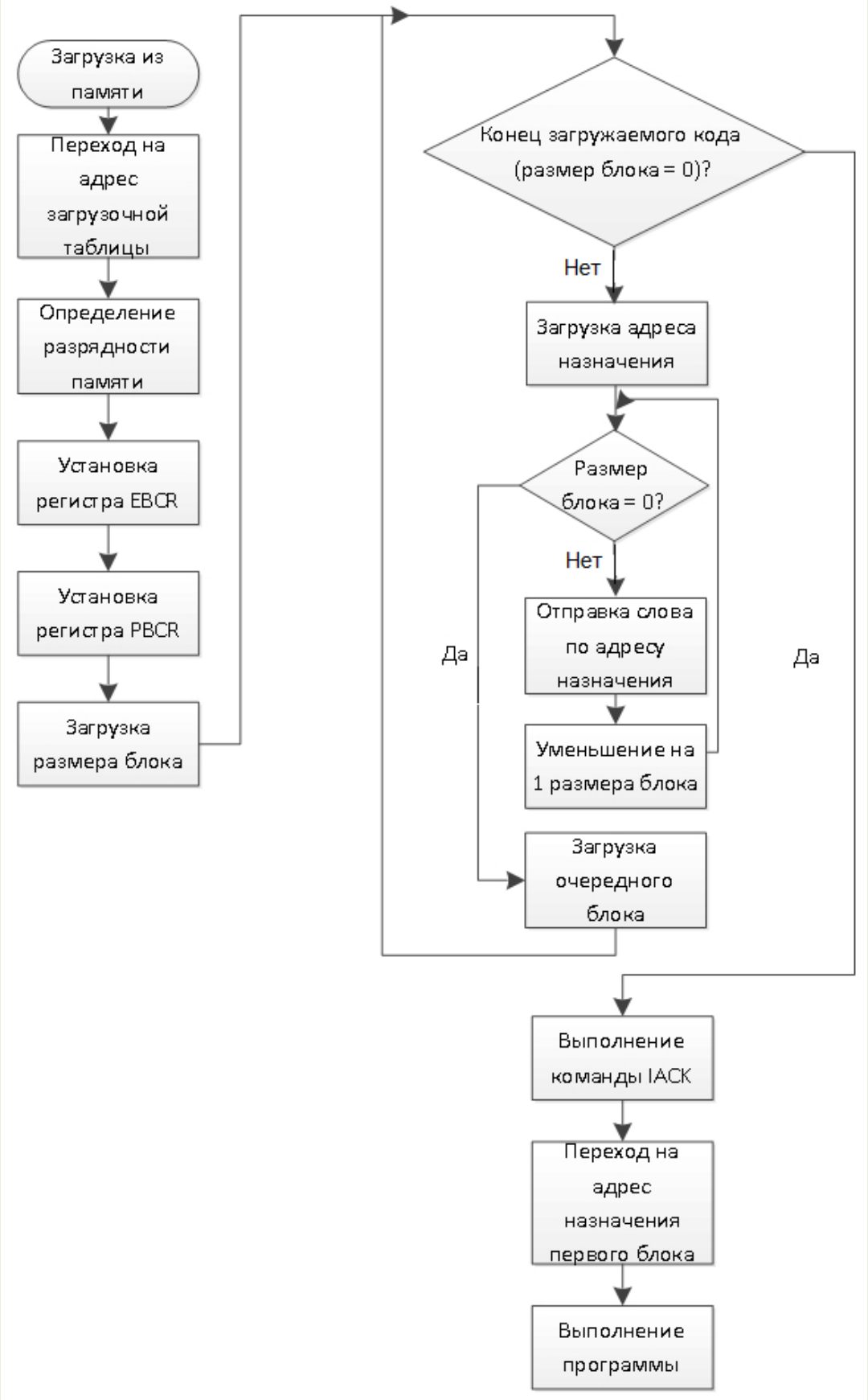


Рисунок 4.7 – Алгоритм работы начального загрузчика через параллельный интерфейс

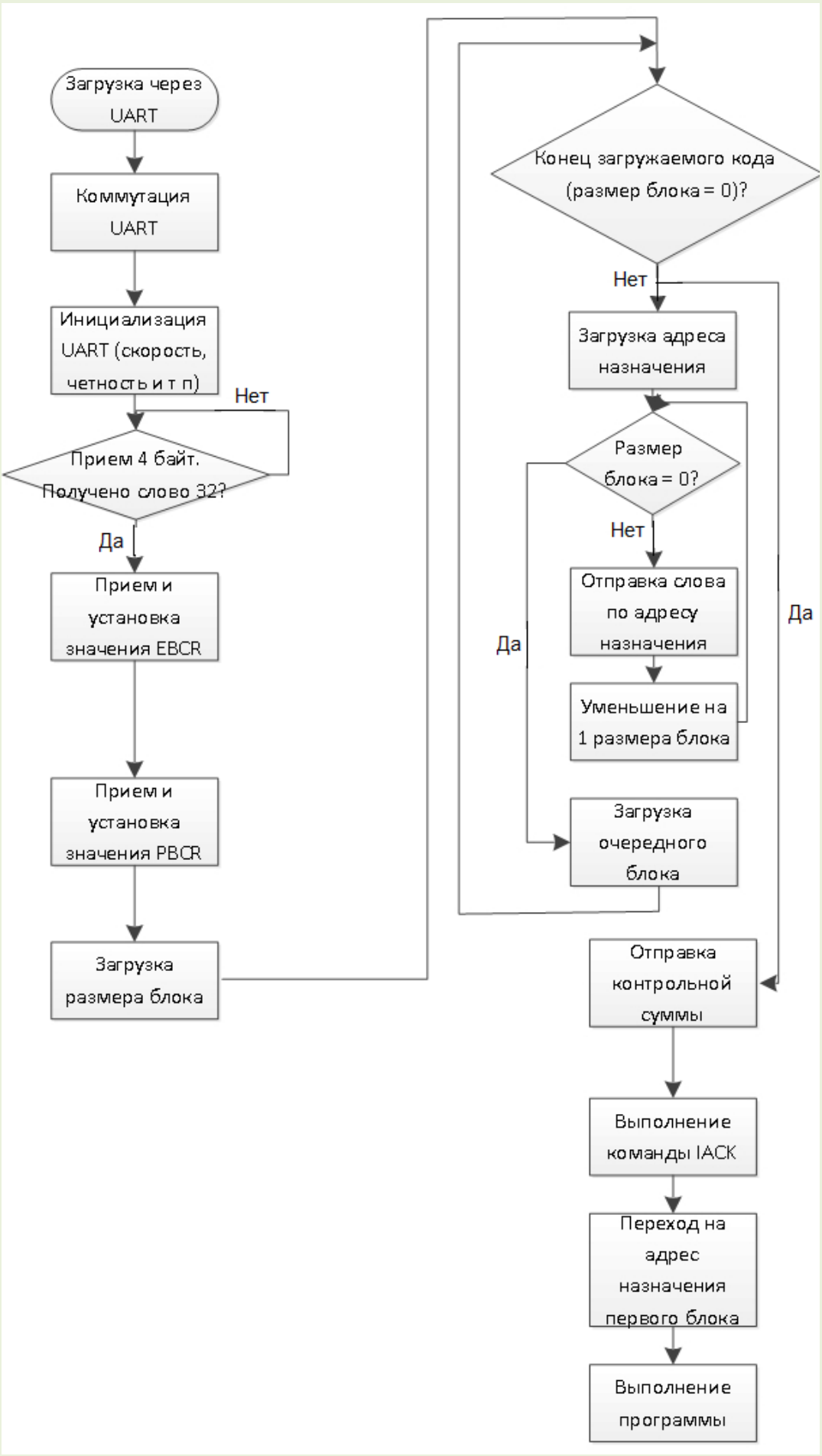


Рисунок 4.8 – Алгоритм работы начального загрузчика через интерфейс UART

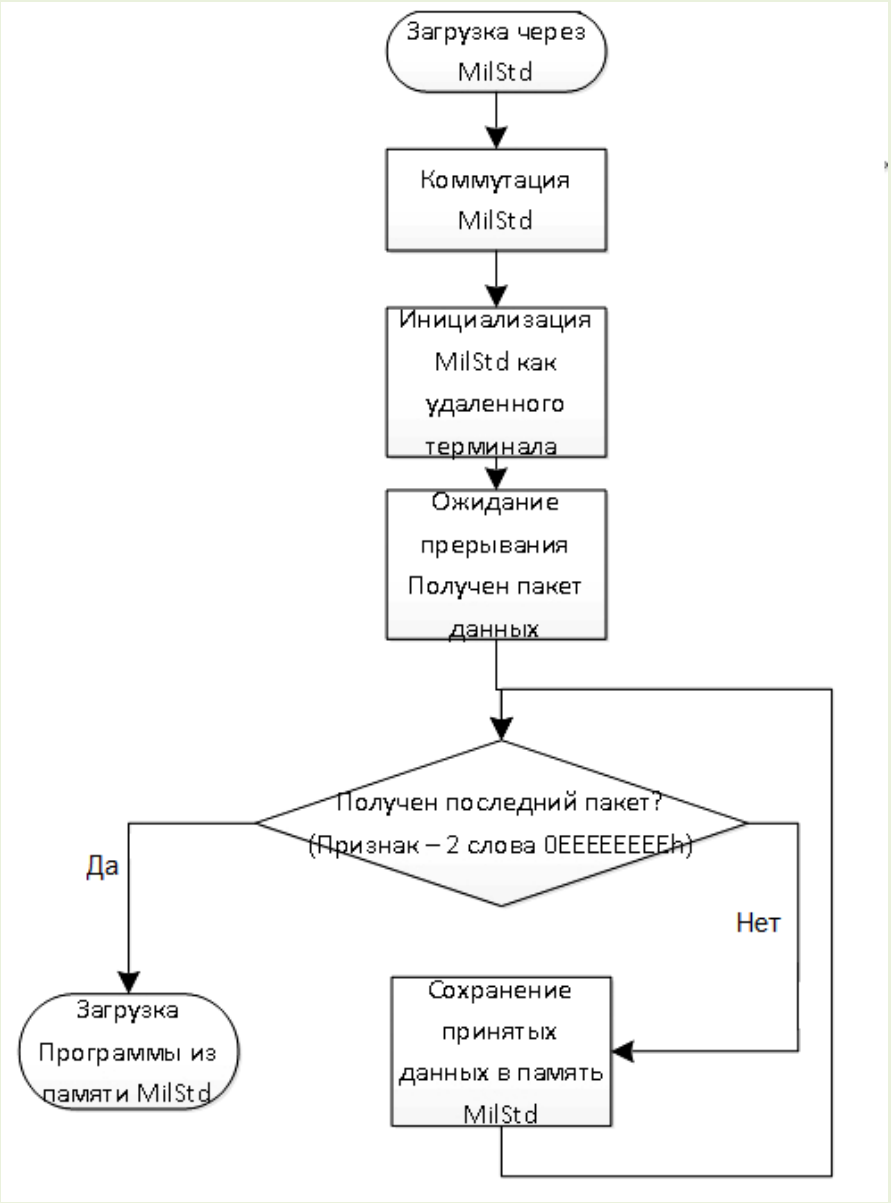


Рисунок 4.9 – Алгоритм работы начального загрузчика через интерфейс MIL-STD-1553

Регистр номера процессора

Регистр номера процессора (Processor Number Register – PNR) доступен только на чтение и служит для программной идентификации процессорного ядра.

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
15	14	13	12	11	10	9	8
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
7	6	5	4	3	2	1	0
RSV	RSV	RSV	RSV	RSV	RSV	PN1	PN0
R-X	R-X	R-X	R-X	R-X	R-X	R	R

Примечание – Принятое условное обозначение: R – бит только читается, X – значение не определено. Допустимые значения поля PN = 01b, 10b.

Рисунок 4.10 – Структура регистра номера процессора PNR /адрес **FF_F800**h/

Таблица 4.15 – Описание битов регистра номера процессора

Бит	Обозначение	Описание
31 – 2	RSV	Зарезервировано.
1 – 0	PN1, PN0	- PN =01b – номер процессора 0; - PN =10b – номер процессора 1

5 Таблица функционального назначения выводов

В таблице 5.1 приведена нумерация, обозначение, описание функционального назначения выводов ИС 1867ВН016.

Таблица 5.1 – Функциональное назначение выводов микросхемы

Номер КП на кристалле(ах)/ координаты вывода корпуса	Обозначение вывода	Описание	Тип вывода
1	2	3	4
Процессор 0			
Первичный интерфейс процессора 0			
10/699/G6	D31_0	31 бит шины данных первичного интерфейса процессора 0	I/O/Z
1/701/G5	D30_0	30 бит шины данных первичного интерфейса процессора 0	I/O/Z
8/740/G4	D29_0	29 бит шины данных первичного интерфейса процессора 0	I/O/Z
7/742/H1	D28_0	28 бит шины данных первичного интерфейса процессора 0	I/O/Z
16/702/H2	D27_0	27 бит шины данных первичного интерфейса процессора 0	I/O/Z
14/706/H3	D26_0	26 бит шины данных первичного интерфейса процессора 0	I/O/Z
12/735/H4	D25_0	25 бит шины данных первичного интерфейса процессора 0	I/O/Z
19/738/H6	D24_0	24 бит шины данных первичного интерфейса процессора 0	I/O/Z
17/649/H7	D23_0	23 бит шины данных первичного интерфейса процессора 0	I/O/Z
22/647/J8	D22_0	22 бит шины данных первичного интерфейса процессора 0	I/O/Z
18/608/J6	D21_0	21 бит шины данных первичного интерфейса процессора 0	I/O/Z
20/606/J5	D20_0	20 бит шины данных первичного интерфейса процессора 0	I/O/Z
24/646/J4	D19_0	19 бит шины данных первичного интерфейса процессора 0	I/O/Z
29/642/K3	D18_0	18 бит шины данных первичного интерфейса процессора 0	I/O/Z
23/613/K4	D17_0	17 бит шины данных первичного интерфейса процессора 0	I/O/Z
28/610/K5	D16_0	16 бит шины данных первичного интерфейса процессора 0	I/O/Z
25/707/K6	D15_0	15 бит шины данных первичного интерфейса процессора 0	I/O/Z
31/709/K8	D14_0	14 бит шины данных первичного интерфейса процессора 0	I/O/Z
35/732/L8	D13_0	13 бит шины данных первичного интерфейса процессора 0	I/O/Z
30/734/L7	D12_0	12 бит шины данных первичного интерфейса процессора 0	I/O/Z
34/641/L5	D11_0	11 бит шины данных первичного интерфейса процессора 0	I/O/Z
38/639/L4	D10_0	10 бит шины данных первичного интерфейса процессора 0	I/O/Z
33/616/L1	D09_0	9 бит шины данных первичного интерфейса процессора 0	I/O/Z
40/614/M3	D08_0	8 бит шины данных первичного интерфейса процессора 0	I/O/Z
42/713/M4	D07_0	7 бит шины данных первичного интерфейса процессора 0	I/O/Z
41/714/M6	D06_0	6 бит шины данных первичного интерфейса процессора 0	I/O/Z
45/725/N5	D05_0	5 бит шины данных первичного интерфейса процессора 0	I/O/Z
44/726/N4	D04_0	4 бит шины данных первичного интерфейса процессора 0	I/O/Z
46/635/N3	D03_0	3 бит шины данных первичного интерфейса процессора 0	I/O/Z
48/634/N1	D02_0	2 бит шины данных первичного интерфейса процессора 0	I/O/Z
51/621/P3	D01_0	1 бит шины данных первичного интерфейса процессора 0	I/O/Z
50/620/P4	D00_0	0 бит шины данных первичного интерфейса процессора 0	I/O/Z
572/D10	A23_0	23 бит адреса первичного интерфейса процессора 0	O/Z
571/D9	A22_0	22 бит адреса первичного интерфейса процессора 0	O/Z
576/D8	A21_0	21 бит адреса первичного интерфейса процессора 0	O/Z
578/D7	A20_0	20 бит адреса первичного интерфейса процессора 0	O/Z
574/D6	A19_0	19 бит адреса первичного интерфейса процессора 0	O/Z

Продолжение таблицы 5.1

1	2	3	4
575/673/D5	A18_0	18 бит адреса первичного интерфейса процессора 0	O/Z
579/669/E5	A17_0	17 бит адреса первичного интерфейса процессора 0	O/Z
583/667/E6	A16_0	16 бит адреса первичного интерфейса процессора 0	O/Z
585/666/E7	A15_0	15 бит адреса первичного интерфейса процессора 0	O/Z
581/748/E9	A14_0	14 бит адреса первичного интерфейса процессора 0	O/Z
588/758/E10	A13_0	13 бит адреса первичного интерфейса процессора 0	O/Z
586/759/F11	A12_0	12 бит адреса первичного интерфейса процессора 0	O/Z
584/760/F10	A11_0	11 бит адреса первичного интерфейса процессора 0	O/Z
589/762/F9	A10_0	10 бит адреса первичного интерфейса процессора 0	O/Z
587/767/F8	A09_0	9 бит адреса первичного интерфейса процессора 0	O/Z
595/768/F7	A08_0	8 бит адреса первичного интерфейса процессора 0	O/Z
598/769/F6	A07_0	7 бит адреса первичного интерфейса процессора 0	O/Z
593/774/F5	A06_0	6 бит адреса первичного интерфейса процессора 0	O/Z
596/775/G7	A05_0	5 бит адреса первичного интерфейса процессора 0	O/Z
599/776/G8	A04_0	4 бит адреса первичного интерфейса процессора 0	O/Z
597/786/G9	A03_0	3 бит адреса первичного интерфейса процессора 0	O/Z
2/683/G11	A02_0	2 бит адреса первичного интерфейса процессора 0	O/Z
6/665/H10	A01_0	1 бит адреса первичного интерфейса процессора 0	O/Z
5/655/H9	A00_0	0 бит адреса первичного интерфейса процессора 0	O/Z
563/G12	R/W#_0	Выход «ЧТЕНИЕ/ЗАПИСЬ» первичного интерфейса процессора 0	O/Z
566/F12	STRB#_0	Выход «СТРОБ» первичного интерфейса процессора 0	O/Z
567/D12	RDY#_0	Вход «ГОТОВ» процессора 0	I
565/E11	HOLD#_0	Вход «ДЕРЖАТЬ» первичного интерфейса процессора 0	I
573/D11	HOLDA#_0	Выход HOLDA#_0 подтверждения захвата внешним устройством первичного интерфейса процессора 0	O/Z
Сигналы прерывания и управления процессора 0			
556/B10	INT3#_0	Вход внешнего прерывания 3 процессора 0. Pull-up	I
555/C9	INT2#_0	Вход внешнего прерывания 2 процессора 0. Pull-up	I
557/C8	INT1#_0	Вход внешнего прерывания 1 процессора 0. Pull-up	I
560/C7	INT0#_0	Вход внешнего прерывания 0 процессора 0. Pull-up	I
562/B7	IACK#_0	Выход подтверждения прерывания	O/Z
561/A7	MCBL/MP#_0	Вход выбора режима микрокомпьютер/микропроцессор процессора 0	I
Последовательный порт 0 процессора 0			
538/H13	CLKX0_0	Вывод тактирования передатчика последовательного порта 0 процессора 0. Вход/выход общего назначения	I/O/Z
541/G13	DX0_0	Выход данных передатчика последовательного порта 0 процессора 0. Вход/выход общего назначения	I/O/Z
540/F14	FSX0_0	Вывод синхронизирующего сигнала начала фрейма передачи последовательного порта 0 процессора 0. Вход/выход общего назначения	I/O/Z
542/F13	CLKR0_0	Вывод тактирования приёмника последовательного порта 0 процессора 0. Вход/выход общего назначения	I/O/Z
544/E13	DR0_0	Вход данных приёмника последовательного порта 0 процессора 0. Вход/выход общего назначения	I/O/Z
543/D13	FSR0_0	Вывод синхронизирующего сигнал начала фрейма приёма последовательного порта 0 процессора 0. Вход/выход общего назначения	I/O/Z

Продолжение таблицы 5.1

1	2	3	4
Последовательный порт 1 процессора 0			
545/C13	CLKX1_0	Вывод тактирования передатчика последовательного порта 1 процессора 0. Вход/выход общего назначения	I/O/Z
547/A13	DX1_0	Выход данных передатчика последовательного порта 1 процессора 0. Вход/выход общего назначения	I/O/Z
549/C12	FSX1_0	Вывод синхронизирующего сигнала начала фрейма передачи последовательного порта 1 процессора 0. Вход/выход общего назначения	I/O/Z
551/B12	CLKR1_0	Вывод тактирования приёмника последовательного порта 1 процессора 0. Вход/выход общего назначения	I/O/Z
552/A12	DR1_0	Вход данных приёмника последовательного порта 1 процессора 0. Вход/выход общего назначения	I/O/Z
554/A11	FSR1_0	Вывод синхронизирующего сигнала начала фрейма приёма последовательного порта 1 процессора 0. Вход/выход общего назначения	I/O/Z
Сигналы таймера 0 и таймера 1 процессора 0			
530/G16	TCLK0_0	Вход тактового сигнала таймера 0 процессора 0. Выход счётчика импульсов таймера 0 процессора 0. Вход/выход общего назначения	I/O/Z
531/G15	TCLK1_0	Вход тактового сигнала таймера 1 процессора 0. Выход счётчика импульсов таймера 1 процессора 0. Вход/выход общего назначения	I/O/Z
Внешние флаги процессора 0			
533/F16	XF0_0	Выход внешнего флага 0 процессора 0. Вывод поддержки инструкций взаимной блокировки (LDFI, LDII, SIGI, STFI, STII) в мультипроцессорных конфигурациях. Вход/выход общего назначения	I/O/Z
534/F15	XF1_0	Выход внешнего флага 1 процессора 0. Вывод поддержки инструкций взаимной блокировки (LDFI, LDII, SIGI, STFI, STII) в мультипроцессорных конфигурациях. Вход/выход общего назначения	I/O/Z
Выходные тактовые сигналы процессора 0			
536/E15	H1_0	Выход тактового сигнала процессора 0	O/Z
537/E14	H3_0	Выход тактового сигнала процессора 0, инверсный по отношению к H1_0	O/Z
Сигналы эмулятора процессора 0			
523/B16	EMU3_0	Выходной сигнал эмулятора процессора 0	O/Z
524/C15	EMU2_0	Входной сигнал 2 эмулятора процессора 0. Pull-up	I
525/D15	EMU1_0	Входной сигнал 1 эмулятора процессора 0. Pull-up	I
526/D14	EMU0_0	Входной сигнал 0 эмулятора процессора 0. Pull-up	I
Сигнал третьего состояния выходов процессора 0			
527/C14	EMU4#_0	Входной сигнал перевода выводов в высокоимпедансное состояние. Pull-up	I
Процессор 1			
Первичный интерфейс процессора 1			
130/885/AB4	D31_1	31 бит шины данных первичного интерфейса процессора 1	I/O/Z
122/887/AA3	D30_1	30 бит шины данных первичного интерфейса процессора 1	I/O/Z
123/926/AA4	D29_1	29 бит шины данных первичного интерфейса процессора 1	I/O/Z
118/928/AA5	D28_1	28 бит шины данных первичного интерфейса процессора 1	I/O/Z

Продолжение таблицы 5.1

1	2	3	4
119/888/AA6	D27_1	27 бит шины данных первичного интерфейса процессора 1	I/O/Z
120/892/AA7	D26_1	26 бит шины данных первичного интерфейса процессора 1	I/O/Z
115/921/Y8	D25_1	25 бит шины данных первичного интерфейса процессора 1	I/O/Z
116/924/Y7	D24_1	24 бит шины данных первичного интерфейса процессора 1	I/O/Z
114/835/Y6	D23_1	23 бит шины данных первичного интерфейса процессора 1	I/O/Z
111/833/Y4	D22_1	22 бит шины данных первичного интерфейса процессора 1	I/O/Z
110/794/Y3	D21_1	21 бит шины данных первичного интерфейса процессора 1	I/O/Z
108/792/Y1	D20_1	20 бит шины данных первичного интерфейса процессора 1	I/O/Z
107/832/W2	D19_1	19 бит шины данных первичного интерфейса процессора 1	I/O/Z
105/828/W3	D18_1	18 бит шины данных первичного интерфейса процессора 1	I/O/Z
104/799/W4	D17_1	17 бит шины данных первичного интерфейса процессора 1	I/O/Z
102/796/W5	D16_1	16 бит шины данных первичного интерфейса процессора 1	I/O/Z
103/893/V8	D15_1	15 бит шины данных первичного интерфейса процессора 1	I/O/Z
101/895/V6	D14_1	14 бит шины данных первичного интерфейса процессора 1	I/O/Z
100/918/V5	D13_1	13 бит шины данных первичного интерфейса процессора 1	I/O/Z
96/920/V4	D12_1	12 бит шины данных первичного интерфейса процессора 1	I/O/Z
95/827/V3	D11_1	11 бит шины данных первичного интерфейса процессора 1	I/O/Z
93/825/U2	D10_1	10 бит шины данных первичного интерфейса процессора 1	I/O/Z
92/802/U4	D09_1	9 бит шины данных первичного интерфейса процессора 1	I/O/Z
90/800/U5	D08_1	8 бит шины данных первичного интерфейса процессора 1	I/O/Z
91/899/U6	D07_1	7 бит шины данных первичного интерфейса процессора 1	I/O/Z
89/900/U7	D06_1	6 бит шины данных первичного интерфейса процессора 1	I/O/Z
88/911/U8	D05_1	5 бит шины данных первичного интерфейса процессора 1	I/O/Z
84/912/T4	D04_1	4 бит шины данных первичного интерфейса процессора 1	I/O/Z
83/821/T3	D03_1	3 бит шины данных первичного интерфейса процессора 1	I/O/Z
82/820/R3	D02_1	2 бит шины данных первичного интерфейса процессора 1	I/O/Z
81/807/R4	D01_1	1 бит шины данных первичного интерфейса процессора 1	I/O/Z
80/806/R5	D00_1	0 бит шины данных первичного интерфейса процессора 1	I/O/Z
163/AD11	A23_1	23 бит адреса первичного интерфейса процессора 1	O/Z
170/AD10	A22_1	22 бит адреса первичного интерфейса процессора 1	O/Z
165/AD9	A21_1	21 бит адреса первичного интерфейса процессора 1	O/Z
167/AD8	A20_1	20 бит адреса первичного интерфейса процессора 1	O/Z
155/AD7	A19_1	19 бит адреса первичного интерфейса процессора 1	O/Z
159/859/AD6	A18_1	18 бит адреса первичного интерфейса процессора 1	O/Z
161/855/AD5	A17_1	17 бит адреса первичного интерфейса процессора 1	O/Z
152/853/AC5	A16_1	16 бит адреса первичного интерфейса процессора 1	O/Z
156/852/AC6	A15_1	15 бит адреса первичного интерфейса процессора 1	O/Z
157/934/AC7	A14_1	14 бит адреса первичного интерфейса процессора 1	O/Z
144/944/AC9	A13_1	13 бит адреса первичного интерфейса процессора 1	O/Z
150/945/AC10	A12_1	12 бит адреса первичного интерфейса процессора 1	O/Z
146/946/AC11	A11_1	11 бит адреса первичного интерфейса процессора 1	O/Z
140/948/AB10	A10_1	10 бит адреса первичного интерфейса процессора 1	O/Z
143/953/AB8	A09_1	9 бит адреса первичного интерфейса процессора 1	O/Z
145/954/AB7	A08_1	8 бит адреса первичного интерфейса процессора 1	O/Z
134/955/AB6	A07_1	7 бит адреса первичного интерфейса процессора 1	O/Z
136/960/AB5	A06_1	6 бит адреса первичного интерфейса процессора 1	O/Z
141/961/AA9	A05_1	5 бит адреса первичного интерфейса процессора 1	O/Z
131/962/AA11	A04_1	4 бит адреса первичного интерфейса процессора 1	O/Z

Продолжение таблицы 5.1

1	2	3	4
133/972/Y12	A03_1	3 бит адреса первичного интерфейса процессора 1	O/Z
128/869/Y11	A02_1	2 бит адреса первичного интерфейса процессора 1	O/Z
132/851/Y10	A01_1	1 бит адреса первичного интерфейса процессора 1	O/Z
124/841/Y9	A00_1	0 бит адреса первичного интерфейса процессора 1	O/Z
168/Y13	R/W#_1	Выход «ЧТЕНИЕ/ЗАПИСЬ» первичного интерфейса процессора 1	O/Z
173/AA12	STRB#_1	Выход «СТРОБ» первичного интерфейса процессора 1	O/Z
166/AB13	RDY#_1	Вход «ГОТОВ» процессора 1	I
171/AC13	HOLD#_1	Вход «ДЕРЖАТЬ» первичного интерфейса процессора 1	I
175/AD12	HOLDA#_1	Выход HOLDA#_1 подтверждения захвата внешним устройством первичного интерфейса процессора 1	O/Z
Сигналы прерывания и управления процессора 1			
180/AE15	INT3#_1	Вход внешнего прерывания 3 процессора 1. Pull-up	I
176/AE14	INT2#_1	Вход внешнего прерывания 2 процессора 1. Pull-up	I
179/AD14	INT1#_1	Вход внешнего прерывания 1 процессора 1. Pull-up	I
183/AC14	INT0#_1	Вход внешнего прерывания 0 процессора 1. Pull-up	I
182/AB14	IACK#_1	Выход подтверждения прерывания	O/Z
185/Y14	MCBL/MP#_1	Вход выбора режима микрокомпьютер/микропроцессор процессора 1	I
Последовательный порт 0 процессора 1			
184/AG17	CLKX0_1	Вывод тактирования передатчика последовательного порта 0 процессора 1. Вход/выход общего назначения	I/O/Z
190/AG16	DX0_1	Выход данных передатчика последовательного порта 0 процессора 1. Вход/выход общего назначения	I/O/Z
193/AF17	FSX0_1	Вывод синхронизирующего сигнала начала фрейма передачи последовательного порта 0 процессора 1. Вход/выход общего назначения	I/O/Z
194/AE16	CLKR0_1	Вывод тактирования приёмника последовательного порта 0 процессора 1. Вход/выход общего назначения	I/O/Z
198/AD17	DR0_1	Вход данных приёмника последовательного порта 0 процессора 1. Вход/выход общего назначения	I/O/Z
202/AD16	FSR0_1	Вывод синхронизирующего сигнала начала фрейма приёма последовательного порта 0 процессора 1. Вход/выход общего назначения	I/O/Z
Последовательный порт 1 процессора 1			
189/AD15	CLKX1_1	Вывод тактового сигнал передатчика последовательного порта 1 процессора 1. Вход/выход общего назначения	I/O/Z
191/AB15	DX1_1	Выход данных передатчика последовательного порта 1 процессора 1. Вход/выход общего назначения	I/O/Z
192/AB16	FSX1_1	Вывод синхронизирующего сигнала начала фрейма передачи последовательного порта 1 процессора 1. Вход/выход общего назначения	I/O/Z
197/AA16	CLKR1_1	Вывод тактового сигнала приёмника последовательного порта 1 процессора 1. Вход/выход общего назначения	I/O/Z
200/AA15	DR1_1	Вход данных приёмника последовательного порта 1 процессора 1. Вход/выход общего назначения	I/O/Z
201/Y15	FSR1_1	Вывод синхронизирующего сигнала начала фрейма приёма последовательного порта 1 процессора 1. Вход/выход общего назначения	I/O/Z

Продолжение таблицы 5.1

1	2	3	4
Сигналы таймера 0 и таймера 1 процессора 1			
203/AF18	TCLK0_1	Вход тактового сигнала таймера 0 процессора 1. Выход счётчика импульсов таймера 0 процессора 1. Вход/выход общего назначения	I/O/Z
205/AE18	TCLK1_1	Вход тактового сигнала таймера 1 процессора 1. Выход счётчика импульсов таймера 1 процессора 1. Вход/выход общего назначения	I/O/Z
Внешние флаги процессора 1			
208/AD18	XF0_1	Вывод внешнего флага 0 процессора 1. Вывод поддержки инструкций взаимной блокировки (LDFI, LDII, SIGI, STFI, STII) в мультипроцессорных конфигурациях. Вход/выход общего назначения	I/O/Z
209/AC18	XF1_1	Вывод внешнего флага 1 процессора 1. Вывод поддержки инструкций взаимной блокировки (LDFI, LDII, SIGI, STFI, STII) в мультипроцессорных конфигурациях. Вход/выход общего назначения	I/O/Z
Выходные тактовые сигналы процессора 1			
211/AC17	H1_1	Выход тактового сигнала процессора 1	O/Z
210/AB17	H3_1	Выход тактового сигнала процессора 1, инверсный по отношению к H1_1	O/Z
Сигналы эмулятора процессора 1			
212/Y17	EMU3_1	Выходной сигнал эмулятора процессора 1	O/Z
214/AB18	EMU2_1	Входной сигнал 2 эмулятора процессора 1. Pull-up	I
213/AB19	EMU1_1	Входной сигнал 1 эмулятора процессора 1. Pull-up	I
218/AA19	EMU0_1	Входной сигнал 0 эмулятора процессора 1. Pull-up	I
Сигнал третьего состояния выходов процессора 1			
219/Y19	EMU4#_1	Входной сигнал перевода выводов в высокоимпедансное состояние. Pull-up	I
Общие сигналы для процессора 0, процессора 1 и внутрикристальной периферии			
72/M7	RESET_COMM#	Вход сброса	I
73/M8	NC	Неиспользуемый вывод	–
74/N7	X2/CLKIN	Входной тактовый сигнал от осциллятора или генератора	I
76/R7	CLKMD	Вход выбора коэффициента умножения тактового сигнала	I
78/1019/R8	GND33_osc_CPU	Общий вывод блока осциллятора CPU	–
79/1020/T7	VDD33_osc_CPU	Вывод питания 3,3 В блока осциллятора CPU	–
Разделяемые периферийные устройства			
Входы/выходы общего назначения GPIO_0			
302/AF19	GPIO15_0	Вход/выход общего назначения, разряд 15	I/O
307/AE19	GPIO14_0	Вход/выход общего назначения, разряд 14	I/O
310/AE20	GPIO13_0	Вход/выход общего назначения, разряд 13	I/O
315/AF20	GPIO12_0	Вход/выход общего назначения, разряд 12	I/O
306/AG20	GPIO11_0	Вход/выход общего назначения, разряд 11	I/O
313/AG22	GPIO10_0	Вход/выход общего назначения, разряд 10	I/O
309/AF22	GPIO9_0	Вход/выход общего назначения, разряд 9	I/O
318/AD23	GPIO8_0	Вход/выход общего назначения, разряд 8	I/O
319/AE23	GPIO7_0	Вход/выход общего назначения, разряд 7	I/O
324/AF23	GPIO6_0	Вход/выход общего назначения, разряд 6	I/O
317/AF24	GPIO5_0	Вход/выход общего назначения, разряд 5	I/O
323/AE24	GPIO4_0	Вход/выход общего назначения, разряд 4	I/O
328/AE25	GPIO3_0	Вход/выход общего назначения, разряд 3	I/O

Продолжение таблицы 5.1

1	2	3	4
322/AF25	GPIO2_0	Вход/выход общего назначения, разряд 2	I/O
326/AG25	GPIO1_0	Вход/выход общего назначения, разряд 1	I/O
329/AG26	GPIO0_0	Вход/выход общего назначения, разряд 0	I/O
Входы/выходы общего назначения GPIO_1			
354/T25	GPIO15_1	Вход/выход общего назначения, разряд 15	I/O
352/V26	GPIO14_1	Вход/выход общего назначения, разряд 14	I/O
350/Y26	GPIO13_1	Вход/выход общего назначения, разряд 13	I/O
348/Y25	GPIO12_1	Вход/выход общего назначения, разряд 12	I/O
346/Y24	GPIO11_1	Вход/выход общего назначения, разряд 11	I/O
347/AA24	GPIO10_1	Вход/выход общего назначения, разряд 10	I/O
345/AB27	GPIO9_1	Вход/выход общего назначения, разряд 9	I/O
343/AC27	GPIO8_1	Вход/выход общего назначения, разряд 8	I/O
344/AC25	GPIO7_1	Вход/выход общего назначения, разряд 7	I/O
340/AC24	GPIO6_1	Вход/выход общего назначения, разряд 6	I/O
335/AD24	GPIO5_1	Вход/выход общего назначения, разряд 5	I/O
337/AD26	GPIO4_1	Вход/выход общего назначения, разряд 4	I/O
336/AD27	GPIO3_1	Вход/выход общего назначения, разряд 3	I/O
332/AE26	GPIO2_1	Вход/выход общего назначения, разряд 2	I/O
333/AF26	GPIO1_1	Вход/выход общего назначения, разряд 1	I/O
334/AF27	GPIO0_1	Вход/выход общего назначения, разряд 0	I/O
MIL-STD-1553_0			
356/T26	RXIN_A#_0	Входной сигнал канала А, инверсный	I
355/R27	RXIN_A_0	Входной сигнал канала А, неинверсный	I
357/R25	TXOUT_A_0	Выходной сигнал канала А, неинверсный	O
359/P25	TXOUT_A#_0	Выходной сигнал канала А, инверсный	O
360/N25	TXINA_0	Выход сигнала блокировки передатчика канала А	O
365/N27	TXOUT_B_0	Выходной сигнал канала В, неинверсный	O
364/M27	TXOUT_B#_0	Выходной сигнал канала В, инверсный	O
366/L26	TXINB_0	Выходной сигнал блокировки передатчика канала В	O
367/K26	RXIN_B_0	Входной сигнал канала В, неинверсный	I
368/K25	RXIN_B#_0	Входной сигнал канала В, инверсный	I
369/J25	A4_RT0	Вход 4 разряда адреса удалённого устройства MIL-STD-1553_0	I
370/H25	A3_RT0	Вход 3 разряда адреса удалённого устройства MIL-STD-1553_0	I
371/H26	A2_RT0	Вход 2 разряда адреса удалённого устройства MIL-STD-1553_0	I
374/H27	A1_RT0	Вход 1 разряда адреса удалённого устройства MIL-STD-1553_0	I
375/G27	A0_RT0	Вход 0 разряда адреса удалённого устройства MIL-STD-1553_0	I
376/F27	PA_RT0	Бит паритета адреса удалённого устройства MIL-STD-1553_0	I
MIL-STD-1553_1			
377/E24	RXIN_A#_1	Входной сигнал канала А, инверсный	I
379/D24	RXIN_A_1	Входной сигнал канала А, неинверсный	I
383/D23	TXOUT_A_1	Выходной сигнал канала А, неинверсный	O
384/E23	TXOUT_A#_1	Выходной сигнал канала А, инверсный	O
385/E22	TXINA_1	Выходной сигнал блокировки передатчика канала А	O
386/E21	TXOUT_B_1	Выходной сигнал канала В, неинверсный	O
387/F20	TXOUT_B#_1	Выходной сигнал канала В, инверсный	O
388/G19	TXINB_1	Выходной сигнал блокировки передатчика канала В	O
392/F19	RXIN_B_1	Входной сигнал канала В, неинверсный	I
393/E19	RXIN_B#_1	Входной сигнал канала В, инверсный	I
383/D23	TXOUT_A_1	Выходной сигнал канала А, неинверсный	O

Продолжение таблицы 5.1

1	2	3	4
MIL-STD-1553_2			
398/C24	RXIN_A#_2	Входной сигнал канала А, инверсный	I
403/B24	RXIN_A_2	Входной сигнал канала А, неинверсный	I
405/B23	TXOUT_A_2	Выходной сигнал канала А, неинверсный	O
404/A23	TXOUT_A#_2	Выходной сигнал канала А, инверсный	O
408/A21	TXINA_2	Выходной сигнал блокировки передатчика канала А	O
407/C21	TXOUT_B_2	Выходной сигнал канала В, неинверсный	O
409/D21	TXOUT_B#_2	Выходной сигнал канала В, инверсный	O
411/A20	TXINB_2	Выходной сигнал блокировки передатчика канала В	O
410/D20	RXIN_B_2	Входной сигнал канала В, неинверсный	I
412/D19	RXIN_B#_2	Входной сигнал канала В, инверсный	I
MIL-STD-1553_3			
419/E25	RXIN_A#_3	Входной сигнал канала А, инверсный	I
417/E26	RXIN_A_3	Входной сигнал канала А, неинверсный	I
418/D26	TXOUT_A_3	Выходной сигнал канала А, неинверсный	O
422/D25	TXOUT_A#_3	Выходной сигнал канала А, инверсный	O
421/C25	TXINA_3	Выходной сигнал блокировки передатчика канала А	O
425/C26	TXOUT_B_3	Выходной сигнал канала В, неинверсный	O
420/C27	TXOUT_B#_3	Выходной сигнал канала В, инверсный	O
429/B27	TXINB_3	Выходной сигнал блокировки передатчика канала В	O
424/B26	RXIN_B_3	Входной сигнал канала В, неинверсный	I
428/A26	RXIN_B#_3	Входной сигнал канала В, инверсный	I
Тактовый сигнал для MIL-STD-1553_(0, 1, 2, 3)			
395/F24	MilStd_CLK/X2	Вход тактового сигнала MIL-STD-1553 с частотой 24 МГц	I
396/F23	NC	Неиспользуемый вывод	–
394/1205/F22	GND33_osc_Mstd	Общий вывод осциллятора MIL-STD-1553	–
397/1206/F21	VDD33_osc_Mstd	Вывод питания 3,3 В осциллятора MIL-STD-1553	–
UART_0			
498/C16	Rx_0	Вход данных UART_0	I
496/D16	Tx_0	Выход данных UART_0	O
495/D17	RTS_0/EnTr_0	Выход разрешения передачи UART_0	O
493/E17	CTS_0	Вход готовности внешнего устройства к приёму	I
UART_1			
483/F17	Rx_1	Вход данных UART_1	I
484/G17	Tx_1	Выход данных UART_1	O
485/F18	RTS_1/EnTr_1	Выход разрешения передачи UART_1	O
481/H19	CTS_1	Вход готовности внешнего устройства к приёму	I
Тактовый сигнал UART_(0, 1)			
491/E18	UART_CLK/X2	Вход тактового сигнала UART_(0, 1)	I
489/D18	NC	Неиспользуемый вывод	–
487/1269/C18	GND33_osc_UART	Общий вывод осциллятора UART	–
492/1270/B18	VDD33_osc_UART	Вывод питания 3,3 В осциллятора UART	–

Продолжение таблицы 5.1

1	2	3	4
USB 2.0 Device			
426/T24	SuspendM	Выход установки трансивера в режим ожидания	O
435/R22	Xcvr_Select_	Выход режима трансивера Full speed/High speed	O
432/R23	Term_Select	Выход окончания приёма Full speed/High speed	O
433/R20	LineState_1	Входной сигнал 1 состояния линии	I
440/L24	LineState_0	Входной сигнал 0 состояния линии	I
436/L23	Op_Mode_1	Выходной сигнал 1 выбора режима работы	O
438/L22	Op_Mode_0	Выходной сигнал 0 выбора режима работы	O
459/L21	DataOut7	Выход данных, бит 7	O
458/K20	DataOut6	Выход данных, бит 6	O
460/K22	DataOut5	Выход данных, бит 5	O
463/K23	DataOut4	Выход данных, бит 4	O
457/K24	DataOut3	Выход данных, бит 3	O
461/J24	DataOut2	Выход данных, бит 2	O
469/J23	DataOut1	Выход данных, бит 1	O
462/J21	DataOut0	Выход данных, бит 0	O
444/R24	TXValid	Выход «ПЕРЕДАВАЕМЫЕ ДАННЫЕ ПРАВИЛЬНЫЕ»	O
441/P23	TXReady	Вход «ГОТОВНОСТЬ ПЕРЕДАЧИ»	I
467/J20	DataIn7	Вход данных, бит 7	I
472/H20	DataIn6	Вход данных, бит 6	I
470/H21	DataIn5	Вход данных, бит 5	I
468/H22	DataIn4	Вход данных, бит 4	I
471/H24	DataIn3	Вход данных, бит 3	I
474/G24	DataIn2	Вход данных, бит 2	I
479/G23	DataIn1	Вход данных, бит 1	I
478/G21	DataIn0	Вход данных, бит 0	I
442/N20	RXValid	Вход «ПРИНИМАЕМЫЕ ДАННЫЕ ПРАВИЛЬНЫЕ»	I
450/N22	RXActive	Вход «ГОТОВНОСТЬ ПРИЕМА ДАННЫХ»	I
448/N23	RXError	Вход «ОШИБКА ПРИЕМА»	I
446/N24	CLK_Phy	Вход тактового сигнала от трансивера 60 МГц	I
451/M24	Reset_phy	Выход сигнала сброса трансивера	O
449/M21	VBusValid	Вход сигнала готовности USB шины	I
ARINC-429 RxTx0			
522/E1	INOUTCH0+	Входной/выходной сигнал канала 0, неинверсный	I/O
520/C1	INOUTCH0–	Входной/выходной сигнал канала 0, инверсный	I/O
ARINC-429 RxTx1			
516/B2	INOUTCH1+	Входной/выходной сигнал канала 1, неинверсный	I/O
518/D2	INOUTCH1–	Входной/выходной сигнал канала 1, инверсный	I/O
ARINC-429 RxTx2			
512/F2	INOUTCH2+	Входной/выходной сигнал канала 2, неинверсный	I/O
514/D3	INOUTCH2–	Входной/выходной сигнал канала 2, инверсный	I/O
ARINC-429 RxTx3			
511/C3	INOUTCH3+	Входной/выходной сигнал канала 3, неинверсный	I/O
513/B3	INOUTCH3–	Входной/выходной сигнал канала 3, инверсный	I/O
ARINC-429 RxTx4			
510/A3	INOUTCH4+	Входной/выходной сигнал канала 4, неинверсный	I/O
509/B4	INOUTCH4–	Входной/выходной сигнал канала 4, инверсный	I/O
ARINC-429 RxTx5			
503/C4	INOUTCH5+	Входной/выходной сигнал канала 5, неинверсный	I/O
505/D4	INOUTCH5–	Входной/выходной сигнал канала 5, инверсный	I/O

Продолжение таблицы 5.1

1	2	3	4
ARINC-429 RxTx6			
502/E4	INOUTCH6+	Входной/выходной сигнал канала 6, неинверсный	I/O
504/F4	INOUTCH6–	Входной/выходной сигнал канала 6, инверсный	I/O
ARINC-429 RxTx7			
501/C5	INOUTCH7+	Входной/выходной сигнал канала 7, неинверсный	I/O
500/A6	INOUTCH7–	Входной/выходной сигнал канала 7, инверсный	I/O
Блок специальной логики			
Технологическая шина (TechBus)			
243/AE5	AD15_TB	15 бит адреса/данных технологической шины	I/O/Z
242/AC4	AD14_TB	14 бит адреса/данных технологической шины	I/O/Z
241/AD4	AD13_TB	13 бит адреса/данных технологической шины	I/O/Z
240/AF4	AD12_TB	12 бит адреса/данных технологической шины	I/O/Z
235/AG3	AD11_TB	11 бит адреса/данных технологической шины	I/O/Z
234/AE3	AD10_TB	10 бит адреса/данных технологической шины	I/O/Z
233/AD3	AD09_TB	9 бит адреса/данных технологической шины	I/O/Z
232/AC3	AD08_TB	8 бит адреса/данных технологической шины	I/O/Z
231/AD2	AD07_TB	7 бит адреса/данных технологической шины	I/O/Z
230/AE2	AD06_TB	6 бит адреса/данных технологической шины	I/O/Z
229/AF2	AD05_TB	5 бит адреса/данных технологической шины	I/O/Z
224/AG2	AD04_TB	4 бит адреса/данных технологической шины	I/O/Z
223/AF1	AD03_TB	3 бит адреса/данных технологической шины	I/O/Z
222/AE1	AD02_TB	2 бит адреса/данных технологической шины	I/O/Z
221/AC1	AD01_TB	1 бит адреса/данных технологической шины	I/O/Z
220/AB1	AD00_TB	0 бит адреса/данных технологической шины	I/O/Z
244/AG5	INIT#_TB	Выход начального запуска	O
245/AF6	WA#_TB	Выход строба адреса	O
249/AE8	RD#_TB	Выход строба чтения данных	O
248/AF8	WD#TB	Выход строба записи данных	O
251/AG8	PKO#_TB	Выходная разовая команда	O
250/AE10	PKI_TB	Входная разовая команда	I
252/AG11	ENLPT#_TB	Технологический вход начального запуска	I
254/AE12	WPK1#_TB	Выход сигнала безадресной записи данных 1	O
253/AD13	ZPR_TB	Вход запроса прерывания процессора	I
256/AE13	ERDYn	Вход внешнего сигнала готовности периферии	I
259/AG13	RRK#_TB	Выход сигнала безадресного чтения данных	O
Сигналы управления ПЗУ			
52/N6	CEROM#_0	Выход выбора ROM от процессора 0	O
54/P6	OEROM#_0	Выход сигнала разрешения выхода ROM от процессора 0	O
53/R6	CEROM#_1	Выход выбора ROM от процессора 1	O
57/T6	OEROM#_1	Выход сигнала разрешения выхода ROM от процессора 1	O
Блок ПЧК			
261/AD19	MI	Вход тактового сигнала 1 кГц	I
262/AD20	UPXp	Входной сигнал отклонения по координате X - плюс	I
264/AD21	UPYp	Входной сигнал отклонения по координате Y - плюс	I
265/AD22	UPZp	Входной сигнал отклонения по координате Z - плюс	I
263/AC23	UMXp	Входной сигнал отклонения по координате X - минус	I
266/AC22	UMYp	Входной сигнал отклонения по координате Y - минус	I
270/AC21	UMZp	Входной сигнал отклонения по координате Z - минус	I

Продолжение таблицы 5.1

1	2	3	4
274/AC19	PPXn	Входной сигнал перегрузки по координате X - плюс	I
269/AB20	PPYn	Входной сигнал перегрузки по координате Y - плюс	I
278/AB21	PPZn	Входной сигнал перегрузки по координате Z - плюс	I
277/AB22	PMXn	Входной сигнал перегрузки по координате X - минус	I
275/AB23	PMYn	Входной сигнал перегрузки по координате Y - минус	I
281/AA23	PMZn	Входной сигнал перегрузки по координате Z - минус	I
279/AA22	T1	Выходной сигнал T1=CTT[3]&CTT[4]	O
284/AA21	T2	Выходной сигнал T2=CTT[3]&~CTT[4]&CTT[5]	O
289/AA20	T3	Выходной сигнал T3=MI&T2	O
282/Y21	T4	Выходной сигнал T4=CTP&MI. CTP=INV1&CTT[3]	O
287/Y22	C1	Выходной сигнал C1=CTT[3]	O
285/W23	C2	Выходной сигнал C2=(CT[5]&RU[2]&~RU[3] | (CT[4]&RU[2]) | (CT[3]&RU[3])	O
293/W21	C3	Выходной сигнал C3=INV0&CTT[3]	O
294/W20	C4	Выходной сигнал C4=INV1&CTT[3]	O
297/V20	FI/X2	Тактовый сигнал 16 МГц	I
299/V22	FI/X1	Выходной сигнал осциллятора	O
291/1183/U23	GND33_osc_FI	Общий вывод осциллятора FI	–
295/1184/U22	VDD33_osc_FI	Вывод питания 3,3 В осциллятора FI	–
305/U21	ST	Вход сигнала «СТОП»	I
303/U20	CTR	Вход сигнала готовности	I
Общие выводы микросхемы			
11/26/37/49/66/77/87/99/112/126/139/149/153/162/ 177/188/199/215/228/239/255/272/290/300/308/321/330/ 339/351/363/380/389/401/414/431/445/455/464/477/490/ 507/519/529/546/569/577/592/601/1101/ 1107/1113/1119/1125/1129/1135/1141/1147/1153/ 1161/1165/1171/1177/1185/1191/1197/1203/1211/ 1217/1223/1229/1235/1239/1245/1251/1257/1263/ 1271/1275/1281/1287/1293/1299/1331/1333/1335/ A5/A16/A24/B19/B20C11/D27/G14/H12/H18/H23/ J2/K1/K21/L25/N8/P1/P5/P7/T5/U3/V7/V21/V25/ W6/W8/W27/Y16/AA13/AA17/AB11/AC12/AC15/ AC16/AE7/AE11/AE17/AE27/AF15/AG4/AG21/		GND_CL	Общие выводы ядра	–

Продолжение таблицы 5.1

1	2	3	4
3/13/21/32/43/55/64/71/75/86/98/113/121/129/137/147/154/ 164/172/186/196/206/217/226/237/246/258/267/273/280/288/ 298/312/316/331/342/353/362/372/382/391/400/406/415/423/ 430/439/456/466/475/482/488/499/508/517/528/539/548/558/ 568/582/590/600/1103/1105/1109/1111/1115/1117/1121/1123/ 1127/1131/1133/1137/1139/1143/1145/1149/1151/1155/1157/ 1159/1163/1167/1169/1173/1175/1179/1181/1187/1189/1193/ 1195/1199/1201/1207/1209/1213/1215/1219/1221/1225/1227/ 1231/1233/1237/1241/1243/1247/1249/1253/1255/1259/1261/ 1265/1267/1273/1277/1279/1283/1285/1289/1291/1295/1297/ 1337/603/607/615/624/626/629/632/640/648/652/656/662/672/ 679/686/692/696/700/708/717/719/722/733/741/745/749/755/ 765/772/779/785/789/793/801/810/812/815/818/826/834/838/ 842/848/858/865/872/878/882/886/894/903/905/908/919/927/ 931/935/941/951/958/965/971/977/981/987/991/995/1001/ 1005/1009/1015/1021/1025/1029/1035/1039/1043/1049/1053/ 1057/1063/1067/1071/1077/1081/1085/1091/1095/1099/1301/ 1305/1309/1315/1319/1323/1329/605/612/619/623/633/638/ 644/651/654/660/664/671/678/685/689/695/698/705/712/716/ 724/728/730/737/744/747/753/757/764/771/778/782/788/791/ 798/805/809/819/824/830/837/840/846/850/857/864/871/875/ 881/884/891/898/902/910/914/916/923/930/933/939/943/950/ 957/964/968/974/975/979/983/985/989/993/997/999/1003/1007/ 1011/1013/1017/1023/1027/1031/1033/1037/1041/1045/1047/ 1051/1055/1059/1061/1065/1069/1073/1075/1079/1083/1087/ 1089/1093/1097/1303/1307/1311/1313/1317/1321/1325/1327/ A4/A8/A9/A14/A19/B13/C6/C17/C22/D1/E8/E12/E16/E20/F1/ F25/G10/G18/G20/G22/G25/H8/H14/H16/J1/J27/K7/L3/L20/ M5/M23/M25/N26/P20/P21/P27/R2/T1/T22/T23/U25/V27/ W26/Y5/Y23/AA2/AA10/AA14/AA18/AA25/AB24/AB25/ AC8/AC20/AD1/AE6/AE22/AF9/AF10/AF21/AG9/AG12/ AG14/AG19/AG24/	GND	Общие выводы буферов ввода-вывода процессора, ядер ОЗУ и буферов выводов ОЗУ	–
Питание микросхемы			
4/15/27/36/47/59/62/68/85/97/106/117/127/135/138/148/151/160/ 169/174/178/187/195/207/216/227/236/247/260/268/276/286/296/ 301/304/314/327/341/349/361/373/381/390/402/416/427/437/443/ 447/452/453/465/476/486/497/506/515/535/550/559/570/580/594/ 602/1104/1106/1110/1112/1116/1118/1122/1124/1128/1132/1134/ 1138/1140/1144/1146/1150/1152/1156/1158/1160/1164/1168/ 1170/1174/1176/1180/1182/1188/1190/1194/1196/1200/1202/ 1208/1210/1214/1216/1220/1222/1226/1228/1232/1234/1238/ 1242/1244/1248/1250/1254/1256/1260/1262/1266/1268/1274/ 1278/1280/1284/1286/1290/1292/1296/1298/ A10/A17/B5/B8/B14/B17/B21/B22/B25/C2/C19/C20/C23/ D22/E2/E27/G1/G2/G26/J3/J26/K27/L2/L27/M20/N21/P2/ P22/P24/P26/R26/T2/T20/T21/U24/U26/U27/V1/V24/W22/ W24/W25/AA1/AA26/AA27/AB26/AC2/AC26/AE9/AF5/ AF7/AF11/AF12/AF14/AF16/AG18/AG23/	VCC_DR	Выводы питания буферов ввода-вывода, 3,3 В	–

Окончание таблицы 5.1

1	2	3	4
9/39/56/94/109/125/142/158/181/204/225/238/257/ 271/283/292/311/320/325/338/358/378/399/413/434/ 454/473/480/494/521/532/553/564/591/1102/1108/ 1114/1120/1126/1130/1136/1142/1148/1154/1162/ 1166/1172/1178/1186/1192/1198/1204/1212/1218/ 1224/1230/1236/1240/1246/1252/1258/1264/1272/ 1276/1282/1288/1294/1300/ A15/A18/A22/A25/B6/C10/F26/G3/H11/H15/H17/J22/K2/M22/M26/P8/R1/R21/T27/V23/W7/Y2/Y18/Y20/Y27/AB12/AD25/AE4/AE21/AG10/AG15	VCC_CL	Выводы питания ядра, 1,8 В	–
609/617/627/628/631/637/645/657/658/661/668/675/ 680/687/690/691/703/711/718/720/721/731/739/750/ 751/754/761/766/773/780/783/784/795/803/813/814/ 817/823/831/843/844/847/854/861/866/873/876/877/ 889/897/904/906/907/917/925/936/937/940/947/952/ 959/966/969/970/978/982/988/992/996/1002/1006/ 1010/1016/1022/1026/1030/1036/1040/1044/1050/ 1054/1058/1064/1068/1072/1078/1082/1086/1092/ 1096/1100/1302/1306/1310/1316/1320/1324/1330/ AB9/AF3/AF13/B9/J7/N2/W1	VDDA33	Выводы питания ядра ОЗУ, 3,3 В	–
604/611/618/622/625/630/636/643/650/653/659/663/ 670/677/684/688/694/697/704/710/715/723/727/729/ 736/743/746/752/756/763/770/777/781/787/790/797/ 804/808/811/816/822/829/836/839/845/849/856/863/ 870/874/880/883/890/896/901/909/913/915/922/929/ 932/938/942/949/956/963/967/973/976/980/984/986/ 990/994/998/1000/1004/1008/1012/1014/1018/1024/ 1028/1032/1034/1038/1042/1046/1048/1052/1056/ 1060/1062/1066/1070/1074/1076/1080/1084/1088/ 1090/1094/1098/1304/1308/1312/1314/1318/1322/ 1326/1328/ AA8/AB2/AB3/AG6/AG7/B11/B15/E3/F3/H5/L6/ M1/M2/T8/U1/V2/	VDDE33	Выводы питания буферов ввода-вывода ОЗУ, 3,3 В	–
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, суффикс # после названия сигнала означает, что сигнал имеет активный уровень – низкий.			

6 Условное графическое обозначение

Условное графическое обозначение (УГО) ИС 1867ВН016 приведено на рисунке 6.1.

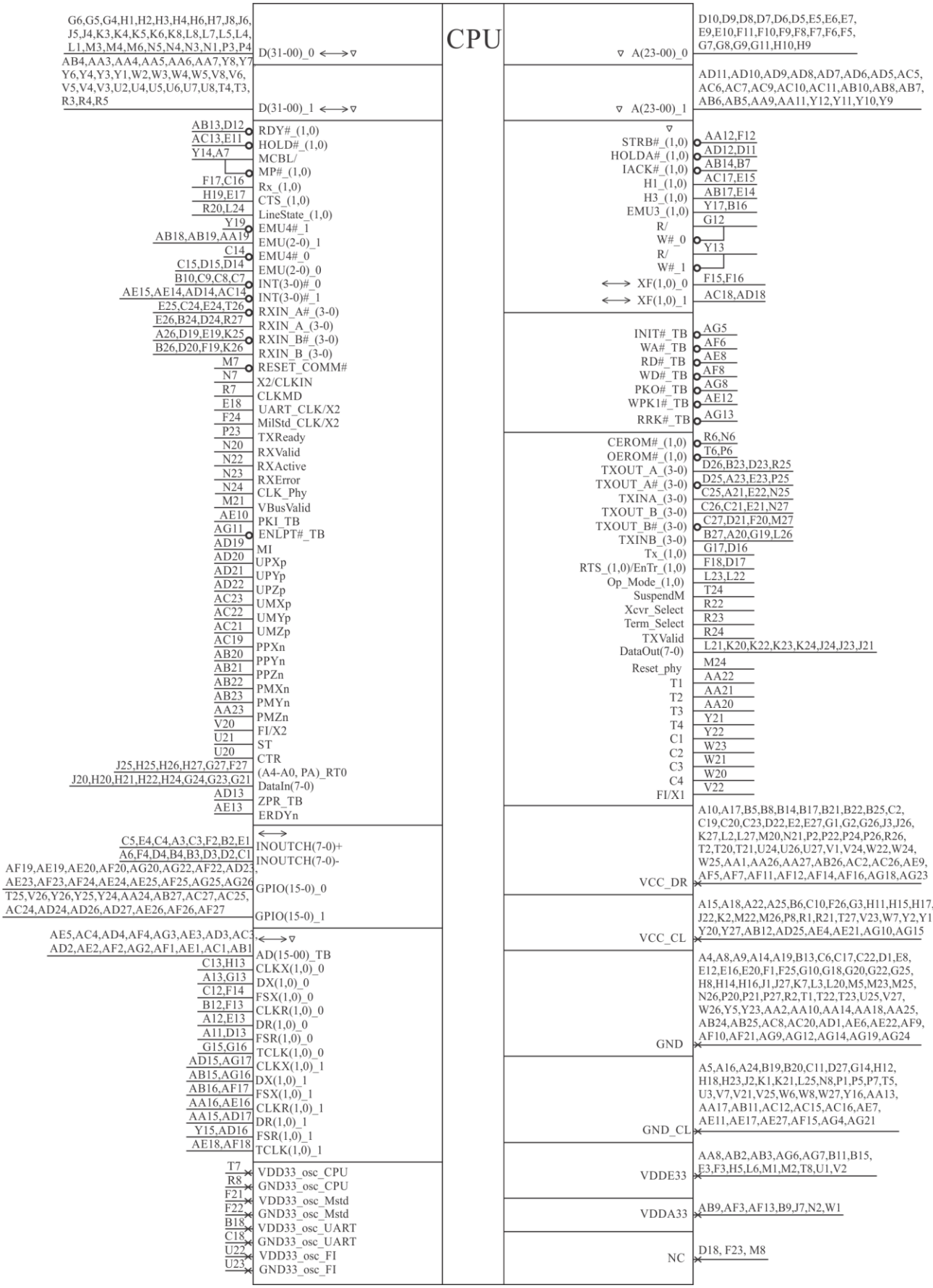


Рисунок 6.1 – Условное графическое обозначение ИС 1867ВН016

7 Корпус и схема расположения кристаллов ИС 1867ВН016

Кристаллы ИС 1867ВН016 развариваются в 602-выводной штырьковый металлокерамический корпус МК 6103.602-А. Габаритные и присоединительные размеры корпуса МК 6103.602-А приведены на рисунке 7.1. Расположение кристаллов «системы в корпусе»: кристалла высокопроизводительного двухъядерного процессора обработки сигналов (CPU) и двух кристаллов индивидуальной однопортовой памяти INTSRAM объёмом 512K × 32 (ОЗУ) – на рисунке 7.2.

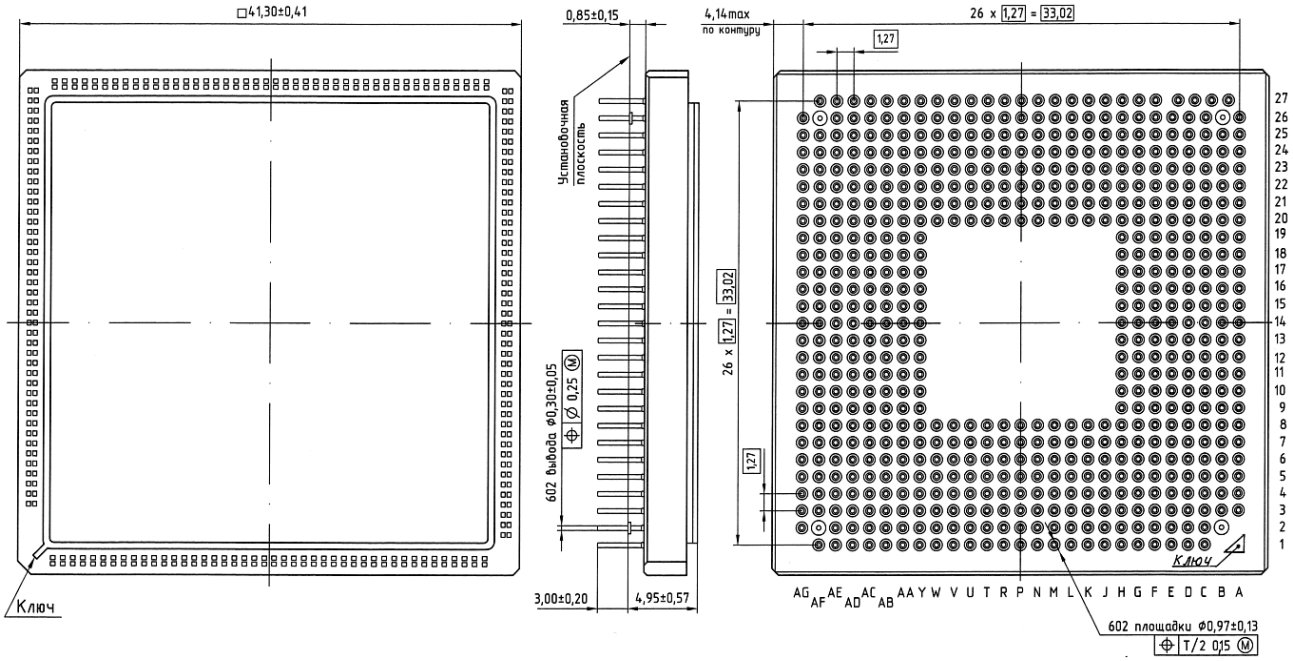


Рисунок 7.1 – Габаритные и присоединительные размеры корпуса МК 6103.602-А

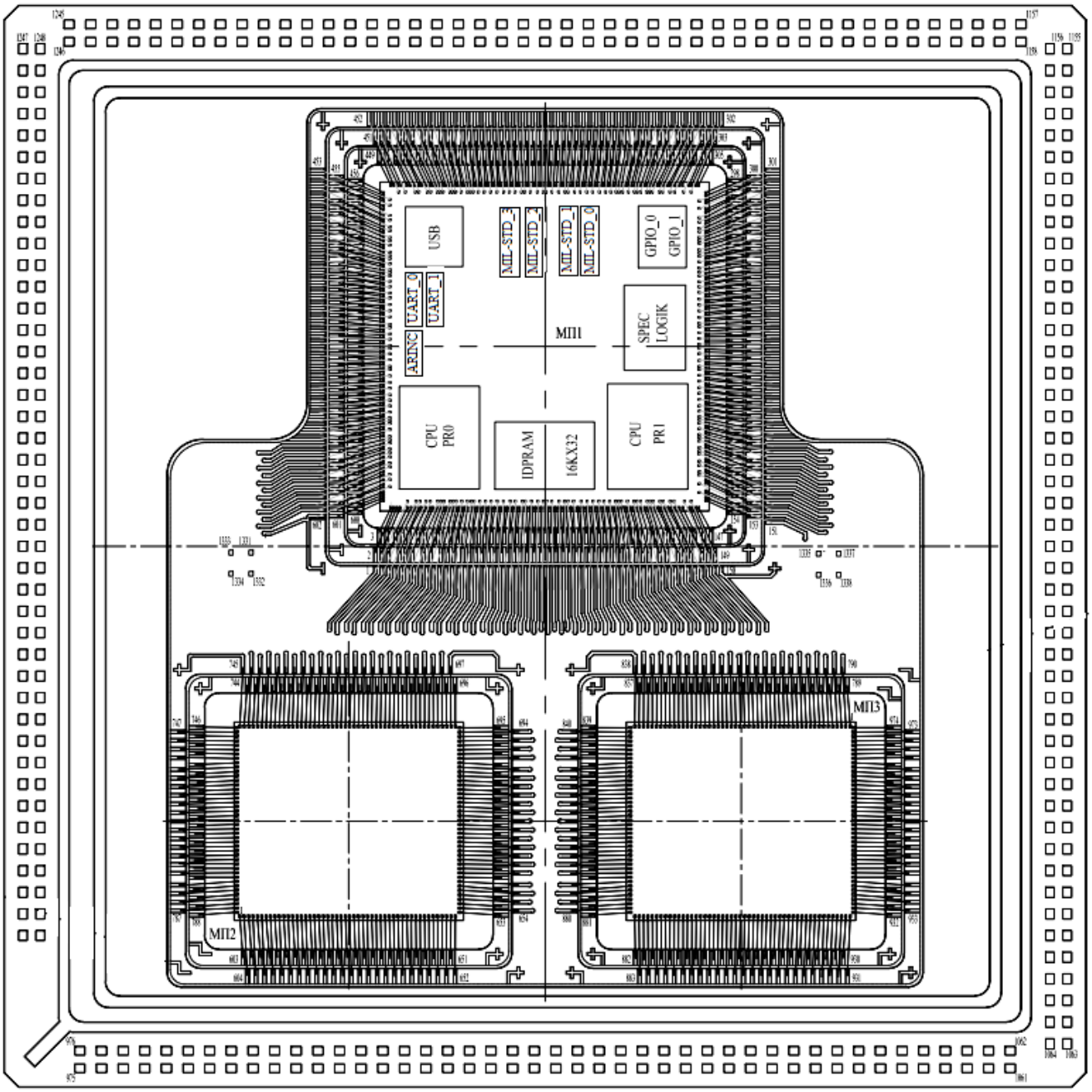


Рисунок 7.2 – Расположение кристаллов ИС 1867ВН016 в корпусе МК 6103.602-А

8 Тактирование микросхемы

ИС 1867ВН016 тактируется внешним тактовым сигналом X2/CLKIN с максимальной частотой 80 МГц для режима синхронизации с делением на 2 (CLKMD = 1) или с частотой до 40 МГц в режиме умножения на 1 (CLKMD = 0 – тестовый режим).

9 Питание микросхемы

ИС 1867ВН016 в соответствии с выбранным техпроцессом изготовления кристаллов и требованиями ТЗ запитывается от двух источников напряжения: 1,8 В ± 10 % для внутренней логики ядра микросхемы и 3,3 В ± 10 % для буферов ввода-вывода, включая блок осциллятора микросхемы.

10 Краткое описание ядра ИС и разделяемых периферийных устройств

10.1 Краткое описание процессорного ядра ИС 1867ВН016

Процессорное ядро двухъядерной «системы в корпусе» ИС 1867ВН016 представляет собой функционально законченный 32-разрядный процессор цифровой обработки сигналов с плавающей запятой. Архитектура и система команд ядра полностью совместимы с процессором 1867ВЦ6Ф (функциональный аналог DSP TMS320C30 фирмы Texas Instruments). Это означает, что программные средства (компиляторы с языков C/C++, отладчики, например, Code Composer и т. п. программные средства), которые поддерживают ИС 1867ВЦ6Ф и её функциональный аналог TMS320C30, будут также поддерживать и отладку ИС 1867ВН016.

Организация данных в архитектуре ядра ИС включает три основных типа данных: целые, целые без знака и числа в формате с плавающей запятой. Поддерживаются короткие и с одинарной точностью форматы для целых чисел со знаком и без знака, а также короткие с одинарной точностью и с повышенной точностью форматы для значений с плавающей запятой.

Реализация арифметики с плавающей запятой обеспечивает высокоточные вычисления со скоростью целочисленных, предотвращая проблемы с переполнением, выравниванием операндов и другие общие проблемы целочисленных операций.

Система команд процессорного ядра ИС 1867ВН016 оптимизирована под решение задач цифровой обработки сигналов и иные приложения, требующие высокопроизводительных вычислений. Все инструкции ядра однословные, и большинство из них выполняется за один цикл.

В дополнение к командам умножения и накопления, предусмотрен полный набор инструкций общего назначения. Система команд содержит 113 инструкций, объединённых в следующие функциональные группы:

- команды загрузки и сохранения;
- двухоперандные арифметико-логические команды;
- трёхоперандные арифметико-логические команды;
- команды программного управления;
- команды управления блокировкой;
- команды параллельных операций.

10.2 Краткое описание периферийных устройств ИС 1867ВН016

Периферийное устройство MIL-STD-1553B

Контроллер канала MIL-STD-1553B предназначен для подключения к магистральному последовательному интерфейсу электронных модулей в соответствии с ГОСТ Р 52070-2003 и выполнения функций контроллера шины (КШ), удалённого устройства (УУ) и монитора шины (МШ). Передача данных между устройствами (до 32 устройств) осуществляется через основную и резервную шины. MIL-STD-1553B обеспечивает реализацию всех команд, предусмотренных стандартом, как в режиме контроллера, так и в режиме удалённого устройства. Сообщения MIL-STD-1553B состоят из командных (CW), ответных слов (AW) и слов данных (DW). Интерфейс обеспечивает отказоустойчивую работу в реальном времени с максимальной частотой шины до 1 Мбит/с.

Периферийное устройство ARINC-429

Периферийное устройство ARINC-429 – двухпроводная компьютерная шина межсистемного обмена данных, применяемая в авионике, реализована по ГОСТ 18977–79. Размер слова составляет 32 бита, большинство сообщений состоит из единственного слова данных. ARINC-429 использует однонаправленный стандарт шины данных с физически разделёнными линиями передачи и приёма, сообщения передаются на одной из двух скоростей – 50 или 250 Кбит/с. Для каждого канала может быть установлена индивидуальная скорость приёма и передачи, включая и стандартные 12,5 и 100 Кбит/с.

Периферийное устройство UART

Периферийное устройство UART (Universal Asynchronous Receiver/Transmitter – универсальный асинхронный приёмник/передатчик), устройство, обеспечивающее последовательный обмен данными для организации взаимодействия с модемом или другими внешними устройствами с использованием протоколов RS-232 и RS-485. Блок UART реализован в соответствии с промышленным стандартом National Semiconductor`s 16550A device. ИС 1867ВН016 содержит 2 контроллера UART (NS16550A) со скоростью передачи/приёма – (50 – 3 000 000) бит/с, 8-/32-разрядными режимами работы.

Периферийное устройство USB 2.0

Периферийное устройство USB 2.0 (Universal Serial Bus Device Controller) обеспечивает интерфейс между процессорным ядром и универсальной последовательной шиной в режиме device. Интерфейс USB 2.0 поддерживает на протокольном уровне стандарт высокоскоростного – 480 Мбит/с High Speed (HS) и полноскоростного – 12 Мбит/с Full Speed (FS) обмена данными в режиме управляющих передач, в режиме передачи массивов данных, в режиме передачи по прерываниям или в изохронном режиме.

Периферийное устройство ПЧК

Периферийное устройство ПЧК (преобразователь частота-код или frequency to number converter – FNC) представляет собой специализированное устройство, реализующее аппаратное преобразование частота-код.

Периферийное устройство технологическая шина (TechBus)

Технологическая шина (ТВ) представляет собой пользовательский параллельный интерфейс с мультиплексной передачей адресов/данных и изменяемой скоростью работы, предназначенный для обеспечения доступа к специальной периферии со стороны ИС 1867ВН016.

Периферийное устройство отладочный порт

Отладочный порт ИС 1867ВН016 представляет собой специализированный последовательный тестовый интерфейс к каждому из ядер, аппаратно и программно совместимый с отладочным портом ИС 1867ВЦ6Ф. Это позволяет обеспечить отладку пользовательских кодов с использованием стандартных программных средств ИС 1867ВЦ6Ф (компиляторы с языков C/C++, отладчики, например, Code Composer), предназначенных для этих микросхем.

Периферийное устройство порт ввода-вывода общего назначения

Порты ввода-вывода общего назначения (GPIO_0, GPIO_1) ИС 1867ВН016 реализованы в виде двух блоков, каждый из которых отвечает за 16 программируемых линий, и обеспечивают суммарно 32 индивидуально управляемых вывода микросхемы. Любой из блоков GPIO_0, GPIO_1 может быть подключен через коммутатор к одному из процессоров. Каждый из блоков поддерживает прерывания, которые генерируются при изменении состояний на входах.

11 Двухпортовая память IDPRAM

11.1 Разделяемая область IDPRAM

В состав ИС 1867ВН016 входит реализованная на главном кристалле «системы в корпусе» двухпортовая память объёмом 16K × 32 бит IDPRAM, см. рисунок 3.1. Доступ (запись/чтение) к памяти IDPRAM, расположенной в диапазоне адресов 80_0000h – 80_3FFFh, осуществляется без активации внешнего интерфейса (через внутренние ресурсы системы – интерфейсы Expansion Bus процессорных ядер ПЦОС_0 и ПЦОС_1). При этом независимо от значения, записанного в регистр управления Expansion Bus (EBCR), доступ к IDPRAM осуществляется за один машинный цикл (см. подраздел 4.1 «Модифицированные карты памяти процессорного ядра ИС 1867ВН016»).

11.2 Арбитр IDPRAM

Процессорные ядра ПЦОС_0 и ПЦОС_1 могут записывать и читать любую область памяти IDPRAM, однако одновременный доступ к одним и тем же ячейкам памяти может привести к некорректным результатам. Область IDPRAM, к которой в ходе исполнения программного кода будет осуществляться доступ со стороны обоих ядер, называется критической. В такой ситуации, при разработке программы, необходимо заранее определить стартовый адрес и размер критической области (в диапазоне – от одной ячейки до максимальных 16K слов). При организации обмена с этой областью, размер которой (диапазон адресов) не фиксирован аппаратно и зависит от конкретной задачи, используется механизм семафора (более подробно см. в подразделе 12.2 «Описание регистров коммутатора»**)** и соответствующие регистры RS_SHMEM и RC_SHMEM. Для доступа к критической области необходимо выполнить инструкцию LDII @RS_SHMEM, Rx, которая формирует сигнал XF0_0/XF0_1.

Если арбитр коммутатора предоставляет доступ одному из процессоров к регистру коммутации RC, то для этого процессора содержимое RS будет равно 0000_0000h. Для смежного процессора в это же время содержимое RS будет равно 0000_0001h. Содержимое RS, равное 0000_0000h, будет означать, что доступ к регистру RC разблокирован и данный процессор может модифицировать его содержимое (записать в поле SW значение 01b для ПЦОС_0 или 10b для ПЦОС_1), т. е. «подключить» критическую область IDPRAM.

После окончания обмена с критической областью процессор должен освободить коммутатор, записав в регистр RC код 00b/11b. После системного сброса регистр коммутации RC заблокирован для модификации.

Пример процедуры доступа к разделяемому ресурсу приведён в подразделе 12.3 «Пример программирования коммутатора».

12 Арбитр и коммутатор к разделяемым периферийным устройствам

Коммутатор предназначен для коммутации (подключения) периферийных устройств (MIL-STD-1553_3, MIL-STD-1553_2, MIL-STD-1553_1, MIL-STD-1553_0, UART_1, UART_0, ARINC-429, GPIO_1, GPIO_0, USB 2.0 и TechBus) к одному из ПЦОС. Каждое из устройств имеет свои регистр семафора и регистр коммутации. 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 (номер процессора 01b/10b) в регистре коммутации RC. Доступ к регистру RC осуществляется через механизм доступа к разделяемым ресурсам, предусмотренным в ядре ПЦОС (используя инструкции interlock). 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 стадий:

- стадия 1 – проверка состояния регистра семафора RS;
- стадия 2 – захват регистра коммутации RC;
- стадия 3 – модификация регистра коммутации RC;
- стадия 4 – освобождение регистра коммутации RC.

Предоставление регистра коммутации RC запрашиваемому процессору осуществляется по принципу «первый пришёл – первый обслужен». Если два процессора одновременно затребовали разделяемый ресурс (выполняют инструкцию LDII), то разделяемый ресурс предоставляется процессору ПЦОС_0.

Все периферийные устройства самостоятельно сбрасывают сигнал запроса на прерывание, поэтому в ПЦОС_0 и ПЦОС_1 захват прерывания от периферийного устройства должен осуществляться по фронту.

Сигнал прерывания должен находиться в активном уровне в течение, как минимум, одного системного такта. После коммутации периферийного устройства к одному из ядер прерывания от скоммутированного устройства поступают на соответствующие входы прерываний ПЦОС.

Адреса векторов прерываний, поступающих от периферийных устройств, приведены в таблице 12.1.

Таблица 12.1 – Карта векторов прерываний ядра «системы в корпусе» ИС 1867ВН016

Адрес вектора	Имя	Функция
1	2	3
Векторы прерывания от внешних источников и неразделяемых (внутриядерных) периферийных устройств		
00h	RESET	Прерывание от RESET_COMM#
01h	INT0	Прерывание от вывода INT0
02h	INT1	Прерывание от вывода INT1
03h	INT2	Прерывание от вывода INT2
04h	INT3	Прерывание от вывода INT3
05h	XINT0	Прерывание от последовательного порта 0, если буфер передатчика пуст
06h	RINT0	Прерывание от последовательного порта 0, если буфер приёмника полон
07h	XINT1	Прерывание от последовательного порта 1, если буфер передатчика пуст
08h	RINT1	Прерывание от последовательного порта 1, если буфер приёмника полон
09h	TINT0	Прерывание, генерируемое таймером 0
0Ah	TINT1	Прерывание, генерируемое таймером 1
0Bh	DINT	Прерывание, генерируемое контроллером DMA
Векторы прерывания разделяемых между ядрами периферийных устройств (п/у)		
0Ch	ZPR_TB/ INTINTR	Прерывание от п/у TechBus/ Внутреннее прерывание от смежного процессора
0Dh	RQVST	Запрос на обслуживание/ПЧК
0Eh	ACKN	Подтверждение обслуживания
0Fh	MILStd1553_0	Прерывание от п/у MIL-STD-1553_0
10h	MILStd1553_1	Прерывание от п/у MIL-STD-1553_1
11h	MILStd1553_2	Прерывание от п/у MIL-STD-1553_2
12h	MILStd1553_3	Прерывание от п/у MIL-STD-1553_3
13h	USB 2.0	Прерывание от п/у USB 2.0
14h	ARINCRxTx0	Прерывание от канала 0 (приём/передача) п/у ARINC-429
15h	ARINCRxTx1	Прерывание от канала 1 (приём/передача) п/у ARINC-429
16h	ARINCRxTx2	Прерывание от канала 2 (приём/передача) п/у ARINC-429
17h	ARINCRxTx3	Прерывание от канала 3 (приём/передача) п/у ARINC-429
18h	ARINCRxTx4	Прерывание от канала 4 (приём/передача) п/у ARINC-429
19h	ARINCRxTx5	Прерывание от канала 5 (приём/передача) п/у ARINC-429
1Ah	ARINCRxTx6	Прерывание от канала 6 (приём/передача) п/у ARINC-429
1Bh	ARINCRxTx7	Прерывание от канала 7 (приём/передача) п/у ARINC-429
1Ch	UART0	Прерывание от п/у UART_0
1Dh	UART1	Прерывание от п/у UART_1
1Eh	GPIO0	Прерывание от входов/выходов общего назначения GPIO_0
1Fh	GPIO1	Прерывание от входов/выходов общего назначения GPIO_1

12.1 Архитектура коммутатора

Для каждого из разделяемых периферийных устройств коммутатор состоит из двух программно доступных регистров:

- регистра семафора RS;
- регистра коммутации RC.

Регистр семафора RS предназначен для доступа к регистру коммутации RC. Процессорное ядро, которое подключает к своей шине (ExpBus_0/ExpBus_1) одно из разделяемых периферийных устройств (MIL-STD-1553_3, MIL-STD-1553_2, MIL-STD-1553_1, MIL-STD-1553_0, UART_1, UART_0, ARINC-429, GPIO_1, GPIO_0, USB 2.0, ПЧК и TechBus), должно выполнить инструкцию LDII, формирующую сигнал XF0_0/XF0_1. Если арбитр коммутатора предоставляет доступ ПЦОС_0 или ПЦОС_1 к регистру коммутации RC, то для этого процессора содержимое RS будет равно 0000_0000h. Для другого процессора в это время содержимое RS будет равно 0000_0001h. Содержимое RS, равное 0000_0000h, будет означать, что доступ к регистру RC разрешён и данный процессор может модифицировать его содержимое.

Регистр коммутации RC предназначен для коммутации (подключения) любого из внешних устройств к своему интерфейсу (ExpBus) путём занесения соответствующего кода в этот регистр. После системного сброса разделяемые периферийные устройства не подключены ни к одному из процессоров. Если разделяемое периферийное устройство подключено к процессору, то выход прерывания от этого устройства коммутируется к соответствующему выводу прерывания процессора. При возникновении в периферийном устройстве прерывания, если его сигнал не замаскирован в процессоре, активируется соответствующий вектор, см. таблицу 12.1.

12.2 Описание регистров коммутатора

В таблице 12.2 приведён список регистров коммутатора с адресами.

Таблица 12.2 – Список регистров коммутатора

Адрес	Название	Функциональное назначение
80_4580h	RS_GPIO0	Регистр семафора GPIO_0
80_4581h	RC_GPIO0	Регистр коммутации GPIO_0
80_4582h	RS_GPIO1	Регистр семафора GPIO_1
80_4583h	RC_GPIO1	Регистр коммутации GPIO_1
80_4584h	RS_MNCH0	Регистр семафора MIL-STD-1553_0
80_4585h	RC_MNCH0	Регистр коммутации MIL-STD-1553_0
80_4586h	RS_MNCH1	Регистр семафора MIL-STD-1553_1
80_4587h	RC_MNCH1	Регистр коммутации MIL-STD-1553_1
80_4588h	RS_MNCH2	Регистр семафора MIL-STD-1553_2
80_4589h	RC_MNCH2	Регистр коммутации MIL-STD-1553_2
80_458Ah	RS_MNCH3	Регистр семафора MIL-STD-1553_3
80_458Bh	RC_MNCH3	Регистр коммутации MIL-STD-1553_3
80_458Ch	RS_USB	Регистр семафора USB 2.0
80_458Dh	RC_USB	Регистр коммутации USB 2.0
80_458Eh	RS_ARINC429	Регистр семафора ARINC-429
80_458Fh	RC_ARINC429	Регистр коммутации ARINC-429
80_4590h	RS_UART0	Регистр семафора UART_0
80_4591h	RC_UART0	Регистр коммутации UART_0
80_4592h	RS_UART1	Регистр семафора UART_1
80_4593h	RC_UART1	Регистр коммутации UART_1
80_4594h	RS_SHMEM	Регистр семафора критической секции разделяемой памяти
80_4595h	RC_SHMEM	Регистр коммутации критической секции разделяемой памяти
80_4596h	RS_TECHBUS	Регистр семафора TechBus
80_4597h	RC_TECHBUS	Регистр коммутации TechBus
80_4598h	RS_PCHK	Регистр семафора ПЧК
80_4599h	RC_PCHK	Регистр коммутации ПЧК

На рисунке 12.1 приведена структура регистра семафора RS, а в таблице 12.3 – описание его разрядов.

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
RSV	RSV	RSV	RSV	RSV	RSV	RSV	S
R-0	R-0	R-0	R-0	R-0	R-0	R-0	RC-1

Примечание – Принятые условные обозначения: R – бит только читается, C – бит сбрасывается, 0/1 – значение после сброса.

Рисунок 12.1 – Структура регистра семафора RS разделяемого периферийного устройства

Таблица 12.3 – Описание битов регистра семафора RS

Бит	Имя	Доступ	Описание
31 – 1	RSV	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 этого регистра с использованием любой инструкции кроме LDII (сигнал XF0_0/XF0_1 не установлен) всегда возвращает 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 (RS – адрес регистра семафора соответствующего устройства). Если содержимое регистра RS = 0000_0000h, то запись в регистр коммутации RC разблокирована; если содержимое регистра RS = 0000_0001h, то устройство занято другим процессором и запись в регистр коммутации RC заблокирована. Для разблокировки регистра коммутации RC необходимо, используя команду STII RX, @RC, записать код 00b/11b в регистр коммутации RC, обеспечив возможность модификации регистра коммутации со стороны другого процессора. Если в поле SW записано значение или 01b, или 10b, то запись в это поле разрешена только процессору, к которому подключено разделяемое периферийное устройство. При записи 00b/11b в это поле бит семафора устанавливается в 1.
Примечание – Принятые условные обозначения: R – бит только читается, C – бит сбрасывается.			

На рисунке 12.2 приведена структура регистра коммутатора RC, а в таблице 12.4 – описание его разрядов.

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
RSV	RSV	RSV	RSV	RSV	RSV	SW	SW
R-0	R-0	R-0	R-0	R-0	R-0	RWC-0	RWC-0

Примечание – Принятые условные обозначения: R – бит только читается, RWC – бит доступен на чтение, запись и сбрасывается при сбросе, 0/1 – значение после сброса.

Рисунок 12.2 – Структура регистра коммутатора RC разделяемого периферийного устройства

Таблица 12.4 – Описание битов регистра коммутатора RC

Бит	Имя	Доступ	Описание
31 – 2	RSV	R	Резервные биты. Читаются как 0.
1 – 0	SW	RWC	Поле коммутации SW: - SW = 01b – периферийное устройство подключено к ПЦОС_0; - SW = 10b – периферийное устройство подключено к ПЦОС_1; - SW = 00b/11b – разделяемое периферийное устройство не подключено ни к ПЦОС_0, ни к ПЦОС_1. После сброса поле SW = 00b. Внимание – При обращении к регистрам неподключенного разделяемого периферийного устройства из ПЦОС (на чтение или запись), сигналы RDY#_0 или RDY#_1 не формируются, и может произойти «зависание» процессора, если поле регистра управления интерфейса Primary Bus SW = 00b/11b. Запись в это поле заблокирована, если бит S = 1.
Примечание – Принятые условные обозначения: R – бит только читается, RWC – бит доступен на чтение, запись и сбрасывается при сбросе.			

12.3 Пример программирования коммутатора

Для коммутации одного или нескольких устройств (MIL-STD-1553_3, MIL-STD-1553_2, MIL-STD-1553_1, MIL-STD-1553_0, UART_1, UART_0, ARINC-429, GPIO_1, GPIO_0, USB 2.0, ПЧК и TechBus) необходимо выполнить следующую последовательность действий:

- прочитать регистр семафора RS, используя инструкцию LDII;

- модифицировать соответствующие биты регистра RC для коммутации выбранного периферийного устройства к ПЦОС;

- выполнить необходимые действия с периферийным устройством;

- разблокировать регистр коммутации RC, для этого в RC необходимо записать 00b/11b с помощью инструкции STII.

Далее приведён пример программного кода для осуществления коммутации GPIO_0 к процессору ПЦОС_0.

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:

L1: ldii @RS, R0
  cmpi  0, R0;
    bnz  L1; переход на метку L1, если RS занят
  rets
;/////////////////////////////////////////////Конец фрагмента программы////////////////////////////////
```

Коммутация периферийного устройства осуществляется с помощью следующего макроса.

```
;/////////////////////////////////////////////Начало фрагмента программы////////////////////////////////
Connect .macro
  call GetRS; захват семафора
  ldi  *AR7,  R1; загружаем значение RC в R1
; модификация R1
; установка соответствующего значения в поле SW регистра коммутации RC

  .endm
;/////////////////////////////////////////////Конец фрагмента программы////////////////////////////////
```

После выполнения данного макроса можно обращаться к регистрам периферийного устройства.

Далее происходит работа с выбранным периферийным устройством.

Для освобождения периферийного устройства (отключения от ПЦОС) можно воспользоваться следующим макросом.

```
;/////////////////////////////////////////////Начало фрагмента программы////////////////////////////////
LDI 0, R1
stii R1, @RC; разблокировка регистра коммутации
  .endm
;/////////////////////////////////////////////Конец фрагмента программы////////////////////////////////
```

13 Периферийное устройство MIL-STD-1553B

Контроллер канала (контроллер магистрального интерфейса) MIL-STD-1553B предназначен для подключения к магистральному последовательному интерфейсу электронных модулей в соответствии с ГОСТ Р 52070-2003 и выполнения функций контроллера шины (КШ), оконечного (далее – удалённого) устройства (УУ) и монитора шины (МШ). Причём возможность одновременного функционирования в различных режимах может использоваться при тестировании MIL-STD-1553B.

MIL-STD-1553B является стандартом для передачи данных между устройствами (до 32 устройств) через общие, обычно дублированные (основную и резервную), шины. MIL-STD-1553B подключается к двум шинам магистрального интерфейса по схеме подключения ответвителя с согласующим трансформатором. MIL-STD-1553B обеспечивает реализацию всех команд, предусмотренных стандартом, как в режиме контроллера, так и в режиме удалённого устройства.

Сообщения, которыми обменивается MIL-STD-1553B с другими устройствами интерфейса, состоят из командных (CW), ответных слов (AW) и слов данных (DW). Стандарт разработан для предсказуемой, отказоустойчивой работы в реальном времени с максимальной частотой шины до 1 Мбит/с.

Одним из устройств на общей шине должен быть контроллер шины, он управляет трафиком на шине. Остальные – удалённые устройства, которые выполняют команды контроллера шины. Каждому удалённому устройству соответствует уникальный адрес в диапазоне 0 – 30. На шине также может находиться пассивный монитор шины.

Существует 5 возможных типов передачи данных по шине MIL-STD-1553B:

- КШ – УУ передача (приём);
- УУ – КШ передача (отправка);
- УУ – УУ передача;
- КШ – всем УУ;
- УУ – всем УУ.

При каждой передаче может передаваться от одного до 32 16-разрядных слов данных. Контроллер шины может также отправлять коды режимов удалённым устройствам для выполнения таких заданий, как синхронизация или чтение статуса устройства.

13.1 Режимы работы

Каждый из четырёх блоков MIL-STD-1553B содержит: контроллер шины, удалённое устройство и монитор шины, управляемые одним 1553 кодеком. За исключением блока MIL-STD-1553_0 все режимы работы MIL-STD-1553B: запуск, остановка КШ, МШ, УУ, а также задание адреса УУ, – задаются с помощью соответствующих регистров. Для блока MIL-STD-1553_0, кроме этого, предусмотрен вариант задания адреса удалённого устройства (УУ) с помощью внешних выводов ИС (см. таблицу 5.1: входы A***n***_RT0, где ***n*** от 0 до 4, и PA_RT0). Блоки и УУ не могут быть активны на шине 1553 одновременно. В то время как контроллер шины запущен (или приостановлен), только он (возможно ещё монитор) имеет доступ к шине 1553, и УУ могут только получать и отвечать на команды КШ до тех пор, пока оба расписания КШ не будут полностью остановлены. Монитор шины только принимает передачи данных, работая независимо от разрешённого/запрещённого состояния двух других частей.

Все четыре блока MIL-STD-1553B полностью конфигурируются и управляются через картированные в памяти регистры, см. подраздел 13.5 «Регистры MIL-STD-1553B». Каждый из блоков (КШ, УУ и МШ) имеет свой набор регистров, также есть несколько общих регистров. Некоторые поля управляющих регистров КШ и УУ защищены ключом, без правильной записи ключевых разрядов нельзя записать данные в другие разряды. Назначение ключевых полей – дать разработчикам гарантии, что программное обеспечение не отреагирует на трафик шины при разрешении КШ или изменении адреса УУ. Если программа написана без учёта ключей, то вероятность правильной записи ключевых полей крайне мала.

Прерывание

MIL-STD-1553B имеет один вывод прерывания, которое может генерироваться различными источниками (событиями). Содержимое регистра маски прерываний определяет, по какому событию произойдёт прерывание.

Кодек

Внутренний кодек MIL-STD-1553B принимает и передаёт данные по шине 1553, он генерирует и проверяет чётность и синхронизацию.

Логика тестирования обратной передачи контролирует, чтобы каждое отправленное слово появилось на входе получателя. Если переданное слово не получено обратно, то передатчик останавливается и сигнализирует об ошибке.

Формат списка передач

Список передач КШ представляет собой массив, состоящий из дескрипторов и переходов, как показано в таблице 13.1. Каждая запись массива должна быть выровнена по 128-битной (16 байт) границе. Два неиспользованных слова при переходе могут быть использованы программным обеспечением для хранения служебных данных.

Таблица 13.1 – Формат дескриптора передачи

Смещение	Значение дескриптора передачи	ПДП	Значение для перехода	ПДП
0x00	Слово передачи 0, см. рисунок 13.1	Чтение	Условие, см. рисунок 13.4	Чтение
0x01	Слово передачи 1, см. рисунок 13.2	Чтение	Адрес перехода, 128-битное выравнивание	Чтение
0x02	Указатель на буфер данных, 16-битное выравнивание. Для буферов записи, если бит 0 установлен, полученные данные игнорируются. Это используется при УУ – УУ передачах, где КШ не использует передаваемые данные	Чтение	Не используется	–
0x03	Слово результата, пишется MIL-STD-1553B, см. рисунок 13.3	Запись	Не используется	–

13.2 Управление контроллером шины MIL-STD-1553B

Когда MIL-STD-1553B является контроллером шины (далее – КШ), он выступает главным устройством шины (master) – инициирует и выполняет передачи данных. Этот режим работает по концепции выполнения расписания списка передач. Программа помещает в память последовательность дескрипторов передачи и переходов, буферов данных для отправки и приёма, кольцевой буфер указателей источников прерываний. Когда расписание запускается через запись регистра действий КШ, MIL-STD-1553B выполняет список действий расписания, осуществляет передачи данных, одну за другой, обновляет статус передачи, записывает принятые данные в соответствующий буфер.

Управление временем

В каждом дескрипторе передачи в расписании есть поле «слота времени». Если передача завершилась раньше, чем её слот времени, то MIL-STD-1553B приостанавливает работу на оставшееся время до следующей команды. Это позволяет пользователю точно управлять временем передачи.

Если передача занимает времени больше его слота, то время отклонения вычитается из слота времени следующей команды. Команда может занять время следующей команды и так до тех пор, пока общий баланс времени остаётся положительным.

Чтобы выполнить расписание как можно быстрее, необходимо установить в ноль все временные слоты расписания. Если необходимо сгруппировать несколько передач, следует указать один слот времени для каждой передачи.

Расписание может быть остановлено или отложено записью в регистр действий КШ. Когда расписание отложено, время продолжает учитываться, таким образом, при возобновлении расписания оно будет корректным. При остановке расписания время сбрасывается.

Когда установлен бит внешней синхронизации в следующем дескрипторе передачи, MIL-STD-1553B для запуска команды будет ждать положительного фронта сигнала на внешнем входе синхронизации. Если сигнал синхронизации приходит до достижения соответствующей передачи в расписании, то он запоминается, и передача осуществляется сразу без задержки. Сигналы синхронизации запоминаются при откладывании расписания, но очищаются при остановке расписания. Синхронизация может осуществляться программно через регистр действий КШ.

Выбор шины

Каждый дескриптор передачи имеет бит выбора шины, который позволяет выбрать шину А (бит выбора шины дескриптора – 0) или шину В (бит выбора шины дескриптора – 1) для осуществления передачи. Альтернативный вариант управления выбором шины осуществляется посредством регистра замены шины удалённого устройства (далее – УУ), в котором есть по одному биту для каждого адреса УУ.

Запись единицы в соответствующий бит регистра замены шины инвертирует значение бита выбора шины передачи для всех передач соответствующего УУ. Таким образом, 0 означает теперь шину В, а 1 – шину А. Это позволяет выбрать передающую шину для одного или для нескольких УУ одновременно с помощью одной записи в регистр, без необходимости модифицировать дескрипторы.

MIL-STD-1553B путём «Исключающего ИЛИ» между битом регистра замены шины и битом выбора шины дескриптора определяет, какая шина будет использоваться. Обычно имеет смысл использовать какой-либо один из этих двух способов для каждого УУ: либо бит выбора шины дескриптора всегда равен 0 и выбор шины определяется битами регистра замены шины, либо наоборот биты регистра замены шины равны 0 и используются биты выбора шины дескриптора.

Если для выбора шины используется регистр замены шины, то бит сохранения шины дескриптора может автоматически обновлять регистр в зависимости от исхода передачи. При успешной передаче по шине А бит регистра замены шины устанавливается в 0, в случае успешной передачи по шине В он устанавливается в 1. В случае неудачной передачи в регистре замены шины устанавливается противоположное значение.

Второй список передач

Для MIL-STD-1553B в дополнение к обычному списку передач может быть задан второй асинхронный список того же формата, что и первый. Этот список передач может быть запущен в любое время при выполнении основного списка. В процессе ожидания слота времени запланированной команды, MIL-STD-1553B проверяет слот времени следующей передачи асинхронного списка. Если слот времени меньше времени ожидания следующей передачи основного списка, то запускается на выполнение команда асинхронного списка.

Если асинхронная команда не будет выполнена вовремя, то время будет заимствовано у следующей команды основного списка. Таким образом, чтобы не разрушить расписание основного списка, время слота для асинхронного списка следует указывать исходя из пессимистичного варианта.

Исключающий бит дескриптора передачи позволяет запретить запуск команды из асинхронного списка в процессе ожидания времени запуска команды из основного списка.

Асинхронные команды не могут быть запущены при ожидании сигнала синхронизации основного расписания, или когда основное расписание отложено, и текущий слот времени истёк.

Генерация прерываний

Каждая команда в расписании передач может быть запрограммирована на генерацию прерывания после завершения передачи (без ошибки или с ошибкой). Некорректная команда всегда генерирует прерывание и останавливает расписание. Перед тем как будет сгенерировано прерывание передачи данных, адрес соответствующего дескриптора записывается в кольцевой буфер источников прерываний КШ; при этом значение регистра позиции буфера источников прерываний инкрементируется.

Возникновение отдельного прерывания, сигнализирующего ошибку, означает ошибку ПДП. Если ПДП ошибка возникает при чтении/записи дескриптора, то выполняемое расписание откладывается. ПДП ошибки в буфере данных вызывают завершение соответствующей передачи с ошибкой (коды ошибок приведены в таблице 13.4).

Любые из этих прерываний вызывают запрос на прерывание по шине AMBA, которое управляется установками регистра маски прерываний.

31	30	29	28	27	26	25	24
0	WTRIG	EXCL	IRQE	IRQN	SUSE	SUSN	RETMD1
R	RW	RW	RW	RW	RW	RW	RW
23	22	21	20	19	18	17	16
RETMD0	NRET2	NRET1	NRET0	STBUS	GAP	RSV	RSV
RW	RW	RW	RW	RW	RW	R	R
15	14	13	12	11	10	9	8
STIME15	STIME14	STIME13	STIME12	STIME11	STIME10	STIME9	STIME8
RW	RW	RW	RW	RW	RW	RW	RW
7	6	5	4	3	2	1	0
STIME7	STIME6	STIME5	STIME4	STIME3	STIME2	STIME1	STIME0
RW	RW	RW	RW	RW	RW	RW	RW

Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.

Рисунок 13.1 – Слово 0 дескриптора передачи КШ (смещение 0x00)

Таблица 13.2 – Описание битов слова 0 дескриптора передачи КШ (смещение 0x00)

Бит	Обозначение	Тип	Описание
31	0	R	Должен быть нулём для идентификации дескриптора.
30	WTRIG	RW	Ожидание внешнего действия.
29	EXCL	RW	Исключающий слот времени. Асинхронные сообщения запрещены.
28	IRQE	RW	Запрос прерывания только в случае ошибки.
27	IRQN	RW	Запрос прерывания; всегда прерывание после передачи.
26	SUSE	RW	Отложить в случае ошибки. Отложить расписание (или остановить асинхронный список) при ошибке.
25	SUSN	RW	Приостановить. Всегда приостанавливать расписание после передачи.
24 – 23	RETMD1, RETMD0	RW	Режим повтора: - 00 – повторная передача только по той же шине; - 01 – повторная передача по другой шине; - 10 – повторная передача по той же шине, затем по другой; - 11 – не используется.
22 – 20	NRET2 – NRET0	RW	Число повторных передач. Число автоматических попыток передач для шины. Общее число попыток (включая первую попытку) NRET + 1 для RETMD = 00; 2 × (NRET + 1) для RETMD = 01/10.
19	STBUS	RW	Сохранение шины. При успешной передаче, если этот бит установлен, сохраняется шина (0 для шины A, 1 для шины B) в регистре замены шины УУ. В случае неудачной передачи, если этот бит установлен, в регистре замены шины УУ выбирается противоположное значение. См. секцию описания УУ.
18	GAP	RW	Дополнительный промежуток между сообщениями. Если установлен, то после передачи добавляется дополнительное время в соответствии с полем RTTO (биты 29 – 26 слова 1 дескриптора передачи КШ, таблица 13.3).
17 – 16	RSV	R	Зарезервировано.
15– 0	STIME15 – STIME0	RW	Время слота. Выделенное время в единицах по 4 микросекунды, оставшееся время после вставленной задержки.
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

31	30	29	28	27	26	25	24
DUM	BUS	RTTO3	RTTO2	RTTO1	RTTO0	RTAD24	RTAD23
RW	RW	RW	RW	RW	RW	RW	RW
23	22	21	20	19	18	17	16
RTAD22	RTAD21	RTAD20	RTSA24	RTSA23	RTSA22	RTSA21	RTSA20
RW	RW	RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8
RTAD14	RTAD13	RTAD12	RTAD11	RTAD10	TR	RTSA14	RTSA13
RW	RW	RW	RW	RW	RW	RW	RW
7	6	5	4	3	2	1	0
RTSA12	RTSA11	RTSA10	WCMC4	WCMC3	WCMC2	WCMC1	WCMC0
RW	RW	RW	RW	RW	RW	RW	RW

Примечание – Принятое условное обозначение: RW – чтение и запись.

Рисунок 13.2 – Слово 1 дескриптора передачи КШ (смещение 0x01)

Таблица 13.3 – Описание битов слова 1 дескриптора передачи КШ (смещение 0x01)

Бит	Обозначение	Тип	Описание
31	DUM	RW	Пустая передача. Если установлен в 1, то передача сразу «успешно завершается», трафик на шине не генерируется.
30	BUS	RW	Выбор шины для передачи: - 0 – шина А; - 1 – шина В.
29 – 26	RTTO3 – RTTO0	RW	Дополнительная задержка к номинированной квантами по 4 микросекунды (0000 – 14 мкс, 1111 – 74 мкс); также используется как задержка между сообщениями передачи КШ, таблица 13.2).
25 – 21	RTAD24 – RTAD20	RW	Второй адрес УУ для передач типа УУ – УУ; конфигурирование полей RTAD1, RTSA1, RTAD2, RTSA2, WCMC, TR для различных типов передач приведено в таблице 13.5
20 – 16	RTSA24 – RTSA20	RW	Второй подадрес УУ для передач типа УУ – УУ.
15 – 11	RTAD14 – RTAD10	RW	Адрес УУ.
10	TR	RW	Отправка/Приём.
9 – 5	RTSA14 – RTSA10	RW	Подадрес УУ.
4 – 0	WCMC4– WCMC0	RW	Число слов/Код режима для различных типов передач.
Примечание – Принятое условное обозначение: RW – чтение и запись.			

31	30	29	28	27	26	25	24
0	RSV	RSV	RSV	RSV	RSV	RSV	RSV
RW	R	R	R	R	R	R	R
23	22	21	20	19	18	17	16
RT2ST7	RT2ST6	RT2ST5	RT2ST4	RT2ST3	RT2ST2	RT2ST1	RT2ST0
RW	RW	RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8
RTST7	RTST6	RTST5	RTST4	RTST3	RTST2	RTST1	RTST0
RW	RW	RW	RW	RW	RW	RW	RW
7	6	5	4	3	2	1	0
RETCNT3	RETCNT2	RETCNT1	RETCNT0	RSV	TFRST2	TFRST1	TFRST0
RW	RW	RW	RW	R	RW	RW	RW

Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.

Рисунок 13.3 – Слово результата дескриптора передачи (смещение 0x03)

Таблица 13.4 – Описание битов слова результата дескриптора передачи (смещение 0x03)

Бит	Обозначение	Тип	Описание
31	0	RW	Всегда пишется 0.
30 – 24	RSV	R	Зарезервировано, при чтении следует маскировать.
23 – 16	RT2ST7 – RT2ST0	RW	Статусные биты от принимающего УУ при УУ – УУ передаче: (в противном случае 0) - 15 – ошибка сообщения; - 14 – установлен зарезервированный бит; - 13 – сервисный запрос; - 12 – получена общая команда; - 11 – бит занятости; - 10 – флаг подсистемы; - 9 – принятие динамического управления шиной; - 8 – флаг терминала.
15 – 8	RTST7 – RTST0	RW	Статусные биты УУ (передающее УУ при УУ – УУ передаче); значения такие же, как в предыдущем поле (RT2ST).
7 – 4	RETCNT3 – RETCNT0	RW	Число выполненных повторных попыток.
3	RSV	R	Зарезервировано, при чтении следует маскировать.
2 – 0	TFRST2 – TFRST0	RW	Результат передачи (последний): - 000 – успешная передача (или бит DUM = 1, см таблицу 13.3.); - 001 – УУ не отвечает (передающее при УУ – УУ передаче); - 010 – получающее УУ при УУ – УУ передаче не отвечает; - 011 – слово состояния УУ имеет ошибку сообщения, занято, установлен зарезервированный бит. Возникает, когда часть данных верна, в противном случае генерируется код 100. Может возникнуть при выполнении команд: - «Передать последнее слово состояния» или «Передать последнюю команду», так как команды не сбрасывают слово состояния; - 100 – ошибка протокола (ошибка декодера, неверное число слов данных и др.); - 101 – неверный формат дескриптора передачи; - 110 – ошибка буфера данных ПДП; - 111 – передача отменена из-за неверной контрольной суммы.
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

Таблица 13.5 – Конфигурирование различных типов передач КШ

Тип передачи	RTAD1 (15-11)	RTSA1 (9-5)	RTAD2 (25-21)	RTSA2 (20-16)	WCMC (4-0)	TR (10)	Буфер данных, направление
Данные, КШ – УУ	УУ адрес (0–30)	УУ подадрес (1–30)	Не важно	0	Число слов (0–32)	0	Чтение (2-64 байта)
Данные, УУ – КШ	УУ адрес (0–30)	УУ подадрес (1-30)	Не важно	0	Число слов (0–32)	1	Запись (2-64 байта)
Данные, УУ – УУ	Адрес УУ получателя (0–30)	Подадрес УУ получателя (1–30)	Адрес УУ отправителя (0–30)	Подадрес УУ отправителя (1–30)	Число слов (0–32)	0	Запись (2-64 байта)
Режим, нет данных	УУ адрес (0–30)	0 или 31	Не важно	Не важно	Режим (0–8)	1	Не используется
Режим, УУ – КШ	УУ адрес (0–30)	0 или 31	Не важно	Не важно	Режим (16/18/19)	1	Запись (2 байта)
Режим, КШ – УУ	УУ адрес (0–30)	0 или 31	Не важно	Не важно	Режим (17/20/21)	0	Чтение (2 байта)
Общий режим, данные, КШ – все УУ	31	УУ подадрес (1–30)	Не важно	0	Число слов (0–32)	0	Чтение (2-64 байта)
Общий режим, данные, УУ – все УУ	31	Подадрес УУ получателя (1–30)	Адрес УУ отправителя (0–30)	Подадрес УУ отправителя (1–30)	Число слов (0–32)	0	Запись (2-64 байта)
Общий режим, нет данных	31	0 или 31	Не важно	Не важно	Режим (1, 3–8)	1	Не используется
Общий режим, КШ – УУ	31	0 или 31	Не важно	Не важно	Режим (17/20/21)	0	Чтение (2 байта)

31	30	29	28	27	26	25	24
1	RSV	RSV	RSV	RSV	IRQC	ACT	MODE
R	R	R	R	R	RW	RW	RW
23	22	21	20	19	18	17	16
RT2CC7	RT2CC6	RT2CC5	RT2CC4	RT2CC3	RT2CC2	RT2CC1	RT2CC0
RW	RW	RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8
RTCC7	RTCC6	RTCC5	RTCC4	RTCC3	RTCC2	RTCC1	RTCC0
RW	RW	RW	RW	RW	RW	RW	RW
7	6	5	4	3	2	1	0
STCC7	STCC6	STCC5	STCC4	STCC3	STCC2	STCC1	STCC0
RW	RW	RW	RW	RW	RW	RW	RW

Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.

Рисунок 13.4 – Условие перехода (смещение 0x00)

Таблица 13.6 – Описание битов условия перехода (смещение 0x00)

Бит	Обозначение	Тип	Описание
31	1	R	1 для идентификации перехода.
30 – 27	RSV	R	Зарезервировано. Читаются как 0.
26	IRQC	RW	Прерывание, если условие выполнено.
25	ACT	RW	Действие, что делать, если условие выполнено: - 0 – приостановить выполнение расписания; - 1 – переход.
24	MODE	RW	Логический режим: - 0 – режим ИЛИ (любой бит, установленный в RT2CC, RTCC устанавливается в RT2ST, RTST, результат ИЛИ в маске STCC); - 1 – режим И (все биты, установленные в RT2CC, RTCC устанавливаются в RT2ST, RTST, результат И в маске STCC).
23 – 16	RT2CC7 – RT2CC0	RW	УУ 2 код условия. Маскируется битами соответствующего RT2ST в слове результата последней передачи
15 – 8	RTCC7 – RTCC0	RW	УУ код условия. Маскируется битами соответствующего RTST в слове результата последней передачи.
7 – 0	STCC7 – STCC0	RW	Код состояния условия. Маскируется битами соответствующего значения состояния последней передачи.
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

Чтобы получить заведомо верное условие, необходимо установить MODE = 0 и STCC = FFh; для заведомо ложного условия – STCC = 00h. Таким образом, значение 80_0000FFh может быть использовано как маркер конца списка.

13.3 Управление удалённым устройством MIL-STD-1553B

Когда MIL-STD-1553B функционирует как удалённое устройство, оно является ведомым на шине (slave), считывая запросы, соответствующие его УУ адресу (или общие передачи). После проверки корректности их конфигурирования либо осуществляется соответствующая передача (при корректном запросе), либо, в противном случае, устанавливается соответствующий флаг ошибки в слове состояния. Корректность определяется управляющим словом подадреса для передачи данных, а для кодов режимов – соответствующим регистром.

Для запуска удалённого устройства необходимо: проинициализировать таблицу подадресов, кольцевой буфер, затем программно задать адрес УУ (для MIL-STD-1553_0 предусмотрена возможность задания адреса УУ с внешних выводов ИС во время сброса) и бит разрешения УУ в регистре конфигурации УУ.

Управление передачей данных

В режиме УУ используется трёхуровневая структура для управления данными. На верхнем уровне таблица подадресов, где каждый подадрес имеет управляющее слово, а также указатели на дескриптор приёма и передачи. Каждый дескриптор содержит слово управления/состояния, указатель на буфер данных, указатель на следующий дескриптор, формируя тем самым связанный список или кольцо дескрипторов. Буферы данных могут размещаться в любом месте памяти с 16-битным выравниванием. Когда УУ получает запрос передачи данных, оно по таблице подадресов определяет, корректный ли этот запрос. Если запрос корректный, то передача осуществляется в или из соответствующего буфера. После выполнения передачи слово состояния/управления дескриптора обновляется информацией об успешной или неуспешной передаче, указатель изменяется – указывает теперь на следующий дескриптор. Если запись отчётов обмена по шине разрешена, то соответствующая запись заносится в журнал. Также может быть разрешено прерывание передачи данных. Для определения передачи источника прерывания УУ необходимо обратиться к регистру отчётов источника прерывания. Поэтому для разрешения прерываний следует разрешить запись отчётов обмена по шине.

Если запрос корректный, но по некоторым причинам не может быть выполнен (например, не получается получить доступ к данным за отведённое время), MIL-STD-1553B посылает УУ сигнал прерывания ошибки доступа и не отвечает на запрос.

Рисунок 13.5 – Пример структуры управления данными УУ

Коды режимов

Регистр управления режимами УУ определяет, какие режимы разрешены, имеют запись отчётов и прерываются. Что касается передачи данных, то для разрешения прерываний следует разрешить запись отчётов обмена по шине. Режимы, которые могут быть предназначены сразу всем УУ, имеют два отдельных поля – для управления общим и отдельным вариантом режима. Различные коды режимов, а также соответствующие действия представлены в таблице 13.7. Некоторые режимы не имеют предопределённой функциональности, её при желании может реализовать программное обеспечение.

Журнал событий

Журнал событий представляет собой кольцо 32-битных записей, формат записей представлен в таблице 13.8. При передаче данных биты 23-0 записи журнала событий идентичны битам 23-0 слова состояния дескриптора.

Таблица 13.7 – Коды режимов

Код режима	Описание	Действие	Прерывание	Доступность после сброса	Биты в регистре
00000	Динамическое управление шиной	Если бит DBCA установлен в регистре состояния шины УУ, посылается ответ принятия динамического управления шиной	Да	Нет	17 – 16
00001	Синхронизация	Обновляется поле времени в регистре синхронизации УУ; сигнал rtsync устанавливается в 1 на один цикл AMBA	Да	Да	3 – 0
00010	Передать слово состояния	Передаёт слово состояния УУ; разрешено всегда, не имеет записи отчётов, не может быть запрещено	Нет	Да	–
00011	Начать самотестирование	Нет предопределённой функциональности	Да	Нет	21 – 18
00100	Завершение работы передатчика	УУ перестаёт отвечать на команды на шине (не на шине, по которой получена команда)	Да	Да	11 – 8
00101	Отменить завершение работы передатчика	Отменить эффект от ранее полученной команды на завершение работы передатчика	Да	Да	11 – 8
00110	Запрещение флага окончания	Маскирование флага окончания в слове состояния УУ	Да	Нет	25 – 22
00111	Отменить запрещение флага окончания	Отменить эффект от ранее полученной команды запрещения флага окончания	Да	Нет	25 – 22
01000	Перезагрузка УУ	Перезагрузка таймеров, флага завершения работы передатчика и флага окончания. Очистка битов флага окончания и запроса на обслуживание, УУ состояние шины. Сигнал extreset устанавливается в 1 на один цикл AMBA	Да	Нет	29 – 26
10000	Передать векторное слово	Передаёт векторное слово из регистра слов состояния УУ	Да	Нет	13 – 12
10001	Синхронизация со словом данных	Поля времени и данных в регистре синхронизации УУ обновляются. Сигнал rtsync устанавливается в 1 на один цикл AMBA	Да	Да	7 – 4
10010	Передать последнюю команду	Передаёт последнюю команду; разрешено всегда, не имеет записи отчётов, не может быть запрещено	Нет	Да	–
10011	Передать битовое слово	Отправляет битовое слово из регистра слова состояния УУ	Да	Нет	15 – 14
10100	Выборочное завершение работы передатчика	Нет предопределённой функциональности	Нет	Нет	–
10101	Отмена выборочного завершения работы передатчика	Нет предопределённой функциональности	Нет	Нет	–

31	30	29	28	27	26	25	24
IRQSR	TYPE1	TYPE0	SAMC4	SAMC3	SAMC2	SAMC1	SAMC0
R	R	R	R	R	R	R	R
23	22	21	20	19	18	17	16
TIMEL13	TIMEL12	TIMEL11	TIMEL10	TIMEL9	TIMEL8	TIMEL7	TIMEL6
R	R	R	R	R	R	R	R
15	14	13	12	11	10	9	8
TIMEL5	TIMEL4	TIMEL3	TIMEL2	TIMEL1	TIMEL0	BC	SZ5
R	R	R	R	R	R	R	R
7	6	5	4	3	2	1	0
SZ4	SZ3	SZ2	SZ1	SZ0	TRES2	TRES1	TRES0
R	R	R	R	R	R	R	R

Примечание – Принятое условное обозначение: R – только чтение.

Рисунок 13.6 – Журнал событий

Таблица 13.8 – Описание битов журнала событий

Бит	Обозначение	Тип	Описание
31	IRQSR	R	Источник прерывания, устанавливается в 1, если передача вызвала прерывание.
30 – 29	TYPE1, TYPE0	R	Тип передачи: - 00 – отправка данных; - 01 – приём данных; - 10 – код режима.
28 – 24	SAMC4 – SAMC0	R	Подадрес/Код режима; для типов передачи 00 или 01 (см. предыдущее поле) – подадрес, иначе – код режима.
23 – 10	TIMEL13 – TIMEL0	R	Младшие 14 бит счётчика времени.
9	BC	R	Признак общей передачи; устанавливается в 1, если запрос был адресован всем.
8 – 3	SZ5 – SZ0	R	Размер передачи: количество16-битных слов (0–32).
2 – 0	TRES2 – TRES0	R	Результат передачи: - 000 – успешная передача; - 001 – замена (отмена, так как новая команда была получена на другой шине); - 010 – ошибка ПДП, превышение времени доступа к памяти; - 011 – ошибка протокола; - 100 – в переданном слове состояния выставлен бит занятости или бит ошибки сообщения или данные не отправлены; - 101 – передача отменена из-за ошибки контрольной суммы обратной передачи.
Примечание – Принятое условное обозначение: R – только чтение.			

Формат подадресов представлен в таблице 13.9.

Таблица 13.9 – Формат записи для подадресов (0 < N < 31) УУ

Смещение	Значение	ПДП чтение/запись
0x10 × N + 0x00	Управляющее слово подадреса N	Чтение
0x10 × N + 0x01	Указатель на дескриптор отправки, 16-байтное выравнивание (0x3 – означает неверный указатель)	Чтение/Запись
0x10 × N + 0x02	Указатель на дескриптор приёма, 16-байтное выравнивание (0x3 – означает неверный указатель)	Чтение/Запись
0x10 × N + 0x03	Не используется	–
Примечание – Записи кода режима подадресов 0 и 31 не используются.		

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R	R	R	R	R	R	R	R
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	WRAP	IGNDV	BCRXEN
R	R	R	R	R	RW	RW	RW
15	14	13	12	11	10	9	8
RXEN	RXLOG	RXIRQ	RXSZ4	RXSZ3	RXSZ2	RXSZ1	RXSZ0
RW	RW	RW	RW	RW	RW	RW	RW
7	6	5	4	3	2	1	0
TXEN	TXLOG	TXIRQ	TXSZ4	TXSZ3	TXSZ2	TXSZ1	TXSZ0
RW	RW	RW	RW	RW	RW	RW	RW

Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.

Рисунок 13.7 – Управляющее слово подадреса УУ (смещение 0x10 × N + 0x00)

Таблица 13.10 – Описание битов управляющего слова подадреса УУ

Бит	Обозначение	Тип	Описание
31 – 19	RSV	R	Зарезервировано.
18	WRAP	RW	Разрешение автоматического циклического возврата: разрешает тестовый режим для данного подадреса, где передачи отправляют обратно полученные данные; это делается копированием указателя завершённого дескриптора передачи в поле указателя дескриптора передачи после каждой успешной передачи. Если WRAP = 1, то нельзя устанавливать TXSZ > RXSZ, так как это может вызвать чтение данных за границами буфера.
17	IGNDV	RW	Игнорирование бита корректности данных. Если этот бит установлен в 1, то полученные данные перепишут буфер вместо отсутствия ответа на запрос.
16	BCRXEN	RW	Разрешение общих передач: разрешает приём данным подадресом общих передач.
15	RXEN	RW	Разрешение приёма: разрешает приём данных этим подадресом.
14	RXLOG	RW	Запись отчёта приёма: запись приёма в журнал событий (только, если RXEN = 1).
13	RXIRQ	RW	Прерывание при приёме: каждый приём вызывает прерывание (только, если RXEN = RXLOG = 1).
12 – 8	RXSZ4 – RXSZ0	RW	Максимально допустимый размер принимаемых данных этим подадресом в 16-битных словах; 0 означает 32.
7	TXEN	RW	Разрешение передачи: разрешает передачу данных этим подадресом.
6	TXLOG	RW	Запись отчёта передач: запись передач в журнал событий (только, если TXEN = 1).
5	TXIRQ	RW	Прерывание при передаче. Каждая передача вызывает прерывание (только если TXEN = TXLOG = 1).
4 –0	TXSZ4 – TXSZ0	RW	Максимально допустимый размер передаваемых данных этим подадресом в 16-битных словах; 0 означает 32.
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

Таблица 13.11 – Формат УУ дескриптора

Смещение	Значение	ПДП Чтение/Запись
0x00	Слово управления и состояния, см. таблицу 13.13	Чтение/Запись
0x01	Указатель на буфер данных, 16 бит выравнивание	Чтение
0x02	Указатель на следующий дескриптор, 16 байт выравнивание или 0x0000003 обозначает конец списка	Чтение

31	30	29	28	27	26	25	24
DV	IRQEN	RSV	RSV	RSV	RSV	TIME15	TIME14
RW	RW	R	R	R	R	RW	RW
23	22	21	20	19	18	17	16
TIME13	TIME12	TIME11	TIME10	TIME9	TIME8	TIME7	TIME6
RW	RW	RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8
TIME5	TIME4	TIME3	TIME2	TIME1	TIME0	BC	SZ5
RW	RW	RW	RW	RW	RW	RW	RW
7	6	5	4	3	2	1	0
SZ4	SZ3	SZ2	SZ1	SZ0	TRES2	TRES1	TRES0
RW	RW	RW	RW	RW	RW	RW	RW

Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.

Рисунок 13.8 – Слово состояния/управления дескриптора УУ (смещение 0x00)

Таблица 13.12 – Описание битов слова состояния/управления дескриптора УУ

Бит	Обозначение	Тип	Описание
31	DV	RW	Готовность данных. До передачи необходимо программно установить в 0; после передачи аппаратура устанавливает в 1.
30	IRQEN	RW	Переопределение разрешения прерывания. Прерывание и запись в журнал после передачи независимо от бита SA управляющего слова. Может быть использован для прерывания по достижении конца списка дескрипторов.
29 – 26	RSV	R	Зарезервировано, следует маскировать нулём при чтении.
25 – 10	TIME15 – TIME0	RW	Тэг времени передачи. Устанавливается значение УУ таймера после передачи.
9	BC	RW	Общая передача; устанавливается, если передача для всех УУ.
8 – 3	SZ5 – SZ0	RW	Размер передачи; количество 16-разрядных слов.
2 – 0	TRES2 – TRES0	RW	Результат передачи: - 000 – успешная передача; - 001 – переопределено (отменено, так как получена новая команда на другой шине); - 010 – ошибка ПДП или превышено время доступа к памяти; - 011 – ошибка протокола; - 100 – в переданном слове состояния установлен бит занятости или бит ошибки сообщения, данные не получены; - 101 – передача отменена из-за ошибки проверки обратной передачи.
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

13.4 Управление монитором шины MIL-STD-1553B

Монитор шины (далее – МШ) может быть разрешён сам по себе, а также параллельно с КШ или УУ. Он работает как пассивное устройство, которое ведёт журнал событий на шине, записывая полученные данные с временным заголовком в кольцевой буфер.

Фильтрация

Монитор шины поддерживает фильтрацию.

Передачи могут фильтроваться по адресу УУ, подадресу, режиму, при этом условия фильтрации могут быть объединены по логическому И. Если все биты трёх регистров фильтрации и биты 2, 3 регистра управления установлены в 1, то МШ будет заносить в журнал все слова, полученные по шине.

Для фильтрации по подадресу/режиму у МШ есть возможность определять слова, принадлежащие к тому же сообщению. Поддерживаются все 10 типов сообщений. Если появится неожидаемое слово, то логика фильтрации будет перезапущена. Данные, не относящиеся к сообщению, могут быть записаны установкой бита в регистре управления.

Фильтр может быть перезапущен вручную установкой бита разрешения монитора сначала в 0, затем в 1.

Отсутствие передачи

В протоколе MIL-STD-1553B командное слово для задания режима, использующее индикатор 0, или передача для подадреса 8, имеет ту же структуру, что и слово состояния. Следовательно, возникает неоднозначность при использовании фильтров подадресов или режимов. Удалённое устройство не отвечает по подадресу, и КШ отправляет ту же команду повторно. Это может привести к тому, что второе слово управления будет интерпретировано, как слово состояния и использовано фильтром.

Контроллер шины может задействовать инструментарный бит или зарезервированные биты для устранения неоднозначности. Такой ситуации не возникнет в случае подадресов 1 – 7, 9 – 30 и кода режима 31. Также такого случая не возникнет, когда фильтры подадресов/режима не используются, а используется только адресный фильтр УУ.

Формат записи в журнал

Каждая запись в журнале представляет собой два 32-битных слова.

31	30	29	28	27	26	25	24
DV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R	R	R	R	R	R	R	R
23	22	21	20	19	18	17	16
TIME23	TIME22	TIME21	TIME20	TIME19	TIME18	TIME17	TIME16
RW	RW	RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8
TIME15	TIME14	TIME13	TIME12	TIME11	TIME10	TIME9	TIME8
RW	RW	RW	RW	RW	RW	RW	RW
7	6	5	4	3	2	1	0
TIME7	TIME6	TIME5	TIME4	TIME3	TIME2	TIME1	TIME0
RW	RW	RW	RW	RW	RW	RW	RW

Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.

Рисунок 13.9 – Слово 0 записи в журнал (смещение 0x00)

Таблица 13.13 – Описание битов слова 0 записи в журнал (смещение 0x00)

Бит	Обозначение	Тип	Описание
31	DV	R	Всегда читается 1.
30 – 24	RSV	R	Зарезервировано; следует маскировать нулём при чтении.
23 – 0	TIME23 – TIME0	RW	Тэг времени.
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

31	30	29	28	27	26	25	24
1	RSV	RSV	RSV	RSV	RSV	RSV	RSV
RW	R	R	R	R	R	R	R
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	BUS	WST1	WST0	WTP
R	R	R	R	RW	RW	RW	RW
15	14	13	12	11	10	9	8
WD15	WD14	WD13	WD12	WD11	WD10	WD9	WD8
RW	RW	RW	RW	RW	RW	RW	RW
7	6	5	4	3	2	1	0
WD7	WD6	WD5	WD4	WD3	WD2	WD1	WD0
RW	RW	RW	RW	RW	RW	RW	RW

Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.

Рисунок 13.10 – Слово 1 записи в журнал (смещение 0x01)

Таблица 13.14 – Описание битов слова 1 записи в журнал (смещение 0x01)

Бит	Обозначение	Тип	Описание
31	1	RW	Всегда пишется 1.
30 – 20	RSV	R	Зарезервировано; следует маскировать нулём при чтении.
19	BUS	RW	Шина, по которой получены данные: - 0 – A; - 1 – B.
18 – 17	WST1, WST0	RW	Слово состояния: - 00 – успешно; - 01 – ошибка Манчестера; - 10 – ошибка чётности; - 11 – xx.
16	WTP	RW	Тип слова: - 0 – данные; - 1 – команда/состояние.
15 – 0	WD15 – WD0	RW	Слово данных.
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

Тактирование

Кодек управляется в двух доменах тактирования: ЦПУ и 1553 кодек, с синхронизацией и взаимодействием между доменами. Задержка распространения сигнала до одного кодек цикла (50 нс) может быть допущена в каждом тактовом домене.

У MIL-STD-1553B есть два отдельных входа сброса для двух тактовых доменов. Сигнал сброса необходимо подавать одновременно на оба входа.

13.5 Регистры MIL-STD-1553B

Регистры MIL-STD-1553B конфигурируются и управляются через картированные в памяти регистры, без добавления новых инструкций в систему команд процессорных ядер ИС 1867ВН016. Каждый из блоков (КШ, УУ и МШ) имеет свой набор регистров, также есть несколько общих регистров.

Сводная информация по регистрам MIL-STD-1553B приведена в таблицах 13.15, 13.16, 13.17 и 13.18.

Таблица 13.15 – Регистры интерфейса MIL-STD-1553B

Адрес	Регистр	Описание регистра	Чтение/Запись	Значение после сброса
80_40F0h 80_41F0h 80_42F0h 80_43F0h	IRQ0 IRQ1 IRQ2 IRQ3	Регистры прерываний	Чтение/Запись для очистки записывается 1	0000_0000h
80_40F4h 80_41F4h 80_42F4h 80_43F4h	IRQE0 IRQE1 IRQE2 IRQE3	Регистры разрешения прерываний	Чтение/Запись	0000_0000h
80_40F5h – 80_40FFh 80_41F5h – 80_41FFh 80_42F5h – 80_42FFh 80_43F5h – 80_43FFh	RSV	Зарезервировано		
80_4100h 80_4200h 80_4300h 80_4400h	HCNFGR0 HCNFGR1 HCNFGR2 HCNFGR3	Регистры конфигу-рации аппаратного обеспечения	Чтение	0000_0000h
80_4104h 80_4204h 80_4304h 80_4404h	SELCLKMNCH0 SELCLKMNCH1 SELCLKMNCH2 SELCLKMNCH3	Регистры выбора источника тактирования	Чтение/Запись	0000_0000h
80_4105h – 80_4113h 80_41F5h – 80_41FFh 80_42F5h – 80_42FFh 80_43F5h – 80_43FFh	RSV	Зарезервировано		
80_4130h – 80_416Fh 80_4230h – 80_426Fh 80_4330h – 80_436Fh 80_4430h – 80_446Fh	Регистровые диапазоны КШ (см. таблицу 13.16)			
80_4170h – 80_41AFh 80_4270h – 80_42AFh 80_4370h – 80_43AFh 80_4470h – 80_44AFh	Регистровые диапазоны УУ (см. таблицу 13.17)			
80_41B0h – 80_57FFh 80_42B0h – 80_5FFFh 80_43B0h – 80_77FFh 80_44B0h – 80_7FFFh	Регистровые диапазоны МШ (см. таблицу 13.18)			

Таблица 13.16 – Регистры контроллера шины MIL-STD-1553B

Адрес	Регистр	Описание регистра	Чтение/Запись	Значение после сброса
80_4130h 80_4230h 80_4330h 80_4430h	BCST&CNFG0 BCST&CNFG1 BCST&CNFG2 BCST&CNFG3	Регистры состояния и конфигурирования КШ	Чтение/Запись	D000_0000h
80_4134h 80_4234h 80_4334h 80_4434h	BCACT0 BCACT1 BCACT2 BCACT3	Регистры действий КШ	Запись	0000_0000h
80_4138h 80_4238h 80_4338h 80_4438h	BCTRLNP0 BCTRLNP1 BCTRLNP2 BCTRLNP3	Регистры указателей списка передачи КШ	Чтение/Запись	0000_0000h
80_413Ch 80_423Ch 80_433Ch 80_443Ch	BCALNP0 BCALNP1 BCALNP2 BCALNP3	Регистры указателей асинхронного списка передачи КШ	Чтение/Запись	0000_0000h
80_4140h 80_4240h 80_4340h 80_4440h	BCTMR0 BCTMR1 BCTMR2 BCTMR3	Регистры таймера КШ	Чтение	0000_0000h
80_4141h – 80_4147h 80_4241h – 80_4247h 80_4341h – 80_4347h 80_4441h – 80_4447h	RSV	Зарезервировано		
80_4148h 80_4248h 80_4348h 80_4448h	BCTRIRQP0 BCTRIRQP1 BCTRIRQP2 BCTRIRQP3	Регистры позиции в кольце прерываний передачи данных КШ	Чтение/Запись	0000_0000h
80_414Ch 80_424Ch 80_434Ch 80_444Ch	BCRTBSW0 BCRTBSW1 BCRTBSW2 BCRTBSW3	Регистры замены шины для УУ	Чтение/Запись	0000_0000h
80_414Dh – 80_4157h 80_424Dh – 80_4257h 80_434Dh – 80_4357h 80_444Dh – 80_4457h	RSV	Зарезервировано		
80_4158h 80_4258h 80_4358h 80_4458h	BCTRLCRNTSLTP0 BCTRLCRNTSLTP1 BCTRLCRNTSLTP2 BCTRLCRNTSLTP3	Регистры указателей текущего слота в списке передач КШ	Чтение	0000_0000h
80_415Ch 80_425Ch 80_435Ch 80_445Ch	BCALCRNTSLTP0 BCALCRNTSLTP1 BCALCRNTSLTP2 BCALCRNTSLTP3	Регистры указателей текущего слота в асинхронном списке передач КШ	Чтение	0000_0000h
80_415Dh – 80_416Fh 80_425Dh – 80_426Fh 80_435Dh – 80_436Fh 80_445Dh – 80_446Fh	RSV	Зарезервировано		

Таблица 13.17 – Регистры удалённого устройства MIL-STD-1553B

Адрес	Регистр	Описание регистра	Чтение/Запись	Значение после сброса
80_4170h 80_4270h 80_4370h 80_4470h	RTSTR0 RTSTR1 RTSTR2 RTSTR3	Регистры состояния УУ	Чтение	8000_0000h
80_4174h 80_4274h 80_4374h 80_4474h	RTCNFGR0 RTCNFGR1 RTCNFGR2 RTCNFGR3	Регистры конфигурации УУ	Чтение/Запись	0000E03Eh
80_4178h 80_4278h 80_4378h 80_4478h	RTBSTR0 RTBSTR1 RTBSTR2 RTBSTR3	Регистры битов состояния шины УУ	Чтение/Запись	0000_0000h
80_417Ch 80_427Ch 80_437Ch 80_447Ch	RTSTWR0 RTSTWR1 RTSTWR2 RTSTWR3	Регистры слов состояния УУ	Чтение/Запись	0000_0000h
80_4180h 80_4280h 80_4380h 80_4480h	RTSYNR0 RTSYNR1 RTSYNR2 RTSYNR3	Регистры синхронизации УУ	Чтение	0000_0000h
80_4184h 80_4284h 80_4384h 80_4184h	RTSUBATBL0 RTSUBATBL1 RTSUBATBL2 RTSUBATBL3	Регистры базового адреса таблицы подадресов УУ	Чтение/Запись	0000_0000h
80_4188h 80_4288h 80_4388h 80_4488h	RTMDCNTRL0 RTMDCNTRL1 RTMDCNTRL2 RTMDCNTRL3	Регистры управления режимами УУ	Чтение/Запись	0000_0555h
80_4189h – 80_4193h 80_4289h – 80_4293h 80_4389h – 80_4393h 80_4489h – 80_4493h	RSV	Зарезервировано		
80_4194h 80_4294h 80_4394h 80_4494h	RTTIMECNTRLR0 RTTIMECNTRLR1 RTTIMECNTRLR2 RTTIMECNTRLR3	Регистры управления тегом времени УУ	Чтение/Запись	0000_0000h
80_4195h – 80_419Bh 80_4295h – 80_429Bh 80_4395h – 80_439Bh 80_4495h – 80_449Bh	RSV	Зарезервировано		
80_419Ch 80_429Ch 80_439Ch 80_449Ch	RTEVLOGMSK0 RTEVLOGMSK1 RTEVLOGMSK2 RTEVLOGMSK3	Регистры маски размера журнала событий УУ	Чтение/Запись	FFFF_FFFCh
80_41A0h 80_42A0h 80_43A0h 80_44A0h	RTEVLOGP0 RTEVLOGP1 RTEVLOGP2 RTEVLOGP3	Регистры позиции в журнале событий УУ	Чтение/Запись	0000_0000h
80_41A4h 80_42A4h 80_43A4h 80_44A4h	RTEVLOGINRP0 RTEVLOGINRP1 RTEVLOGINRP2 RTEVLOGINRP3	Регистры позиции прерываний в журнале событий УУ	Чтение	0000_0000h
80_41A5h – 80_41AFh 80_42A5h – 80_42AFh 80_43A5h – 80_43AFh 80_44A5h – 80_44AFh	RSV	Зарезервировано		

Таблица 13.18 – Регистры монитора шины MIL-STD-1553B

Адрес	Регистр	Описание регистра	Чтение/Запись	Значение после сброса
80_41B0h 80_42B0h 80_43B0h 80_44B0h	BMSTS0 BMSTS1 BMSTS2 BMSTS3	Регистры состояния МШ	Чтение	8000_0000h
80_41B4h 80_42B4h 80_43B4h 80_44B4h	BMCNTRL0 BMCNTRL1 BMCNTRL2 BMCNTRL3	Регистры управления МШ	Чтение/Запись	0000_0000h
80_41B8h 80_42B8h 80_43B8h 80_44B8h	BMRTAF0 BMRTAF1 BMRTAF2 BMRTAF3	Регистры фильтра адреса УУ	Чтение/Запись	FFFF_FFFFh
80_41BCh 80_42BCh 80_43BCh 80_44BCh	BMRTSUBAF0 BMRTSUBAF1 BMRTSUBAF2 BMRTSUBAF3	Регистры фильтра подадреса УУ	Чтение/Запись	FFFF_FFFFh
80_41C0h 80_42C0h 80_43C0h 80_44C0h	BMRTMF0 BMRTMF1 BMRTMF2 BMRTMF3	Регистры фильтра кода режима УУ	Чтение/Запись	FFFF_FFFFh
80_41C4h 80_42C4h 80_43C4h 80_41C4h	BMLOGBST0 BMLOGBST1 BMLOGBST2 BMLOGBST3	Регистры адреса начала журнала МШ	Чтение/Запись	0000_0000h
80_41C8h 80_42C8h 80_43C8h 80_44C8h	BMLOGBE0 BMLOGBE1 BMLOGBE2 BMLOGBE3	Регистры адреса конца журнала МШ	Чтение/Запись	0000_0007h
80_41CCh 80_42CCh 80_43CCh 80_44CCh	BMLOGBP0 BMLOGBP1 BMLOGBP2 BMLOGBP3	Регистры позиции в журнале МШ	Чтение/Запись	0000_0000h
80_41D0h 80_42D0h 80_43D0h 80_44D0h	BMTIMECNTRL0 BMTIMECNTRL1 BMTIMECNTRL2 BMTIMECNTRL3	Регистры управле-ния тегом времени МШ	Чтение/Запись	0000_0000h
80_41D1h – 80_41EFh 80_42D1h – 80_42EFh 80_43D1h – 80_43EFh 80_44D1h – 80_44EFh	RSV	Зарезервировано		
80_5000h – 80_57FFh 80_5800h – 80_5FFFh 80_7000h – 80_77FFh 80_7800h – 80_7FFFh	MNCH_RAM0 MNCH_RAM1 MNCH_RAM2 MNCH_RAM3	ОЗУ объёмом 2K × 32 бит	Чтение/Запись	–

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	BMTOF	BMD
R-0	R-0	R-0	R-0	R-0	R-0	RW-0	RW-0
15	14	13	12	11	10	9	8
RSV	RSV	RSV	RSV	RSV	RTTE	RTD	RTEV
R-0	R-0	R-0	R-0	R-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
RSV	RSV	RSV	RSV	RSV	RSV	BCD	BCEV
R-0	R-0	R-0	R-0	R-0	R-0	RW-0	RW-0

Примечание – Принятые условные обозначения: R – чтение, W – запись; биты читаются как 1, если прерывание возникло, для сброса следует писать 1.

Здесь и далее в названиях рисунков /адрес регистра/ (***n*** = 0–3) – номер блока MIL-STD-1553B.

Например:
адрес регистра IRQ3 для MIL-STD-1553B_**3** – 80_4(***n***)F0h = 80_4**3**F0h;
адрес регистра HCNFGR3 для MIL-STD-1553B_**3** – 80_4(***n+1***)F0h = 80_4**4**00h.

Рисунок 13.11 – Структура регистра прерываний IRQ***n*** /адрес **80_4(*n*)F0**h/

Таблица 13.19 – Описание битов регистра прерываний MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31 – 18	RSV	R	Зарезервировано
17	BMTOF	RW	Переполнение таймера МШ
16	BMD	RW	Ошибка ПДП МШ
15 – 11	RSV	R	Зарезервировано.
10	RTTE	RW	Ошибка доступа к таблице УУ
9	RTD	RW	Ошибка ПДП УУ
8	RTEV	RW	Прерывание при передаче данных УУ
7 – 2	RSV	R	Зарезервировано
1	BCD	RW	Ошибка ПДП КШ
0	BCEV	RW	Прерывание при передаче данных КШ
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	BMTOE	BMDE
R-0	R-0	R-0	R-0	R-0	R-0	RW-0	RW-0
15	14	13	12	11	10	9	8
RSV	RSV	RSV	RSV	RSV	RTTEE	RTDE	RTEVE
R-0	R-0	R-0	R-0	R-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
RSV	RSV	RSV	RSV	RSV	RSV	BCDE	BCEVE
R-0	R-0	R-0	R-0	R-0	R-0	RW-0	RW-0

Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.

Рисунок 13.12 – Структура регистра прерываний IRQE***n*** /адрес **80_4(*n*)F4**h/

Таблица 13.20 – Описание битов регистра разрешения прерываний MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31 – 18	RSV	R	Зарезервировано.
17	BMTOE	RW	Прерывание по переполнению таймера МШ разрешено.
16	BMDE	RW	Прерывание по ошибке ПДП МШ разрешено.
15 – 11	RSV	R	Зарезервировано.
10	RTTEE	RW	Прерывание по ошибке доступа к таблице УУ разрешено.
9	RTDE	RW	Прерывание по ошибке ПДП УУ разрешено.
8	RTEVE	RW	Прерывание при передаче данных УУ разрешено.
7 – 2	RSV	R	Зарезервировано.
1	BCDE	RW	Прерывание по ошибке ПДП КШ разрешено.
0	BCEVE	RW	Прерывание при передаче данных КШ разрешено.
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
RSV	RSV	RSV	RSV	RSV	ENDIAN1	ENDIAN0	SLCK
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
CCFREQ7	CCFREQ6	CCFREQ5	CCFREQ4	CCFREQ3	CCFREQ2	CCFREQ1	CCFREQ0
R-0	R-0	R-0	R-1	R-1	R-0	R-0	R-0

Примечание – Принятое условное обозначение: R – только чтение.

Рисунок 13.13 – Структура регистра аппаратной конфигурации HCNFGR***n*** /адрес **80_4(*n*+*1*)00**h/

Таблица 13.21 – Описание битов регистра аппаратной конфигурации MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31 – 11	RSV	R	Зарезервировано.
10 – 9	ENDIAN1, ENDIAN0	R	Реализованная комбинация 00 определяет AHB порядок байт, в котором первый байт – старший.
8	SLCK	R	0 означает возможность работы приёмопередатчика и процессора в разных доменах.
7 – 0	CCFREQ7 – CCFREQ0	R	Значение поля 18h соответствует частоте 24 МГц 1553 кодека.
Примечание – Принятое условное обозначение: R – только чтение.			

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
RSV	RSV	RSV	RSV	RSV	RSV	RSV	MNCHCLK
R-0	R-0	R-0	R-0	R-0	R-0	R-0	RW-0

Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.

Рисунок 13.14 – Структура регистра выбора источника тактирования SELCLKMNCH***n*** /адрес **80_4(*n*+*1*)04**h/

Таблица 13.22 – Описание битов регистра выбора источника тактирования MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31 – 1	RSV	R	Зарезервировано.
0	MNCHCLK	RW	Тактирование MIL-STD-1553B (24 МГц): - MNCHCLK = 0 – тактирование от CPUCLK - MNCHCLK = 1 – тактирование от внешнего вывода MIL-STD-1553B.
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

31	30	29	28	27	26	25	24
BCSUP	BCFEAT2	BCFEAT1	BCFEAT0	RSV	RSV	RSV	RSV
R-1	R-1	R-0	R-1	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	BCCHK
R-0	R-0	R-0	R-0	R-0	R-0	R-0	RW-0
15	14	13	12	11	10	9	8
ASADL4	ASADL3	ASADL2	ASADL1	ASADL0	0	ASST1	ASST0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
SCADL4	SCADL3	SCADL2	SCADL1	SCADL0	SCST2	SCST1	SCST0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.

Рисунок 13.15 – Структура регистра состояния и управления КШ BCST&CNFG***n*** /адрес **80_4(*n*+*1*)30**h/

Таблица 13.23 – Описание битов регистра состояния и управления КШ MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31	BCSUP	R	Поддержка режима КШ. Читается как 1.
30 – 28	BCFEAT2 – BCFEAT0	R	Особенности КШ (двоичное значение поля – 101). Битовое поле, указывающее реализованную конфигурацию КШ (1 – поддерживается): - 30 – таймер расписания КШ поддерживается; - 29 – таймер пробуждения КШ не поддерживается; - 28 – регистр переключения шины УУ и бита дескриптора STBUS поддерживается.
27 – 17	RSV	R	Зарезервировано.
16	BCCHK	RW	Проверка общих передач. Записываемый бит, установка в 1, разрешает ожидание и проверку для неожидаемых ответов на все общие запросы.
15 – 11	ASADL4 – ASADL0	R	Младшие биты адреса асинхронного списка. Биты 8 – 4 адреса исполняемой (если ASST = 01) или следующей команды.
10	0	R	Читается как 0, запись игнорируется.
9 – 8	ASST1, ASST0	R	Состояние асинхронного списка: - 00 – остановлен; - 01 – выполнение команды; - 10 – ожидание слота времени; - 11 – xx.
7 – 3	SCADL4 – SCADL0	R	Младшие биты адреса расписания. Биты 8 – 4 адреса исполняемой (если SCST = 01) или следующей команды.
2 – 0	SCST2 – SCST0	R	Состояние расписания: - 000 – остановлено; - 001 – выполнение команды; - 010 – ожидание слота времени; - 011 – отложено; - 100 – ожидание внешней синхронизации; - (101 – 111) – xxx.
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

31	30	29	28	27	26	25	24
BCKEY15	BCKEY14	BCKEY13	BCKEY12	BCKEY11	BCKEY10	BCKEY9	BCKEY8
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
BCKEY7	BCKEY6	BCKEY5	BCKEY4	BCKEY3	BCKEY2	BCKEY1	BCKEY0
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
RSV	RSV	RSV	RSV	RSV	RSV	ASSTP	ASSTR
R	R	R	R	R	R	W	W
7	6	5	4	3	2	1	0
RSV	RSV	RSV	CLRT	SETT	SCSTP	SCSUS	SCSRT
R	R	R	W	W	W	W	W

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

Рисунок 13.16 – Структура регистра действий КШ ВСАСТ***n*** /адрес **80_4(*n*+*1*)34**h/

Таблица 13.24 – Описание битов регистра действий КШ MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31 – 16	BCKEY15 – BCKEY0	W	Код безопасности; во время записи должен быть 0x1552, иначе запись в регистр игнорируется.
15 – 10	RSV	R	Зарезервировано.
9	ASSTP	W	Остановка асинхронного списка. Запись 1 останавливает асинхронный список команд после текущей передачи.
8	ASSTR	W	Запуск асинхронного списка. Запись 1 запускает асинхронный список.
7 – 5	RSV	R	Зарезервировано.
4	CLRT	W	Очистка внешнего триггера. Запись 1 очищает память триггера.
3	SETT	W	Установка внешнего триггера. Запись 1 устанавливает триггер.
2	SCSTP	W	Остановка расписания. Запись 1 останавливает расписание после текущей передачи.
1	SCSUS	W	Приостановка расписания. Запись 1 приостанавливает расписание после текущей передачи.
0	SCSRT	W	Запуск расписания. Запись 1 запускает расписание.
Примечание – Принятые условные обозначения: R – только чтение, W – только запись.			

31	30	29	28	27	26	25	24
STLP31	STLP30	STLP29	STLP28	STLP27	STLP26	STLP25	STLP24
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
23	22	21	20	19	18	17	16
STLP23	STLP22	STLP21	STLP20	STLP19	STLP18	STLP17	STLP16
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
15	14	13	12	11	10	9	8
STLP15	STLP14	STLP13	STLP12	STLP11	STLP10	STLP9	STLP8
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
STLP7	STLP6	STLP5	STLP4	STLP3	STLP2	STLP1	STLP0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Примечание – Принятое условное обозначение: RW – чтение и запись.

Рисунок 13.17 – Указатель следующего адреса списка передач регулярного расписания КШ BCTRLNP***n*** /адрес **80_4(*n*+*1*)38**h/

Таблица 13.25 – Описание битов указателя следующего адреса списка передач регулярного расписания КШ MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31 – 0	STLP31 – STLP0	RW	Чтение: указатель адреса исполняемой (текущей, если SCST = 001) передачи или следующей, которая будет выполняться в регулярном расписании. Запись: изменение адреса; если расписание запущено, то будет выполнен переход после завершения текущей передачи.
Примечание – Принятое условное обозначение: RW – чтение и запись.			

31	30	29	28	27	26	25	24
ASLP31	ASLP30	ASLP29	ASLP28	ASLP27	ASLP26	ASLP25	ASLP24
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
23	22	21	20	19	18	17	16
ASLP23	ASLP22	ASLP21	ASLP20	ASLP19	ASLP18	ASLP17	ASLP16
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
15	14	13	12	11	10	9	8
ASLP15	ASLP14	ASLP13	ASLP12	ASLP11	ASLP10	ASLP9	ASLP8
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
ASLP7	ASLP6	ASLP5	ASLP4	ASLP3	ASLP2	ASLP1	ASLP0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Примечание – Принятое условное обозначение: RW – чтение и запись.

Рисунок 13.18 – Указатель следующего адреса асинхронного списка передач КШ BCALNP***n*** /адрес **80_4(*n*+*1*)3C**h/

Таблица 13.26 – Описание битов указателя следующего адреса асинхронного списка передач КШ MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31 – 0	ASLP31 – ASLP0	RW	Чтение: указатель адреса исполняемой (текущей, если ASST = 01) передачи или следующей, которая будет выполняться в асинхронном расписании. Запись: изменение адреса; если расписание запущено, то будет выполнен переход после завершения текущей передачи.
Примечание – Принятое условное обозначение: RW – чтение и запись.			

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
SCTM23	SCTM22	SCTM21	SCTM20	SCTM19	SCTM18	SCTM17	SCTM16
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
SCTM15	SCTM14	SCTM13	SCTM12	SCTM11	SCTM10	SCTM9	SCTM8
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
SCTM7	SCTM6	SCTM5	SCTM4	SCTM3	SCTM2	SCTM1	SCTM0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Примечание – Принятое условное обозначение: R – только чтение.

Рисунок 13.19 – Структура регистра таймера КШ BCTMR***n*** /адрес **80_4(*n*+*1*)40**h/

Таблица 13.27 – Описание битов регистра таймера КШ MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31 – 24	RSV	R	Зарезервировано.
23 – 0	SCTM23 – SCTM0	R	Оставшееся время в микросекундах (только для чтения); устанавливается в 0 при остановке расписания и при внешней синхронизации.
Примечание – Принятое условное обозначение: R – только чтение.			

31	30	29	28	27	26	25	24
SPRP25	SPRP24	SPRP23	SPRP22	SPRP21	SPRP20	SPRP19	SPRP18
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
23	22	21	20	19	18	17	16
SPRP17	SPRP16	SPRP15	SPRP14	SPRP13	SPRP12	SPRP11	SPRP10
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
15	14	13	12	11	10	9	8
SPRP9	SPRP8	SPRP7	SPRP6	SPRP5	SPRP4	SPRP3	SPRP2
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
SPRP1	SPRP0	RSV	RSV	RSV	RSV	0	0
RW-0	RW-0	R-0	R-0	R-0	R-0	R-0	R-0

Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.

Рисунок 13.20 – Структура регистра позиции в кольце прерываний передачи BCTRIRQP***n*** /адрес **80_4(*n*+*1*)48**h/

Таблица 13.28 – Описание битов регистра позиции в кольце прерываний передачи КШ MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31 – 6	SPRP25 – SPRP0	RW	Текущий указатель записи в кольцо прерываний передач. Кольцо выравнивается по 64-байтной границе; пользователь может поменять только биты 31 – 6.
5 – 2	RSV	R	Зарезервировано.
1 – 0	0	R	Биты 1 – 0 всегда нулевые (4-байтное выравнивание).
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

31	30	29	28	27	26	25	24
BS31	BS30	BS29	BS28	BS27	BS26	BS25	BS24
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
23	22	21	20	19	18	17	16
BS23	BS22	BS21	BS20	BS19	BS18	BS17	BS16
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
15	14	13	12	11	10	9	8
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Примечание – Принятое условное обозначение: RW – чтение и запись.

Рисунок 13.21 – Структура регистра замены шины УУ КШ BCRTBSW***n*** /адрес **80_4(*n*+*1*)4C**h/

Таблица 13.29 – Описание битов регистра замены шины УУ КШ MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31 – 0	BS31 – BS0	RW	Номер выбранной шины будет результатом операции логического «исключающего ИЛИ» с битом в маске соответствующего УУ (получающего УУ при УУ–УУ передаче). Этот регистр обновляется MIL-STD-1553B, если используется бит дескриптора STBUS.
Примечание – Принятое условное обозначение: RW – чтение и запись.			

31	30	29	28	27	26	25	24
CRNT27	CRNT26	CRNT25	CRNT24	CRNT23	CRNT22	CRNT21	CRNT20
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
CRNT19	CRNT18	CRNT17	CRNT16	CRNT15	CRNT14	CRNT13	CRNT12
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
CRNT11	CRNT10	CRNT9	CRNT8	CRNT7	CRNT6	CRNT5	CRNT4
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
CRNT3	CRNT2	CRNT1	CRNT0	0	0	0	0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Примечание – Принятое условное обозначение: R – только чтение.

Рисунок 13.22 – Структура указателя дескриптора текущего слота времени КШ BCTRLCRNTSLTP***n*** /адрес **80_4(*n*+*1*)58**h/

Таблица 13.30 – Описание битов указателя дескриптора текущего слота времени КШ MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31 – 4	CRNT27 – CRNT0	R	Указывает на дескриптор передачи, соответствующий текущему слоту времени (только для чтения, актуальное значение только при выполнении расписания).
3 – 0	0	R	Биты 3 – 0 всегда нулевые (128 бит/16-байтное выравнивание).
Примечание – Принятое условное обозначение: R – только чтение.			

31	30	29	28	27	26	25	24
ACRNT27	ACRNT26	ACRNT25	ACRNT24	ACRNT23	ACRNT22	ACRNT21	ACRNT20
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
ACRNT19	ACRNT18	ACRNT17	ACRNT16	ACRNT15	ACRNT14	ACRNT13	ACRNT12
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
ACRNT11	ACRNT10	ACRNT9	ACRNT8	ACRNT7	ACRNT6	ACRNT5	ACRNT4
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
ACRNT3	ACRNT2	ACRNT1	ACRNT0	0	0	0	0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Примечание – Принятое условное обозначение: R – только чтение.

Рисунок 13.23 – Структура указателя дескриптора асинхронного списка текущего слота времени КШ BCALCRNTSLTP***n*** /адрес **80_4(*n*+*1*)5C**h/

Таблица 13.31 – Описание битов указателя дескриптора асинхронного списка текущего слота времени КШ MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31 – 4	ACRNT27 – ACRNT0	R	Указывает на дескриптор передачи, соответствующий текущему слоту времени асинхронного списка (только для чтения, актуальное значение только при выполнении асинхронного списка).
3 – 0	0	R	Биты 3–0 всегда нулевые (128 бит/16-байтное выравнивание).
Примечание – Принятое условное обозначение: R – только чтение.			

31	30	29	28	27	26	25	24
RTSUP	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-1	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
RSV	RSV	RSV	RSV	ACT	SHDA	SHDB	RUN
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Примечание – Принятое условное обозначение: R – только чтение.

Рисунок 13.24 – Структура регистра состояния УУ (только для чтения) RTSTR***n*** / адрес **80_4(*n*+*1*)70**h/

Таблица 13.32 – Описание битов регистра состояния УУ (только для чтения) MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31	RTSUP	R	Поддержка УУ. Читается как 1.
30 – 4	RSV	R	Зарезервировано.
3	ACT	R	УУ активно; читается как 1, если УУ в данный момент выполняет передачу данных.
2	SHDA	R	Завершение работы шины А; читается как 1, если КШ завершил работу шины А, используя команду.
1	SHDB	R	Завершение работы шины В; читается как 1, если КШ завершил работу шины В, используя команду.
0	RUN	R	УУ запущено; читается как 1, если УУ слушает команды.
Примечание – Принятое условное обозначение: R – только чтение.			

31	30	29	28	27	26	25	24
RTKEY15	RTKEY14	RTKEY13	RTKEY12	RTKEY11	RTKEY10	RTKEY9	RTKEY8
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
23	22	21	20	19	18	17	16
RTKEY7	RTKEY6	RTKEY5	RTKEY4	RTKEY3	RTKEY2	RTKEY1	RTKEY0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
15	14	13	12	11	10	9	8
SYS	SYDS	BRS	RSV	RSV	RSV	RSV	RSV
RW-1	RW-1	RW-1	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
RSV	RTEIS	RTADDR4	RTADDR3	RTADDR2	RTADDR1	RTADDR0	RTEN
R-0	RW-0	RW-1	RW-1	RW-1	RW-1	RW-1	RW-0

Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.

Рисунок 13.25 – Структура регистра конфигурации УУ RTCNFGR***n*** /адрес **80_4(*n*+*1*)74**h/

Таблица 13.33 – Описание битов регистра конфигурации УУ MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31 – 16	RTKEY15 – RTKEY0	RW	Код безопасности. Используется для подтверждения изменения адреса УУ путём записи значения 1553h; читается всегда нулями.
15	SYS	RW	Разрешение сигнала синхронизации; при приёме кода режима синхронизации (без данных) необходимо установить в 1 для активации сигнала rtsync.
14	SYDS	RW	Разрешение сигнала синхронизации с данными; при получении кода режима синхронизации (со словом данных) необходимо установить в 1 для активации сигнала rtsync.
13	BRS	RW	Разрешение сигнала перезагрузки шины; при приёме кода режима сброса УУ необходимо установить в 1 для активации сигнала busreset.
12 – 7	RSV	R	Зарезервировано.
6	RTEIS	RW	В MIL-STD-1553_0 читается как 1, если текущий адрес был установлен через внешние входы; после установки адреса из программы, этот бит ставится в ноль. В остальных MIL-STD-1553_ – всегда 0.
5 – 1	RTADDR4 – RTADDR0	RW	УУ адрес: адрес этого УУ (0 – 30); в любой момент адрес УУ может быть изменён программно. Для MIL-STD-1553_0 предусмотрена возможность задания адреса УУ аппаратно, путём опроса состояния соответствующих внешних входов ИС в момент сброса. При условии корректности уровня на входе PA_RT0 (бит паритета), в это поле во время сброса переписывается значение адреса, выставленное на входах A*n*_RT0. В остальное время (при неактивном сбросе) никакие изменения уровней на этих входах микросхемы не приводят к изменению текущего адреса УУ MIL-STD-1553_0.
0	RTEN	RW	Разрешение УУ; для разрешения УУ установить в 1.
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
RSV	RSV	RSV	RSV	RSV	RSV	RSV	TFDE
R-0	R-0	R-0	R-0	R-0	R-0	R-0	RW-0
7	6	5	4	3	2	1	0
RSV	RSV	RSV	SREQ	BUSY	SSF	DBCA	TFLG
R-0	R-0	R-0	RW-0	RW-0	RW-0	RW-0	RW-0

Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.

Рисунок 13.26 – Структура регистра состояния шины УУ RTBSTR***n***
/адрес **80_4(*n*+*1*)78**h/

Таблица 13.34 – Описание битов регистра состояния шины УУ MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31 – 9	RSV	R	Зарезервировано.
8	TFDE	RW	Автоматическая установка флага терминала при ошибках ПДП и таблицы дескрипторов.
7 – 5	RSV	R	Зарезервировано.
4 – 0 – эти биты будут высланы в случае запроса состояния УУ по 1553 шине:			
4	SREQ	RW	Запрос обслуживания
3	BUSY	RW	Бит занятости; если УУ ответит только словом состояния, передача считается неудачной.
2	SSF	RW	Флаг подсистемы.
1	DBCA	RW	Динамическое управление шиной; бит высылается только по соответствующему запросу.
0	TFLG	RW	Флаг терминала; КШ может замаскировать этот флаг соответствующей командой.
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

31	30	29	28	27	26	25	24
BITW15	BITW14	BITW13	BITW12	BITW11	BITW10	BITW9	BITW8
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
23	22	21	20	19	18	17	16
BITW7	BITW6	BITW5	BITW4	BITW3	BITW2	BITW1	BITW0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
15	14	13	12	11	10	9	8
VECW15	VECW14	VECW13	VECW12	VECW11	VECW10	VECW9	VECW8
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
VECW7	VECW6	VECW5	VECW4	VECW3	VECW2	VECW1	VECW0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Примечание – Принятое условное обозначение: RW – чтение и запись.

Рисунок 13.27 – Структура регистра слов состояния УУ RTSTWR***n***
/адрес **80_4(*n*+*1*)7C**h/

Таблица 13.35 – Описание битов регистра слов состояния УУ MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31 – 16	BITW15 – BITW0	RW	Слово, которое передаётся КШ по команде передачи битового слова, если разрешено в регистре управления кодами режимов УУ.
15 – 0	VECW15 – VECW0	RW	Слово, которое передаётся КШ по команде передачи векторного слова, если разрешено в регистре управления кодами режимов УУ.
Примечание – Принятое условное обозначение: RW – чтение и запись.			

31	30	29	28	27	26	25	24
SYTM15	SYTM14	SYTM13	SYTM12	SYTM11	SYTM10	SYTM9	SYTM8
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
SYTM7	SYTM6	SYTM5	SYTM4	SYTM3	SYTM2	SYTM1	SYTM0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
SYD15	SYD14	SYD13	SYD12	SYD11	SYD10	SYD9	SYD8
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
SYD7	SYD6	SYD5	SYD4	SYD3	SYD2	SYD1	SYD0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Примечание – Принятое условное обозначение: R – только чтение.

Рисунок 13.28 – Структура регистра синхронизации УУ RTSYNR***n***
/адрес **80_4(*n*+*1*)80**h/

Таблица 13.36 – Описание битов регистра синхронизации УУ MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31 – 16	SYTM15 – SYTM0	R	Значение таймера УУ при последней команде синхронизации или синхронизации с данными, если разрешено в регистре управления кодами режимов УУ.
15 – 0	SYD15 – SYD0	R	Данные, полученные с последней командой синхронизации с данными, если разрешено в регистре управления кодами режимов УУ.
Примечание – Принятое условное обозначение: R – только чтение.			

31	30	29	28	27	26	25	24
SATB23	SATB22	SATB21	SATB20	SATB19	SATB18	SATB17	SATB16
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
23	22	21	20	19	18	17	16
SATB15	SATB14	SATB13	SATB12	SATB11	SATB10	SATB9	SATB8
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
15	14	13	12	11	10	9	8
SATB7	SATB6	SATB5	SATB4	SATB3	SATB2	SATB1	SATB0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.

Рисунок 13.29 – Структура регистра базового адреса таблицы подадресов УУ RTSUBATBL***n*** /адрес **80_4(*n*+*1*)84**h/

Таблица 13.37 – Описание битов регистра базового адреса таблицы подадресов УУ MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31 – 8	SATB23 – SATB0	RW	Базовый адрес; биты 31 – 9 для таблицы подадресов.
7 – 0	0	R	Читаются как 0, запись игнорируется.
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

31	30	29	28	27	26	25	24
RSV	RSV	RRTB1	RRTB0	RRT1	RRT0	ITFB1	ITFB0
R-0	R-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
23	22	21	20	19	18	17	16
ITF1	ITF0	ISTB1	ISTB0	IST1	IST0	DBC1	DBC0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
15	14	13	12	11	10	9	8
TBW1	TBW0	TVW1	TVW0	TSB1	TSB0	TS1	TS0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-1	RW-0	RW-1
7	6	5	4	3	2	1	0
SDB1	SDB0	SD1	SD0	SB1	SB0	S1	S0
RW-0	RW-1	RW-0	RW-1	RW-0	RW-1	RW-0	RW-1

Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись; для каждого кода: 00 – запрещён, 01 – разрешён, 10 – разрешён с журналированием, 11 – разрешён с журналированием и прерыванием.

Рисунок 13.30 – Структура регистра управления кодами режимов УУ RTMDCNTRL***n*** /адрес **80_4(*n*+*1*)88**h/

Таблица 13.38 – Описание битов регистра управления кодами режимов УУ MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31 – 30	RSV	R	Зарезервировано
29 – 28	RRTB1, RRTB0	RW	Перезапуск всех УУ
27 – 26	RRT1, RRT0	RW	Перезапуск УУ
25 – 24	ITFB1, ITFB0	RW	Запрещение/перезапись всех флагов запрещения терминала
23 – 22	ITF1, ITF0	RW	Запрещение/перезапись флага запрещения терминала
21 – 20	ISTB1, ISTB0	RW	Всем начать самопроверку
19 – 18	IST1, IST0	RW	Начать самопроверку
17 – 16	DBC1, DBC0	RW	Динамическое управление шиной
15 – 14	TBW1, TBW0	RW	Передать битовое слово
13 – 12	TVW1, TVW0	RW	Передать векторное слово
11 – 10	TSB1, TSB0	RW	Завершение работы всех передатчиков и перезапись
9 – 8	TS1, TS0	RW	Завершение работы передатчика и перезапись
7 – 6	SDB1, SDB0	RW	Синхронизация со словом данных для всех
5 – 4	SD1, SD0	RW	Синхронизация со словом данных
3 – 2	SB1, SB0	RW	Синхронизация для всех
1 – 0	S1, S0	RW	Синхронизация

Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись; для каждой пары бит кодов режимов:

- 00 – запрещено;
- 01 – разрешено;
- 10 – разрешено, разрешён журнал;
- 11 – разрешено, журнал и прерывание.

31	30	29	28	27	26	25	24
TRES15	TRES14	TRES13	TRES12	TRES11	TRES10	TRES9	TRES8
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
23	22	21	20	19	18	17	16
TRES7	TRES6	TRES5	TRES4	TRES3	TRES2	TRES1	TRES0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
15	14	13	12	11	10	9	8
TVAL15	TVAL14	TVAL13	TVAL12	TVAL11	TVAL10	TVAL9	TVAL8
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
TVAL7	TVAL6	TVAL5	TVAL4	TVAL3	TVAL2	TVAL1	TVAL0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Примечание – Принятое условное обозначение: RW – чтение и запись.

Рисунок 13.31 – Структура регистра управления тегом времени УУ RTTIMECNTRLR***n*** /адрес **80_4(*n*+*1*)94**h/

Таблица 13.39 – Описание битов регистра управления тегом времени УУ MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31 – 16	TRES15 – TRES0	RW	Разрешение (дискретность) тега времени; единица времени счётчика тега УУ в микросекундах минус один.
15 – 0	TVAL15 – TVAL0	RW	Значение тега времени; текущее значение запущенного счётчика тега времени.
Примечание – Принятое условное обозначение: RW – чтение и запись.			

31	30	29	28	27	26	25	24
1	1	1	1	1	1	1	1
RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1
23	22	21	20	19	18	17	16
1	1	1	ELSM18	ELSM17	ELSM16	ELSM15	ELSM14
RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1
15	14	13	12	11	10	9	8
ELSM13	ELSM12	ELSM11	ELSM10	ELSM9	ELSM8	ELSM7	ELSM6
RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1
7	6	5	4	3	2	1	0
ELSM5	ELSM4	ELSM3	ELSM2	ELSM1	ELSM0	0	0
RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	RW-0	RW-0

Примечание – Принятое условное обозначение: RW – чтение и запись.

Рисунок 13.32 – Структура регистра маски журнала событий УУ RTEVLOGMSK***n*** /адрес **80_4(*n*+*1*)9C**h/

Таблица 13.40 – Описание битов регистра маски журнала событий УУ

Бит	Обозначение	Тип	Описание
31 – 21	1	RW	Должны быть установлены в 1.
20 – 2	ELSM18 – ELSM0	RW	Маскирует определённый размер и выравнивает УУ буфер журнала.
1 – 0	0	RW	Должны быть установлены в 0.
Примечание – Принятое условное обозначение: RW – чтение и запись.			

31	30	29	28	27	26	25	24
ELWP31	ELWP30	ELWP29	ELWP28	ELWP27	ELWP26	ELWP25	ELWP24
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
23	22	21	20	19	18	17	16
ELWP23	ELWP22	ELWP21	ELWP20	ELWP19	ELWP18	ELWP17	ELWP16
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
15	14	13	12	11	10	9	8
ELWP15	ELWP14	ELWP13	ELWP12	ELWP11	ELWP10	ELWP9	ELWP8
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
ELWP7	ELWP6	ELWP5	ELWP4	ELWP3	ELWP2	ELWP1	ELWP0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Примечание – Принятое условное обозначение: RW – чтение и запись.

Рисунок 13.33 – Структура регистра позиции в журнале событий УУ RTEVLOGP***n*** / адрес **80_4(*n*+*1*)A0**h/

Таблица 13.41 – Описание битов регистра позиции в журнале событий УУ

Бит	Обозначение	Тип	Описание
31 – 0	ELWP31 – ELWP0	RW	Адрес первой неиспользуемой/старой записи в буфер журнала событий, 32-битное выравнивание.
Примечание – Принятое условное обозначение: RW – чтение и запись.			

31	30	29	28	27	26	25	24
ELIP31	ELIP30	ELIP29	ELIP28	ELIP27	ELIP26	ELIP25	ELIP24
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
ELIP23	ELIP22	ELIP21	ELIP20	ELIP19	ELIP18	ELIP17	ELIP16
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
ELIP15	ELIP14	ELIP13	ELIP12	ELIP11	ELIP10	ELIP9	ELIP8
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
ELIP7	ELIP6	ELIP5	ELIP4	ELIP3	ELIP2	ELIP1	ELIP0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Примечание – Принятое условное обозначение: R – только чтение.

Рисунок 13.34 – Структура регистра позиции в журнале прерываний УУ RTEVLOGINRP***n*** /адрес **80_4(*n*+*1*)A4**h/

Таблица 13.42 – Описание битов регистра позиции в журнале прерываний УУ

Бит	Обозначение	Тип	Описание
31 – 0	ELIP31 – ELIP0	R	Адрес записи в журнал, соответствующей прерыванию, 32-битное выравнивание. Регистр устанавливается для первого прерывания, не устанавливается снова до распознавания прерывания.
Примечание – Принятое условное обозначение: R – только чтение.			

31	30	29	28	27	26	25	24
BMSUP	KEYEN	RSV	RSV	RSV	RSV	RSV	RSV
R-1	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Примечание – Принятое условное обозначение: R – только чтение.

Рисунок 13.35 – Структура регистра состояния МШ BMSTS***n*** /адрес **80_4(*n*+*1*)B0**h/

Таблица 13.43 – Описание битов регистра состояния МШ MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31	BMSUP	R	Поддержка монитора шины. Читается как 1.
30	KEYEN	R	Ключ безопасности для МШ не используется. Читается как 0.
29 – 0	RSV	R	Зарезервировано.
Примечание – Принятое условное обозначение: R – только чтение.			

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
RSV	RSV	WRSTP	EXST	IMCL	UDWL	MANL	BMEN
R-0	R-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.

Рисунок 13.36 – Структура регистра управления МШ BMCNTRL***n*** /адрес **80_4(*n*+*1*)B4**h/

Таблица 13.44 – Описание битов регистра управления МШ MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31 – 6	RSV	R	Зарезервировано.
5	WRSTP	RW	Останов по заполнению буфера; если установить в 1, то BMEN установится в ноль, когда буфер монитора будет заполнен.
4	EXST	RW	Старт по сигналу внешней синхронизации; если установить в 1, то BMEN также установится в 1, МШ запустится по получении сигнала внешней синхронизации.
3	IMCL	RW	Запись в отчёт неверного кода режима; разрешается при установке в 1.
2	UDWL	RW	Запись в отчёт неожидаемых данных; разрешается при установке в 1.
1	MANL	RW	Запись в отчёт ошибок Манчестера/чётность; разрешается при установке в 1.
0	BMEN	RW	Разрешение МШ; должен быть в 1 для записи в отчёт событий.
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

31	30	29	28
ADDRFM31	ADDRFM30	ADDRFM29	ADDRFM28
RW-1	RW-1	RW-1	RW-1
27	26	25	24
ADDRFM27	ADDRFM26	ADDRFM25	ADDRFM24
RW-1	RW-1	RW-1	RW-1
23	22	21	20
ADDRFM23	ADDRFM22	ADDRFM21	ADDRFM20
RW-1	RW-1	RW-1	RW-1
19	18	17	16
ADDRFM19	ADDRFM18	ADDRFM17	ADDRFM16
RW-1	RW-1	RW-1	RW-1
15	14	13	12
ADDRFM15	ADDRFM14	ADDRFM13	ADDRFM12
RW-1	RW-1	RW-1	RW-1
11	10	9	8
ADDRFM11	ADDRFM10	ADDRFM9	ADDRFM8
RW-1	RW-1	RW-1	RW-1
7	6	5	4
ADDRFM7	ADDRFM6	ADDRFM5	ADDRFM4
RW-1	RW-1	RW-1	RW-1
3	2	1	0
ADDRFM3	ADDRFM2	ADDRFM1	ADDRFM0
RW-1	RW-1	RW-1	RW-1

Примечание – Принятое условное обозначение: RW – чтение и запись.

Рисунок 13.37 – Структура регистра фильтра адресов УУ МШ BMRTAF***n*** /адрес **80_4(*n*+*1*)B8**h/

Таблица 13.45 – Описание битов регистра фильтра адресов УУ МШ MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31	ADDRFM31	RW	Разрешение записи в отчёт общих передач.
30 – 0	ADDRFM30 – ADDRFM0	RW	Каждый бит, установленный в 1, разрешает запись в отчёт передач, связанных с адресом как номер бита.
Примечание – Принятое условное обозначение: RW – чтение и запись.			

31	30	29	28
SADDRFM31	SADDRFM30	SADDRFM29	SADDRFM28
RW-1	RW-1	RW-1	RW-1
27	26	25	24
SADDRFM27	SADDRFM26	SADDRFM25	SADDRFM24
RW-1	RW-1	RW-1	RW-1
23	22	21	20
SADDRFM23	SADDRFM22	SADDRFM21	SADDRFM20
RW-1	RW-1	RW-1	RW-1
19	18	17	16
SADDRFM19	SADDRFM18	SADDRFM17	SADDRFM16
RW-1	RW-1	RW-1	RW-1
15	14	13	12
SADDRFM15	SADDRFM14	SADDRFM13	SADDRFM12
RW-1	RW-1	RW-1	RW-1
11	10	9	8
SADDRFM11	SADDRFM10	SADDRFM9	SADDRFM8
RW-1	RW-1	RW-1	RW-1
7	6	5	4
SADDRFM7	SADDRFM6	SADDRFM5	SADDRFM4
RW-1	RW-1	RW-1	RW-1
3	2	1	0
SADDRFM3	SADDRFM2	SADDRFM1	SADDRFM0
RW-1	RW-1	RW-1	RW-1

Примечание – Принятое условное обозначение: RW – чтение и запись.

Рисунок 13.38 – Структура регистра фильтра подадресов УУ МШ BMRTSUBAF***n*** /адрес **80_4(*n*+*1*)BC**h/

Таблица 13.46 – Описание битов регистра фильтра подадресов УУ МШ MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31	SADDRFM31	RW	Разрешение записи в отчёт команд с подадресом 31.
30 – 1	SADDRFM30 – SADDRFM1	RW	Каждый бит, установленный в 1, разрешает запись в отчёт передач, связанных с подадресом как номер бита.
0	SADDRFM0	RW	Разрешение записи в отчёт команд с подадресом 0.
Примечание – Принятое условное обозначение: RW – чтение и запись.			

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-1	R-1	R-1	R-1	R-1	R-1	R-1	R-1
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	STSB	STS	TLC
R-1	R-1	R-1	R-1	R-1	RW-1	RW-1	RW-1
15	14	13	12	11	10	9	8
TSW	RRTB	RRT	ITFB	ITF	ISTB	IST	DBC
RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1
7	6	5	4	3	2	1	0
TBW	TVW	TSB	TS	SDB	SD	SB	S
RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1	RW-1

Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись; каждый бит, установленный в 1, разрешает журналирование режима с соответствующим кодом.

Рисунок 13.39 – Структура регистра фильтра кодов режимов УУ МШ BMRTMF***n*** /адрес **80_4(*n*+*1*)C0**h/

Таблица 13.47 – Описание битов регистра фильтра кодов режимов УУ МШ MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31 – 19	RSV	R	Зарезервировано
18	STSB	RW	Выбор завершения работы передатчиков и переопределение общего завершения работы передатчиков
17	STS	RW	Выбор завершения работы передатчика и переопределение завершения работы передатчика
16	TLC	RW	Передать последнюю команду
15	TSW	RW	Передать слово состояния
14	RRTB	RW	Перезапуск всех УУ
13	RRT	RW	Перезапуск УУ
12	ITFB	RW	Запрет и переопределение всех битов флагов запрещения терминалов
11	ITF	RW	Запрет и переопределение бита флага запрещения терминала
10	ISTB	RW	Запуск самопроверки для всех
9	IST	RW	Запуск самопроверки
8	DBC	RW	Динамическое управление шиной
7	TBW	RW	Передать битовое слово
6	TVW	RW	Передать векторное слово
5	TSB	RW	Завершение работы всех передатчиков и переопределение завершения работы всех передатчиков
4	TS	RW	Завершение работы передатчика и переопределение завершения работы передатчика
3	SDB	RW	Синхронизация со словом данных для всех
2	SD	RW	Синхронизация со словом данных
1	SB	RW	Синхронизация для всех
0	S	RW	Синхронизация
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

31	30	29	28	27	26	25	24
LBS28	LBS27	LBS26	LBS25	LBS24	LBS23	LBS22	LBS21
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
23	22	21	20	19	18	17	16
LBS20	LBS19	LBS18	LBS17	LBS16	LBS15	LBS14	LBS13
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
15	14	13	12	11	10	9	8
LBS12	LBS11	LBS10	LBS9	LBS8	LBS7	LBS6	LBS5
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
LBS4	LBS3	LBS2	LBS1	LBS0	0	0	0
RW-0	RW-0	RW-0	RW-0	RW-0	R-0	R-0	R-0

Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.

Рисунок 13.40 – Структура регистра начала буфера журнала МШ BMLOGBST***n*** /адрес **80_4(*n*+*1*)C4**h/

Таблица 13.48 – Описание битов регистра начала буфера журнала МШ MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31 – 3	LBS28 – LBS0	RW	Указатель на начало журнала, 8-байтное выравнивание.
2 – 0	0	R	Из-за выравнивания биты 2 – 0 всегда 0.
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

31	30	29	28	27	26	25	24
SADDRB9	SADDRB8	SADDRB7	SADDRB6	SADDRB5	SADDRB4	SADDRB3	SADDRB2
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
23	22	21	20	19	18	17	16
SADDRB1	SADDRB0	LBE18	LBE17	LBE16	LBE15	LBE14	LBE13
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
15	14	13	12	11	10	9	8
LBE12	LBE11	LBE10	LBE9	LBE8	LBE7	LBE6	LBE5
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
LBE4	LBE3	LBE2	LBE1	LBE0	1	1	1
RW-0	RW-0	RW-0	RW-0	RW-0	R-1	R-1	R-1

Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.

Рисунок 13.41 – Структура регистра конца буфера журнала МШ BMLOGBE***n*** /адрес **80_4(*n*+*1*)C8**h/

Таблица 13.49 – Описание битов регистра конца буфера журнала МШ MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31 – 22	SADDRB9 – SADDRB0	RW	Значения разрядов 31 – 22 совпадают с таковыми в регистре начала буфера журнала BMLOGBST (LBS28 – LBS19).
21 – 3	LBE18 – LBE0	RW	Указатель на конец буфера журнала; только эти биты (21 – 3) доступны для записи; т. о. буфер не может выходить за пределы 4 Мбайт границы.
2 – 0	1	R	Из-за выравнивания биты 2 – 0 всегда 1.
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

31	30	29	28	27	26	25	24
SADDRB9	SADDRB8	SADDRB7	SADDRB6	SADDRB5	SADDRB4	SADDRB3	SADDRB2
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
23	22	21	20	19	18	17	16
SADDRB1	SADDRB0	LBP18	LBP17	LBP16	LBP15	LBP4	LBP13
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
15	14	13	12	11	10	9	8
LBP12	LBP11	LBP10	LBP9	LBP8	LBP7	LBP6	LBP5
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
LBP4	LBP3	LBP2	LBP1	LBP0	0	0	0
RW-0	RW-0	RW-0	RW-0	RW-0	R-0	R-0	R-0

Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.

Рисунок 13.42 – Структура регистра позиции буфера журнала МШ BMLOGBP***n*** /адрес **80_4(*n*+*1*)CC**h/

Таблица 13.50 – Описание битов регистра позиции буфера журнала МШ MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31 – 22	SADDRB9 – SADDRB0	RW	Значения разрядов 31 – 22 совпадают с таковыми в регистре начала буфера журнала BMLOGBST (LBS28 – LBS19).
21 – 3	LBP18 – LBP0	RW	Указатель на следующую позицию, куда будет производиться запись в буфер монитора; только эти биты (21 – 3) доступны для записи; т. о. буфер не может выходить за пределы 4 Мбайт границы.
2 – 0	0	R	Из-за выравнивания биты 2 – 0 всегда 0.
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

31	30	29	28	27	26	25	24
TRES7	TRES6	TRES5	TRES4	TRES3	TRES2	TRES1	TRES0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
23	22	21	20	19	18	17	16
TVAL23	TVAL22	TVAL21	TVAL20	TVAL19	TVAL18	TVAL17	TVAL16
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
15	14	13	12	11	10	9	8
TVAL15	TVAL14	TVAL13	TVAL12	TVAL11	TVAL10	TVAL9	TVAL8
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
TVAL7	TVAL6	TVAL5	TVAL4	TVAL3	TVAL2	TVAL1	TVAL0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Примечание – Принятое условное обозначение: RW – чтение и запись.

Рисунок 13.43 – Структура регистра управления тегом времени МШ BMTIMECNTRL***n*** /адрес **80_4(*n*+*1*)D0**h/

Таблица 13.51 – Описание битов регистра управления тегом времени МШ MIL-STD-1553B

Бит	Обозначение	Тип	Описание
31 – 24	TRES7 – TRES0	RW	Разрешение (дискретность) тега времени; единица времени счётчика тега МШ в микросекундах минус один.
23 – 0	TVAL23 – TVAL0	RW	Текущее значение запущенного счётчика времени.
Примечание – Принятое условное обозначение: RW – чтение и запись.			

14 Периферийное устройство UART

Периферийное устройство UART, Universal Asynchronous Receiver/Transmitter – универсальный асинхронный приёмник/передатчик, устройство, обеспечивающее последовательный обмен данными для организации взаимодействия с модемом или другими внешними устройствами с использованием протоколов RS-232 и RS-485. Блок UART спроектирован в соответствии с промышленным стандартом National Semiconductor`s 16550A device.

Частота входного кварцевого генератора выбирается равной 18 432 000 Гц. Для получения других скоростей приёма/передачи требуется подбирать коэффициенты N и M (в соответствии с требованиями к PLL), чтобы обеспечить тактирование последовательного канала UART и рассчитать коэффициент делителя UART. Тактовая частота последовательного канала UART (входная частота PLL) должна быть в диапазоне (65,536 – 100) МГц. В таблице 14.1 приведены значения скорости передачи по последовательному каналу при внешнем тактировании от входа UART_CLK/X2.

Таблица 14.1 – Скорость передачи при внешнем тактировании от вывода UART_CLK/X2

Скорость передачи, бит/с	UART_CLK/X2 = 1 843 200 Гц		UART_CLK/X2 = 3 072 000 Гц		UART_CLK/X2 = 18 432 000 Гц	
	Значение делителя	Ошибка, %	Значение делителя	Ошибка, %	Значение делителя	Ошибка, %
50	2304	–	3840	–	23040	–
75	1536	–	2560	–	15360	–
110	1047	0,026	1745	0,026	10473	–
134,5	857	0,058	1428	0,034	8565	–
150	768	–	1280	–	7680	–
300	384	–	640	–	3840	–
600	192	–	320	–	1920	–
1200	96	–	160	–	920	–
1800	64	–	107	0,312	640	–
2000	58	0,690	96	–	576	–
2400	48	–	80	–	480	–
3600	32	–	53	0,628	320	–
4800	24	–	40	–	240	–
7200	16	–	27	1,230	160	–
9600	12	–	20	–	120	–
19200	6	–	10	–	60	–
38400	3	–	5	–	30	–
56000	2	2,860	–	–	21	2,04
128000	–	–	–	–	9	–

14.1 Описание регистров UART

Модуль UART имеет архитектуру устройства 16550. В таблице 14.2 приведён список регистров периферийного устройства UART.

Таблица 14.2 – Регистры модулей UART_***n*** (***n*** = 0, 1)

Адрес	Доступ	Название	Функциональное назначение
UART_0			
80_4020h	R	RB0	Выход FIFO приёмника 0
80_4020h	W	THR0	Вход FIFO передатчика 0
80_4020h	RW	DLB10	Младший байт регистра делителя 0
80_4021h	RW	IER0	Регистр разрешения (маски) прерываний 0
80_4021h	RW	DLB20	Старший байт регистра делителя 0
80_4022h	R	II0	Регистр идентификации прерываний 0
80_4022h	W	FC0	Регистр управления FIFO 0
80_4023h	RW	LCR0	Регистр управления линией 0
80_4024h	W	MCR0	Регистр управления модемом 0
80_4025h	R	LSR0	Регистр состояния линии 0
80_4026h	R	MSR0	Регистр статуса модема 0
80_4028h	R	DBG10	Первый отладочный регистр 0
80_402Ch	R	DBG20	Второй отладочный регистр 0
80_4033h	RW	SELCLKUART0	Регистр выбора источника тактирования 0
80_4034h – 80_403Fh		RSV	Зарезервировано
UART_1			
80_4040h	R	RB1	Выход FIFO приёмника 1
80_4040h	W	THR1	Вход FIFO передатчика 1
80_4040h	RW	DLB11	Младший байт регистра делителя 1
80_4041h	RW	IER1	Регистр разрешения (маски) прерываний 1
80_4041h	RW	DLB21	Старший байт регистра делителя 1
80_4042h	R	II1	Регистр идентификации прерываний 1
80_4042h	W	FC1	Регистр управления FIFO 1
80_4043h	RW	LCR1	Регистр управления линией 1
80_4044h	W	MCR1	Регистр управления модемом 1
80_4045h	R	LSR1	Регистр состояния линии 1
80_4046h	R	MSR1	Регистр статуса модема 1
80_4048h	R	DBG11	Первый отладочный регистр 1
80_404Ch	R	DBG21	Второй отладочный регистр 1
80_4053h	RW	SELCLKUART1	Регистр выбора источника тактирования 1
80_4054h – 80_405Fh		RSV	Зарезервировано

Регистр разрешения прерываний UART (IER) служит для управления запретом / разрешением генерации прерываний от этого модуля.

На рисунке 14.1 приведена структура, а в таблице 14.3 – описание разрядов этого регистра.

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
RSV	RSV	RSV	RSV	MOD	RSUR	FIFOEMP	DATRSV
R-0	R-0	R-0	R-0	RW-0	RW-0	RW-0	RW-0

Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.

Рисунок 14.1 – Структура регистра разрешения прерываний IER***n***
/адрес **80_4021**h – UART_0; адрес **80_4041**h – UART_1/

Таблица 14.3 – Описание битов регистра разрешения прерываний IER

Бит	Обозначение	Тип	Описание
31 – 4	RSV	R	Зарезервировано.
3	MOD	RW	Прерывание от модема: - 0 – запрещено; - 1 – разрешено.
2	RSUR	RW	Прерывание от приёмника: - 0 – запрещено; - 1 – разрешено.
1	FIFOEMP	RW	Прерывание в случае, когда FIFO передатчика пуст: - 0 – запрещено; - 1 – разрешено.
0	DATRSV	RW	Прерывание при приёме данных: - 0 – запрещено; - 1 – разрешено.
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

Регистр идентификации прерывания II позволяет определить текущий приоритет прерывания. Бит 0 индицирует, что прерывание обрабатывается, когда бит равен 0. Когда этот бит равен 1, то нет необработанных прерываний. На рисунке 14.2 приведена структура регистра идентификации прерывания UART. Таблица 14.4 показывает список возможных прерываний с их приоритетом, источником и управления их сбросом.

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
1	1	0	0	D3	D2	D1	D0
R-1	R-1	R-0	R-0	RW-0	RW-0	RW-0	RW-1

Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.

Рисунок 14.2 – Структура регистра идентификации прерывания II***n***
/адрес **80_4022**h – UART_0; адрес **80_4042**h – UART_1/

Таблица 14.4 – Список возможных прерываний

Бит 3	Бит 2	Бит 1	Приоритет	Тип прерывания	Источник прерывания	Управление сбросом прерывания
0	1	1	1	От приёмника	Ошибки паритета, переполнение данных или формата, или обрыв линии	Чтение регистра Line Status
0	1	0	2	Есть принятые данные	FIFO заполнен	Чтение FIFO ниже уровня заполнения
1	1	0	2	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	3	Регистр передатчика пуст	Регистр передатчика пуст	Запись в регистр передатчика или чтение IIR
0	0	0	4	Статус модема	CTS, DSR, RI или DCD	Чтение регистра статуса модема

Регистр FIFO (FC) позволяет выбрать количество байт в FIFO, требуемых для разрешения прерывания по приёму данных. В дополнение FIFO может сбрасываться, используя этот регистр. На рисунке 14.3 приведена структура регистра управления FIFO FC, а разряды этого регистра описаны в таблице 14.5.

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
TRL1	TRL0	RSV	RSV	RSV	FTRESET	FRRESET	RSV
W-0	W-0	R-0	R-0	R-0	W-0	W-0	R-0

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

Рисунок 14.3 – Структура регистра управления FIFO FC
/адрес **80_4022**h – UART_0; адрес **80_4042**h – UART_1/

Таблица 14.5 – Описание битов регистра FC

Бит	Обозначение	Тип	Описание
31 – 8	RSV	R	Зарезервировано.
7 – 6	TRL1, TRL0	W	Определяет уровень прерывания FIFO приёмника: - 00 – 1 байт; - 01 – 4 байта; - 10 – 8 байт; - 11 – 14 байт.
5 – 3	RSV	R	Зарезервировано.
2	FTRESET	W	Запись 1 в этот бит очищает FIFO передатчика, но не сбрасывает сдвиговый регистр. Передача текущего символа продолжается.
1	FRRESET	W	Запись в этот бит 1 очищает FIFO приёмника, но не сбрасывает сдвиговый регистр. Приём текущего символа продолжается.
0	RSV	R	Зарезервировано.
Примечание – Принятые условные обозначения: R – только чтение, W – только запись.			

Регистр управления линией (LCR) позволяет задать формат асинхронных данных. 7 бит регистра позволяют получить доступ к защёлкам делителя, которые определяют скорость передачи. Чтение этого регистра позволяет проверить текущее состояние соединения.

На рисунке 14.4 приведена структура регистра управления линией LCR, а в таблице 14.6 – описание разрядов этого регистра.

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
ENDL	BREAK	PAR	ODD EVEN	ENPAR	STBIT	TRL1	TRL0
RW-0	RW-0	RW-0	RW-1	RW-1	RW-0	RW-1	RW-1

Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.

Рисунок 14.4 – Структура регистра управления линией LCR
/адрес **80_4023**h – UART_0; адрес **80_4043**h – UART_1/

Таблица 14.6 – Описание битов регистра LCR

Бит	Обозначение	Тип	Описание
31 – 8	RSV	R	Зарезервировано.
7	ENDL	RW	Доступ к делителю: - 1 – защёлки делителя могут адресоваться; - 0 – нормальный доступ к регистрам.
6	BREAK	RW	Управляющий бит обрыв: - 0 – обрыв запрещён; - 1 – последовательный выход устанавливается в 0.
5	PAR	RW	Добавка бита паритета: - 0 – добавка бита паритета запрещена. - 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.
4	ODD EVEN	RW	Чётный паритет: - 0 – нечётное число 1; - 1 – чётное число единиц.
3	ENPAR	RW	Разрешение паритета: - 0 – нет паритета; - 1 – бит паритета формируется и проверяется на каждом входящем символе.
2	STBIT	RW	Определяет число стоповых бит: - 0 – 1 стоповый бит; - 1 – 1,5 бита; - 1 – 2 стоповых бита, когда выбирается длина символа 5 бит.
1 – 0	TRL1, TRL0	RW	Выбор числа битов в каждом символе: - 00 – 5 бит; - 01 – 6 бит; - 10 – 7 бит; - 11 – 8 бит.
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

Регистр управления модемом MCR позволяет передавать управляющие сигналы на модем, подключенный к UART. На рисунке 14.5 приведена структура регистра MCR, а в таблице 14.7 – описание разрядов этого регистра.

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
RSV	RSV	RSV	LOOP	RSV	RSV	RTS	RSV
R-0	R-0	R-0	W-0	W-0	W-0	W-0	W-0

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

Рисунок 14.5 – Структура регистра управления модемом MCR
/адрес **80_4024**h – UART_0; адрес **80_4044**h – UART_1/

Таблица 14.7 – Описание битов регистра MCR

Бит	Обозначение	Тип	Описание
31 – 5	RSV	R	Зарезервировано.
4	LOOP	W	Режим «петли»: - 0 – нормальный режим; - 1 – режим «петли». Выход Tx_(1,0) устанавливается в 1. Выход сдвигового регистра передатчика внутренним образом коммутируется ко входу регистра сдвига приёмника.
3 – 2	RSV	R	Зарезервировано.
1	RTS	W	Сигнал запроса на передачу данных (RTS): - 0 – Выход RTS_(1,0)/EnTr_(1,0) устанавливается в 1; - 1 – Выход RTS_(1,0)/EnTr_(1,0) сбрасывается в 0.
0	RSV	R	Зарезервировано.
Примечание – Принятые условные обозначения: R – только чтение, W – только запись.			

Регистр состояния линии отражает текущее состояние линии приёма/передачи. На рисунке 14.6 приведена структура регистра LSR, а в таблице 14.8 – описание разрядов этого регистра.

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
ERROR	TE	FTE	BI	FE	PE	OE	DR
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Примечание – Принятое условное обозначение: R – только чтение.

Рисунок 14.6 – Структура регистра состояния линии LSR
/адрес **80_4025**h – UART_0; адрес **80_4045**h – UART_1/

Таблица 14.8 – Описание битов регистра LSR

Бит	Обозначение	Тип	Описание
1	2	3	4
31 – 8	RSV	R	Зарезервировано.
7	ERROR	R	Ошибка: - 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.
6	TE	R	Передатчик пустой: - 1 – FIFO и сдвиговый регистр передатчика пусты; Сбрасывается, когда данные записываются в FIFO. - 0 – либо FIFO, либо сдвиговый регистр имеют символ.
5	FTE	R	FIFO передатчика пустой: - 0 – FIFO не пуст; - 1 – FIFO передатчика пуст, генерируется прерывание «Регистр передатчика пуст». Бит сбрасывается записью символа в FIFO.
4	BI	R	Индикатор обрыва: - 0 – нет обрыва в текущем символе; - 1 – обрыв произошёл на текущем символе. Сигнал «Обрыв» происходит, 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 ожидает стартовый бит для приёма следующего символа. Бит сбрасывается чтением из этого регистра. Генерируется прерывание Receiver Line Status.
3	FE	R	Индикатор ошибки кадра: - 0 – нет ошибки кадра в текущем символе; - 1 –символ, который находится в вершине стека, принят с ошибкой кадра (неправильный стоповый бит). Бит сбрасывается чтением из регистра. Генерирует Receiver Line Status прерывание.

Окончание таблицы 14.8

1	2	3	4
2	PE	R	Индикатор ошибки паритета: - 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.
1	OE	R	Индикатор ошибки переполнения: - 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.
0	DR	R	Индикатор готовности данных: - 0 – нет символа в FIFO; - 1 – по крайней мере, один символ принят и находится в FIFO.
Примечание – Принятое условное обозначение: R – только чтение.			

Регистр статуса модема (MSR) отображает текущее состояние управляющих линий модема. Четыре бита индицируют состояние статусных линий модема: устанавливаются в единицу, когда обнаруживается изменение в соответствующей линии и сбрасываются при чтении регистра. На рисунке 14.7 приведена структура регистра MSR, а в таблице 14.9 – описание разрядов этого регистра.

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
ADVDCD	ADVRI	ADVDSR	ADVCTS	DDCD	TERI	DDSR	DCTS
RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0	RC-0

Примечание – Принятые условные обозначения: R – чтение. C – бит сбрасывается.

Рисунок 14.7 – Структура регистра статуса модема MSR
/адрес **80_4026**h – UART_0; адрес **80_4046**h – UART_1/

Таблица 14.9 – Описание битов регистра MSR

Бит	Обозначение	Тип	Описание
31 – 8	RSV	R	Зарезервировано.
7	AVDVCD	RC	Дополнение DCD или равный Out2 в режиме «петли».
6	ADVRI	RC	Дополнение RI или равный Out1 в режиме «петли».
5	ADVDSR	RC	Дополнение входа DSR или равный DTR в режиме «петли».
4	ADVCTS	RC	Дополнение входа CTS или равный RTS в режиме «петли».
3	DDCD	RC	Индикатор несущей Delta Data Carrier Detect: 1 – линия DCD изменила своё состояние.
2	TERI	RC	Детектор Trailing Edge of Ring Indicator detector. Линия RI изменила своё состояние из низкого состояния в высокое.
1	DDSR	RC	Индикатор Delta Data Set Ready: 1 – линия DSR изменила своё состояние.
0	DCTS	RC	Индикатор Delta Clear To Send: 1 – линия CTS изменяет своё состояние.
Примечание – Принятые условные обозначения: R – только чтение, RC – чтение и сброс.			

Регистр делителя DL. К защёлкам делителя разрешён доступ установкой в 1 бита 7 регистра LCR (см. таблицу 14.6). Необходимо восстановить значение этого бита в 0 после установки защёлок делителя для восстановления 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). На рисунках 14.8 и 14.9 приведён формат регистра DL.

31	30	29	28	27	26	25	24
D31	D30	D29	D28	D27	D26	D25	D24
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
23	22	21	20	19	18	17	16
D23	D22	D21	D20	D19	D18	D17	D16
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
15	14	13	12	11	10	9	8
D15	D14	D13	D12	D11	D10	D9	D8
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
D7	D6	D5	D4	D3	D2	D1	D0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.

Рисунок 14.8 – Структура регистра делителя DLB1
/адрес **80_4020**h – UART_0; адрес **80_4040**h – UART_1/

31	30	29	28	27	26	25	24
D63	D62	D61	D60	D59	D58	D57	D56
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
23	22	21	20	19	18	17	16
D55	D54	D53	D52	D51	D50	D49	D48
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
15	14	13	12	11	10	9	8
D47	D46	D45	D44	D43	D42	D41	D40
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0
7	6	5	4	3	2	1	0
D39	D38	D37	D36	D35	D34	D33	D32
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.

Рисунок 14.9 – Структура регистра делителя DLB2
/адрес **80_4021**h – UART_0; адрес **80_4041**h – UART_1/

Регистры отладки 1 (DBG1) и 2 (DBG2) используются для отладочных целей и не являются частью спецификации UART 16550. На рисунках 14.10 и 14.11 приведены структуры регистров DBG1 и DBG2, а в таблицах 14.10 и 14.11 – описание разрядов этих регистров соответственно.

31	30	29	28	27	26	25	24
ADVDCD	ADVRI	ADVDSR	ADVCTS	DDCD	TERI	DDSR	DCTS
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
ENDL	BREAK	PAR	ODD EVEN	ENPAR	STBIT	TRL1	TRL0
R-0	R-0	R-0	R-1	R-1	R-0	R-0	R-1
15	14	13	12	11	10	9	8
D3	D2	D1	D0	MOD	RSUR	FIFOEMP	DATRSV
R-0	R-0	R-0	R-1	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
ERROR	TE	FTE	BI	FE	PE	OE	DR
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Примечание – Принятое условное обозначение: R – только чтение.

Рисунок 14.10 – Структура регистра отладки 1 DBG1
/адрес **80_4028**h – UART_0; адрес **80_4048**h – UART_1/

Таблица 14.10 – Описание битов регистра DBG1

Бит	Обозначение	Тип	Описание
31 – 24	MSR	R	Значение регистра статуса модема; см. рисунок 14.7 и таблицу 14.9
23 – 16	LCR	R	Значение регистр линий управления; см. рисунок 14.4 и см. таблицу 14.6
15 – 12	II	R	Значение регистра идентификации прерывания (bits 3 – 0); см. рисунок 14.2
11 – 8	IER	R	Значение регистра разрешения прерываний (bits 3 – 0) см. рисунок 14.1 и таблицу 14.3
7 – 0	LSR	R	Значение линий статусного регистра; см. рисунок 14.6 и таблицу 14.8
Примечание – Принятое условное обозначение: R – только чтение.			

31	30	29	28
RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0
27	26	25	24
RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0
23	22	21	20
FCR7	FCR6	MCR4	MCR3
R-0	R-0	R-0	R-0
19	18	17	16
MCR2	MCR1	MCR0	RFCOUNT4
R-0	R-0	R-0	R-0
15	14	13	12
RFCOUNT3	RFCOUNT2	RFCOUNT1	RFCOUNT0
R-0	R-0	R-0	R-0
11	10	9	8
RSTATE3	RSTATE2	RSTATE1	RSTATE0
R-0	R-0	R-0	R-0
7	6	5	4
TFCOUNT4	TFCOUNT3	TFCOUNT2	TFCOUNT1
R-0	R-0	R-0	R-0
3	2	1	0
TFCOUNT0	TSTATE2	TSTATE1	TSTATE0
R-0	R-0	R-0	R-0

Примечание – Принятое условное обозначение: R – только чтение.

Рисунок 14.11 – Структура регистра отладки 2 DBG2
/адрес **80_402C**h – UART_0; адрес **80_404C**h – UART_1/

Таблица 14.11 – Описание битов регистра DBG2

Бит	Обозначение	Тип	Описание
31 – 24	RSV	R	Зарезервировано
23 – 22	FCR7, FCR6	R	Значение регистра управления FIFO (биты 7, 6); см. рисунок 14.3 и таблицу 14.5
21 – 17	MCR4 – MCR0	R	Значение регистра управления модемом (биты 4 – 0); см. рисунок 14.5 и таблицу 14.7
16 – 12	RFCOUNT4 – RFCOUNT0	R	Число символов в FIFO приёмника (RFCOUNT)
11 – 8	RSTATE3 – RSTATE0	R	Состояние конечного автомата (FSM) приёмника
7 – 3	TFCOUNT4 – TFCOUNT0	R	Количество символов в FIFO передатчика (TFCOUNT)
2 – 0	TSTATE2 – TSTATE0	R	Состояние конечного автомата (FSM) передатчика
Примечание – Принятое условное обозначение: R – только чтение.			

Регистр выбора источника тактирования SELCLKUART используется для выбора тактирования источника UART. На рисунке 14.12 приведена структура регистра SELCLKUART, а в таблице 14.12 – описание разрядов этого регистра.

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
RSV	RSV	RSV	RSV	RSV	RSV	RSV	UARTCLK
R-0	R-0	R-0	R-0	R-0	R-0	R-0	RW-0

Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.

Рисунок 14.12 – Структура регистра выбора источника тактирования SELCLKUART /адрес **80_4033**h – UART_0; адрес **80_4053**h – UART_1/

Таблица 14.12 – Описание битов регистра SELCLKUART

Бит	Обозначение	Тип	Описание
31 – 1	RSV	R	Зарезервировано
0	UARTCLK	RW	Тактирование UART (18,432 МГц): - 0 – тактирование от CPUCLK; - 1 – тактирование от внешнего вывода UART_CLK/X2
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

14.2 Пример программирования периферийного устройства UART

После RESET UART выполняет следующие действия:

- FIFO передатчика и приёмника очищаются;

- сдвиговые регистры передатчика и приёмника также очищаются;

- регистр делителя 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.

Далее приведён фрагмент программного кода, в делителях устанавливается значение, необходимое для получения частоты 38400 бит/с:

(устанавливаемое значение = ("тактовая частота")/(16×"желаемая скорость")):

```
  ; устанавливаем бит 7 LCR в 1, чтобы получить доступ 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
  ; устанавливаем бит 7 LCR в 0
        ldi @Line_Control, AR0
        and 0FF7Fh, AR0
        sti AR0, @Line_Control
```

После установки делителя необходимо установить нужный уровень спуска FIFO и разрешить желаемые прерывания. В приведённом далее фрагменте программного кода разрешаются внешние прерывания процессора, прерывание UART. Принятые данные готовы; уровень FIFO установлен равным восьми байтам.

```
        Or 2000h, ST; разрешаем глобальные прерывания ядра
        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
    ; очищаем 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
    ; проверяем бит0 (Data Ready)
        tstb 1, AR2
        bz Wait; если 0, то продолжаем ждать
    ; приём
        ldi @Receiver_Buffer, AR1
```

15 Периферийное устройство ARINC-429

Периферийное устройство ARINC-429 (далее – ARINC-429) реализует ГОСТ 18977-79 – стандарт двухпроводной компьютерной шины межсистемного обмена данными, предназначенной для применения в авионике. Размер слова составляет 32 бита, большинство сообщений состоит из единственного слова данных. ARINC-429 использует однонаправленный стандарт шины данных с физически разделёнными линиями передачи и приёма, сообщения передаются на одной из двух скоростей – 50 или 250 Кбит/с. Для каждого канала может быть установлена индивидуальная скорость приёма и передачи, включая и стандартные 12,5 и 100 Кбит/с.

Периферийное устройство ARINC-429 обеспечивает два режима работы с передатчиком/приёмником физической линии:

- Режим A_B (стандартный режим). В этом режиме по линии A (неинверсный вывод) передаётся/принимается значение положительного уровня передатчика/приёмника; по линии B (инверсный вывод) передаётся значение отрицательного уровня передатчика/приёмника.

- Режим D_S. В этом режиме данные передаются/принимаются по линии D (неинверсный вывод); по линии S (инверсный вывод) передаётся сигнал строба.

Стандарт ARINC-429 предусматривает три уровня сигнала: высокий (плюс 5 В), низкий (минус 5 В) и нулевой (0 В). Передача каждого бита посылки как высокого, так и низкого уровня обязательно сопровождается возвратом линии в нулевой уровень. Передаваемые слова разделены минимум четырьмя нулевыми битами. Передатчик всегда активен, он либо передаёт 32-битные слова данных или выдаёт нулевой уровень. Каждое слово представляет собой 32-разрядную последовательность, состоящую из пяти областей:

- Бит 32 является паритетным битом и используется для проверки того, что слово не было повреждено или искажено во время передачи (бит чётности).

- Биты 31, 30 являются матрицей знака/статуса (SSM) и часто указывают статус данных в слове:

  - рабочее состояние – предполагается, что в этом слове находятся данные;

  - тестовое состояние – данные в этом слове используются для тестовых целей;

  - отказ – отказ аппаратных средств, связанных с выдачей этого слова;

  - отсутствие данных – данные отсутствуют, неточны или устарели по причинам, несвязанным с отказом аппаратных средств;

  - данное поле может указывать знак (+/–) данных или быть частью информации.

  Для передачи отрицательного значения используется обратный код.

- Биты 29 – 11 содержат основные данные в различных представлениях – двоично-десятичном, бинарном дополнительном коде или в смешанном коде.

- Биты 10 – 9 идентифицируют источник/назначение (SDI), указывая, для какого приёмника предназначены данные или какая подсистема передала данные.

- Биты 8 – 1 содержат идентификатор (метку) типа данных в восьмеричном формате.

Самый младший бит каждого байта, за исключением метки, передаётся первым, метка является первой в каждой посылке. Порядок битов, передаваемых по шине ARINC-429, выглядит следующим образом: 8, 7, 6, 5, 4, 3, 2, 1, 9, 10, 11, 12, 13, ..., 32.

Блок ARINC-429, входящий в состав ИС 1867ВН016, содержит 8 модулей (каналов), которые индивидуально (независимо от других) могут быть сконфигурированы для работы в качестве передатчика или в качестве приёмника.

Каждый из 8 модулей для работы в режиме приёмника имеет два FIFO:

- FIFO меток на 8-битных значения;

- FIFO данных объёмом 128 32-разрядных слов, суммарно для всех модулей 1K × 32 бит.

Скорость приёма/передачи данных в 250 Кбит/с обеспечивается подачей на вход канала тактового сигнала с частотой, равной 2,5 МГц. K = F_{H1}/250 (где K – значение поля «div coeff» в регистре AR429RXTXCON***n***, см. биты 15 – 8) при значении clk div 8 = 1 (бит 16 регистра AR429RXTXCON***n***).

Скорость приёма/передачи данных в 50 Кбит/с обеспечивается подачей на вход канала тактового сигнала с частотой, равной 4 МГц. K = F_{H1}/50, (где K – значение поля «div coeff» в регистре AR429RXTXCON***n***, см. биты 15 – 8) при значении clk div 8 = 0 (бит 16 регистра AR429RXTXCON***n***).

При приёме предусмотрена возможность фильтрации принятой посылки по битам SDI (9 и 10 биты) путём их сравнения с соответствующими разрядами регистра AR429RX~~TX~~CON***n*** (см. биты 24, 25 этого регистра). Также имеется возможность фильтрации принятой посылки по меткам и по биту чётности.

В таблице 15.1 приведён список регистров ARINC-429.

Таблица 15.1 – Регистры ARINC-429

Адрес	Название	Назначение
80_4000h	AR429RXTXCON0	Регистр управления приёмником/передатчиком 0
80_4001h	AR429RXTXCON1	Регистр управления приёмником/передатчиком 1
80_4002h	AR429RXTXCON2	Регистр управления приёмником/передатчиком 2
80_4003h	AR429RXTXCON3	Регистр управления приёмником/передатчиком 3
80_4004h	AR429RXTXCON4	Регистр управления приёмником/передатчиком 4
80_4005h	AR429RXTXCON5	Регистр управления приёмником/передатчиком 5
80_4006h	AR429RXTXCON6	Регистр управления приёмником/передатчиком 6
80_4007h	AR429RXTXCON7	Регистр управления приёмником/передатчиком 7
80_4008h	AR429RXTXSTAT0	Регистр состояния приёмника/передатчика 0
80_4009h	AR429RXTXSTAT1	Регистр состояния приёмника/передатчика 1
80_400Ah	AR429RXTXSTAT2	Регистр состояния приёмника/передатчика 2
80_400Bh	AR429RXTXSTAT3	Регистр состояния приёмника/передатчика 3
80_400Ch	AR429RXTXSTAT4	Регистр состояния приёмника/передатчика 4
80_400Dh	AR429RXTXSTAT5	Регистр состояния приёмника/передатчика 5
80_400Eh	AR429RXTXSTAT6	Регистр состояния приёмника/передатчика 6
80_400Fh	AR429RXTXSTAT7	Регистр состояния приёмника/передатчика 7
80_4010h	AR429RXLABELFIFO	Регистр меток и номера FIFO
80_4011h	AR429RXTXDATA0	Регистр 32-разрядных данных приёмника/передатчика 0
80_4012h	AR429RXTXDATA1	Регистр 32-разрядных данных приёмника/передатчика 1
80_4013h	AR429RXTXDATA2	Регистр 32-разрядных данных приёмника/передатчика 2
80_4014h	AR429RXTXDATA3	Регистр 32-разрядных данных приёмника/передатчика 3
80_4015h	AR429RXTXDATA4	Регистр 32-разрядных данных приёмника/передатчика 4
80_4016h	AR429RXTXDATA5	Регистр 32-разрядных данных приёмника/передатчика 5
80_4017h	AR429RXTXDATA6	Регистр 32-разрядных данных приёмника/передатчика 6
80_4018h	AR429RXTXDATA7	Регистр 32-разрядных данных приёмника/передатчика 7

Независимо от того, сконфигурирован ли модуль с номером ***n*** (где ***n*** – номер канала; от 0 до 7) как приёмник или как передатчик, регистры AR429RXTXCON***n***, AR429RXTXSTAT***n*** имеют одинаковые для каждого модуля адреса (адреса AR429RX~~TX~~*****n*** не отличаются от адресов AR429~~RX~~TX*****n***, где * – CON или STAT), но отличаются структурой и назначением полей в режиме приёма и в режиме передачи.

Регистр AR429RXLABELFIFO является общим для всех модулей, сконфигурированных на приём. Регистры данных AR429RXTXDATA***n*** – универсальные 32-разрядные регистры (свой регистр для каждого из восьми модулей).

На рисунке 15.1 приведена структура регистра AR429RX~~TX~~CON***n*** в режиме приёма; в таблице 15.2 – описание его разрядов в этом режиме. Рисунок 15.2 и таблица 15.3 содержат соответствующую информацию об этом (AR429~~RX~~TXCON***n***) регистре в режиме передачи.

31	30	29	28	27	26	25	24
fifo not empty int mask	fifo halffull int mask	fifo full int mask	fifo label full int mask	error int mask	label compare en	sdi bit 10	sdi bit 9
RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0
23	22	21	20	19	18	17	16
sdi comp en	async rx	flush label fifo	flush fifo	A_B #D_S inputs	parity to odd	parity en	clk div 8
RWC-0	RWC-1	RWC-0	RWC-0	RWC-0	RWC-1	RWC-0	RWC-0
15	14	13	12	11	10	9	8
div coeff							
RWC-**01**h							
7	6	5	4	3	2	1	0
0	0	0	int en	0	0	m is tx	en m
RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0

Примечание – Принятые условные обозначения: RWC-* – чтение, запись и значение при сбросе.

Здесь и далее в названиях рисунков /адрес регистра/ (***n*** = 0–7) – номер канала ARINC-429

Например:

адрес регистра AR429RXTXCON3 для канала **3** ARINC-429 – 80_400(***n***)h = 80_400**3**h;
адрес регистра AR429RXTXSTAT3 для канала **3** ARINC-429 – 80_400(***n*+*8***)0h = 80_400**B**h;
адрес регистра AR429RXTXDATA3 для канала **3** ARINC-429 – 80_401(***n*+*1***)0h = 80_401**4**h.

Рисунок 15.1 – Структура регистра управления в режиме приёма AR429RX~~TX~~CON***n*** /адрес **80_400(*n*)**h/

Таблица 15.2 – Описание разрядов регистра AR429RX~~TX~~CON***n*** в режиме приёма

Биты	Обозначение	Описание
1	2	3
31	fifo empty int mask	Маска прерывания по опустошению буфера FIFO данных: - fifo empty int mask = 1 – прерывание разрешено; - fifo empty int mask = 0 – прерывание запрещено
30	fifo halffull int mask	Маска прерывания по заполнению наполовину FIFO данных: - fifo halffull int mask = 1 – прерывание разрешено; - fifo halffull int mask = 0 – прерывание запрещено
29	fifo full int mask	Маска прерывания по полному заполнению FIFO данных: - fifo full int mask = 1 – прерывание разрешено; - fifo full int mask = 0 – прерывание запрещено
28	fifo label full int mask	Маска прерывания по полному заполнению FIFO меток: - fifo label full int mask = 1 – прерывание разрешено; - fifo label full int mask = 0 – прерывание запрещено
27	error int mask	Маска прерывания при ошибке приёма (по чётности): - error int mask = 1 – прерывание при ошибке приёма; - error int mask = 0 – прерывание запрещено
26	label compare en	Включение сравнения принятой посылки с метками из буфера меток FIFO: - label compare en = 1 – принятая посылка сравнивается с метками; - label compare en = 0 – принятая посылка ни с чем не сравнивается

Окончание таблицы 15.2

1	2	3
25	sdi bit 10	Значение бита 10 SDI для сравнения и фильтрации принятой посылки.
24	sdi bit 9	Значение бита 9 SDI для сравнения и фильтрации принятой посылки.
23	sdi comp en	Включение разрешения сравнения по битам SDI: - sdi comp en = 1 – разрешение сравнения по битам SDI; - sdi comp en = 0 – запрет сравнения по битам SDI.
22	async rx	Режим приёма, - async rx = 1 – приём асинхронный; - async rx = 0 – приём синхронный.
21	flush label fifo	- flush label fifo = 1 – сброс меток FIFO; - flush label fifo = 0, – метки FIFO не сбрасываются.
20	flush fifo	- flush fifo = 1 – FIFO данные сбрасываются; - flush fifo = 0 – FIFO данные не сбрасываются.
19	A_B #D_S inputs	Выбор режима работы линий: - A_B #D_S inputs = 0 – выбирается режим работы A_B; - A_B #D_S inputs = 1 – выбирается режим работы D_S.
18	parity to odd	- parity to odd = 1 – бит чётности дополняется до нечётного; - parity to odd = 0 – бит чётности дополняется до чётного. Активен только тогда, когда бит «parity en» = 1 (см. бит 17)
17	parity en	- parity en = 1 – включается проверка на чётность/нечётность; - parity en = 0 – выключается проверка на чётность/нечётность.
16	clk div 8	Бит включения делителя скорости приёма модулей 0…7: - clk div 8 = 0 – скорость приёма равна 250 Кбит/с [1]; - clk div 8 = 1 – скорость приёма равна 50 Кбит/с [2]. [1] Скорость приёма данных 250 Кбит/с обеспечивается при частоте на входе канала равной 2,5 МГц. K = F_{H1}/250. [2] Скорость приёма данных 50 Кбит /с обеспечивается при частоте на входе канала равной 4 МГц. K = F_{H1}/50.
15 – 8	div coeff	Значение коэффициента K деления частоты тактового сигнала процессорного ядра H1: $K = F_{H1}/F_{ARINC}$ (15.1)
7 – 5	0	Не используются.
4	int en	- int en = 1 – включение прерывания от модуля; - int en = 0 – выключение прерывания от модуля
3 – 2	0	Не используются.
1	m is tx	- m is tx = 1 – модуль работает как передатчик; - m is tx = 0 – модуль работает как приёмник.
0	en m	- en m = 1 – включение модуля; - en m = 0 – выключение модуля.

31	30	29	28	27	26	25	24
fifo not empty int mask	fifo halffull int mask	fifo full int mask	0	0	0	0	0
RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0
23	22	21	20	19	18	17	16
0	posedge data set	0	flush fifo	A_B #D_S outputs	parity to odd	parity en	clk div 8
RWC-0	RWC-1	RWC-0	RWC-0	RWC-0	RWC-1	RWC-0	RWC-0
15	14	13	12	11	10	9	8
div coeff							
RWC-**01**h							
7	6	5	4	3	2	1	0
0	0	0	int en	0	0	m is tx	en m
RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0

Примечание – Принятое условное обозначение: RWC-* – чтение, запись и значение при сбросе.

Рисунок 15.2 – Структура регистра управления в режиме передачи AR429~~RX~~TXCON***n*** /адрес **80_400(*n*)**h/

Таблица 15.3 – Описание разрядов регистра AR429~~RX~~TXCON***n*** в режиме передачи

Биты	Обозначение	Описание
1	2	3
31	fifo empty int mask	Маска прерывания по опустошению буфера FIFO данных: - fifo empty int mask = 1 – прерывание разрешено; - fifo empty int mask = 0 – прерывание запрещено.
30	fifo halffull int mask	Маска прерывания по заполнению наполовину FIFO данных: - fifo halffull int mask = 1 – прерывание разрешено; - fifo halffull int mask = 0 – прерывание запрещено.
29	fifo full int mask	Маска прерывания по полному заполнению FIFO данных: - fifo full int mask = 1 – прерывание разрешено; - fifo full int mask = 0 – прерывание запрещено.
28 – 23	0	Не используются.
22	posedge data set	Используется, когда бит A_B #D_S outputs = 0. - posedge data set = 1 – выставление данных по переднему фронту сигнала strobe; - posedge data set = 0 – выставление данных по заднему фронту сигнала strobe.
21	0	Не используется
20	flush fifo	- flush fifo = 1 – FIFO данные сбрасываются; - flush fifo = 0 – FIFO данные не сбрасываются.
19	A_B # D_S outputs	Выбор режима работы линий: - A_B # D_S outputs = 1 – выбирается режим работы A_B; - A_B # D_S outputs = 0 – выбирается режим работы D_S.

Окончание таблицы 15.3

1	2	3
18	parity to odd	- parity to odd = 1 – бит чётности дополняется до нечётного; - parity to odd = 0 – бит чётности дополняется до чётного. Активен только тогда, когда бит «parity en» = 1 (см. бит 17)
17	parity en	- parity en = 1 – включается проверка на чётность/нечётность; - parity en = 0 – выключается проверка на чётность/нечётность.
16	clk div 8	Бит включения делителя скорости передачи модулей 0..7: - clk div 8 = 0 – скорость передачи равна 250 Кбит/с [1]; - clk div 8 = 1 – скорость передачи равна 50 Кбит/с [2]. [1] Скорость передачи данных 250 Кбит/с обеспечивается при частоте на входе канала равной 2,5 МГц. K = F_{H1}/250. [2] Скорость передачи данных 50 Кбит /с обеспечивается при частоте на входе канала равной 4 МГц. K = F_{H1}/50.
15 – 8	div coeff	Значение коэффициента K деления частоты тактового сигнала процессорного ядра H1 по формуле (15.1): $K = F_{H1}/F_{ARINC}$
7 – 5	0	Не используются.
4	int en	- int en = 1 – включение прерывания от модуля; - int en = 0 – выключение прерывания от модуля
3 – 2	0	Не используются
1	m is tx	- m is tx = 1 – модуль работает как передатчик; - m is tx = 0 – модуль работает как приёмник.
0	en m	- en m = 1 – включение модуля; - en m = 0 – выключение модуля.

На рисунке 15.3 приведена структура регистра состояния AR429RX~~TX~~STAT***n*** в режиме приёма; в таблице 15.4 – описание его разрядов в этом режиме. Рисунок 15.4 и таблица 15.5 содержат соответствующую информацию о регистре AR429~~RX~~TXSTAT***n*** в режиме передачи.

31	30	29	28	27	26	25	24
0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R
23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R
15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R
7	6	5	4	3	2	1	0
fifo not empty status	fifo halffull status	error receive	error parity	fifo label full	fifo full	fifo halffull	fifo not empty
RWC-1	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0

Примечание – Принятые условные обозначения: R – чтение, RWC-* – чтение, запись и значение при сбросе.

Рисунок 15.3 – Структура регистра состояния в режиме приёма AR429RX~~TX~~STAT***n*** /адрес **80_400(*n*+*8*)**h

Таблица 15.4 – Описание разрядов регистра AR429RX~~TX~~STAT***n*** в режиме приёма

Биты	Обозначение	Описание
1	2	3
31 – 8	0	Не используются
7	fifo not empty status[#]	- fifo not empty status = 1 – в FIFO есть данные; - fifo not empty status = 0 – в FIFO нет данных.
6	fifo halffull status[#]	- fifo halffull status = 1 – FIFO данных заполнен наполовину или больше; - fifo halffull status = 0 – FIFO данных заполнен меньше чем наполовину.
5	error receive	Ошибка при приёме: - error receive = 1 – ошибка приёма; - error receive = 0 – приём без ошибок. * бит error receive очищается записью в него 1
4	error parity	Ошибка чётности при приёме: - error parity = 1 – ошибка чётности при приёме; - error parity = 0 – нет ошибок чётности при приёме * бит error parity очищается записью в него 1
3	fifo label full	Статусный бит, выставляемый при полном заполнении FIFO меток: - fifo label full = 1 – FIFO меток полностью заполнен; - fifo label full = 0 – FIFO меток частично заполнен или пуст; * бит fifo label full очищается записью в него 1
2	fifo full	- fifo full = 1 – FIFO данных полностью заполнен; - fifo full = 0 – FIFO данных частично заполнен или пуст. * бит fifo full очищается записью в него 1

Окончание таблицы 15.4

1	2	3
1	fifo halffull	- fifo halffull = 1 – FIFO данных заполнен наполовину или больше; - fifo halffull = 0 – FIFO данных заполнен меньше чем наполовину. * бит fifo halffull очищается записью в него 1
0	fifo not empty	- fifo not empty = 1 – FIFO есть данные; - fifo not empty = 0 – FIFO нет данных. * бит fifo not empty очищается записью в него 1
Примечание – Принятые условные обозначения: # – биты 7, 6 являются реальными индикаторами текущего (для каждого такта) состояния соответствующих сигналов, в отличие от битов 5–0, включая одноимённые «*fifo not empty*» (биты 7 и 0) и «*fifo halffull*» (биты 6 и 1), которые функционируют в режиме флагов – будучи единожды установленными (в 1) остаются неизменными, вплоть до их программной очистки путём записи 1 в соответствующий бит (см. *).		

31	30	29	28	27	26	25	24
0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R
23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R
15	14	13	12	11	10	9	8
0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R
7	6	5	4	3	2	1	0
fifo not empty status	fifo halffull status	0	0	0	fifo full	fifo halffull	fifo not empty
RWC-1	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0

Примечание – Принятые условные обозначения: R – чтение, RWC-(0/1) – чтение, запись и значение при сбросе.

Рисунок 15.4 – Структура регистра состояния в режиме передачи AR429~~RX~~TXSTAT***n*** /адрес **80_400(*n*+*8*)**h

Таблица 15.5 – Описание разрядов регистра AR429~~RX~~TXSTAT***n*** в режиме передачи

Биты	Обозначение	Описание
31 – 8	0	Не используются
7	fifo not empty status#	- fifo not empty status = 1 – в FIFO есть данные; - fifo not empty status = 0 – в FIFO нет данных.
6	fifo halffull status#	- fifo halffull status = 1 – FIFO данных заполнен наполовину или больше; - fifo halffull status = 0 – FIFO данных заполнен меньше чем наполовину.
5 – 3	0	Не используются
2	fifo full	- fifo full = 1 – FIFO данных полностью заполнен; - fifo full = 0 – FIFO данных частично заполнен или пуст. * бит fifo full очищается записью в него 1
1	fifo halffull	- fifo halffull = 1 – FIFO данных заполнен наполовину или больше; - fifo halffull = 0 – FIFO данных заполнен меньше чем наполовину. * бит fifo halffull очищается записью в него 1
0	fifo not empty	- fifo not empty = 1 – FIFO есть данные; - fifo not empty = 0 – FIFO нет данных. * бит fifo not empty очищается записью в него 1
Примечание – Принятые условные обозначения: # – биты 7–6 являются реальными индикаторами текущего (для каждого такта) состояния соответствующих сигналов, в отличие от битов 5–0, включая одноимённые «*fifo not empty*» (биты 7 и 0) и «*fifo halffull*» (биты 6 и 1), которые функционируют в режиме флагов – будучи единожды установленными (в 1) остаются неизменными, вплоть до их программной очистки путём записи 1 в соответствующий бит (см. *).		

На рисунке 15.5 приведена структура регистра меток и номера FIFO AR429RXLABELFIFO в режиме приёма; в таблице 15.6 – описание его полей в этом режиме. Для модулей, сконфигурированных на работу в режиме передачи, регистр AR429RXLABELFIFO не используется.

31	30	29	28	27	26	25	24
0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R
23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0
R	R	R	R	R	R	R	R
15	14	13	12	11	10	9	8
0	0	0	0	0	number of fifo module		
R	R	R	R	R	W	W	W
7	6	5	4	3	2	1	0
label value for write							
W	W	W	W	W	W	W	W

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

Рисунок 15.5 – Структура регистра меток и номера FIFO AR429RXLABELFIFO /адрес **80_4010**h/

Таблица 15.6 – Описание полей регистра AR429RXLABELFIFO

Биты	Обозначение	Описание
31 – 11	0	Не используются.
10 – 8	number of fifo module	Номер модуля FIFO меток, в который будет записано значение метки.
7 – 0	label value for write	Значение метки для записи в FIFO.

При обращении к регистру AR429RXLABELFIFO пользователь должен указать в битах 10–8 номер модуля, в который будет записана метка из битов 7–0. Максимальное количество меток, хранимых одним модулем, равняется 32.

При конфигурировании модуля в качестве приёмника и установке в 1 бита label compare en в регистре AR429RX~~TX~~CON***n*** (бит 26), происходит сравнение принятой посылки с метками из FIFO меток и последующая фильтрация принятых посылок. Если метка совпадает с каким-либо из значений FIFO меток модуля, то посылка принимается, иначе посылка игнорируется.

AR429RXTXDATA***n*** – универсальные 32-разрядные регистры данных, не изменяющие состояние при сбросе и доступные только на чтение в режиме приёма, и только на запись в режиме передачи /адрес 80_401(***n+1***)h/.

16 Порты ввода-вывода общего назначения GPIO

«Система в корпусе» ИС 1867ВН016 содержит два блока портов ввода-вывода общего назначения (GPIO), каждый из которых может управлять 16 внешними выводами ИС. В таблице 16.1 приведено соответствие номеров разрядов регистров данных блоков GPIO и координат внешних выводов микросхемы.

Таблица 16.1 – Соответствие номеров разрядов регистров и координат выводов ИС

Номер блока GPIO	Номера разрядов регистров данных	Координаты выводов микросхемы
1	15	T25
	14	V26
	13	Y26
	12	Y25
	11	Y24
	10	AA24
	9	AB27
	8	AC27
	7	AC25
	6	AC24
	5	AD24
	4	AD26
	3	AD27
	2	AE26
	1	AF26
	0	AF27
0	15	AF19
	14	AE19
	13	AE20
	12	AF20
	11	AG20
	10	AG22
	9	AF22
	8	AD23
	7	AE23
	6	AF23
	5	AF24
	4	AE24
	3	AE25
	2	AF25
	1	AG25
	0	AG26

Любой из блоков GPIO может быть подключен через коммутатор к одному из процессорных ядер. Каждый блок GPIO поддерживает прерывания, которые генерируются при изменении состояний на входах.

Каждый бит в регистре данных блока GPIO соответствует определённому выводу и может быть индивидуально сконфигурирован как вход или как выход.

Условия генерации прерывания по конкретному входу могут быть установлены в зависимости от полярности, уровня или фронта входного сигнала.

На рисунке 16.1 приведена структурная схема одного вывода блока GPIO.

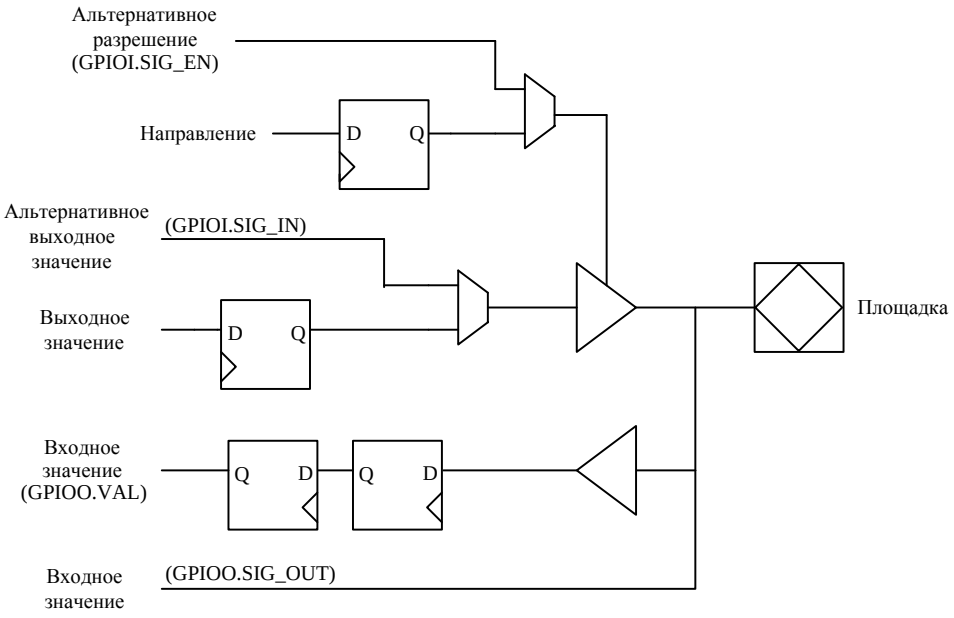


Рисунок 16.1 – Структурная схема вывода блока GPIO

16.1 Операции GPIO

Каждый порт ввода-вывода реализован в виде двунаправленного буфера с программируемым разрешением выхода.

Вход извне микросхемы в буфер синхронизируется двумя последовательно включенными триггерами для устранения метастабильного (неопределённого) состояния на этом входе. Синхронизированные значения входных сигналов программно доступны в регистре IOPDATINR*n*.

Буфер управляется регистром направления порта ввода-вывода IOPDIRR*n*. Установка в 1 бита, соответствующего определённому выводу, конфигурирует этот вывод как выход; при этом уровень сигнала на том или ином выходе определяется ассоциированным с ним битом регистра выходных данных IOPDATOUTR*n*.

Блок GPIO позволяет реализовать три различных варианта генерации прерываний. Условия генерации прерываний определяются тремя регистрами: регистром маски прерываний INTMSKR*n*, регистром полярности INTPOLR*n* и регистром фронта сигнала прерывания INTEDGR*n*. Для разрешения прерывания необходимо установить в 1 соответствующий бит в регистре маски прерываний.

Если бит в регистре фронта установлен в 0, прерывание на входе, который соответствует этому биту, будет генерироваться по уровню.

Если бит в регистре полярности установлен в 0, прерывание будет генерироваться по низкому уровню сигнала на соответствующем входе, в противном случае (когда бит регистра полярности в 1) – по высокому уровню сигнала на соответствующем входе.

Если бит регистра фронта установлен в 1, прерывание будет происходить по изменению фронта входного сигнала**.** Регистр полярности определяет детектирование по переднему (нарастающему) фронту при 1 в соответствующем разряде или по заднему ниспадающему фронту при 0.

16.2 Описание регистров GPIO

Управление блоками (0 и 1) GPIO осуществляется через картированные в памяти регистры: адреса, названия (мнемоники) и назначения которых приведены в таблице 16.2.

Таблица 16.2 – Регистры блоков GPIO

Адрес	Название	Функциональное назначение
GPIO_0		
80_44F0h	IOPDATINR0	Регистр входных данных порта ввода-вывода GPIO_0
80_44F1h	IOPDATOUTR0	Регистр выходных данных порта ввода-вывода GPIO_0
80_44F2h	IOPDIRR0	Регистр направления порта ввода-вывода GPIO_0
80_44F3h	INTMSKR0	Регистр маски прерываний GPIO_0
80_44F4h	INTPOLR0	Регистр полярности прерываний GPIO_0
80_44F5h	INTEDGR0	Регистр фронта прерываний GPIO_0
80_44F6h	BPSR0	Регистр обхода GPIO_0; не используется
80_44F7h	CPBLR0	Регистр возможностей GPIO_0; не используется
80_4500h	INTFR0	Регистр флагов прерываний GPIO_0
80_4520h – 80_453Ch	MAPR0	Регистр(ы) карты прерываний GPIO_0; не используется (не используются)
GPIO_1		
80_4540h	IOPDATINR1	Регистр входных данных порта ввода-вывода GPIO_1
80_4541h	IOPDATOUTR1	Регистр выходных данных порта ввода-вывода GPIO_1
80_4542h	IOPDIRR1	Регистр направления порта ввода-вывода GPIO_1
80_4543h	INTMSKR1	Регистр маски прерываний GPIO_1
80_4544h	INTPOLR1	Регистр полярности прерываний GPIO_1
80_4545h	INTEDGR1	Регистр фронта прерываний GPIO_1
80_4546h	BPSR1	Регистр обхода GPIO_1; не используется
80_4547h	CPBLR1	Регистр возможностей GPIO_1; не используется
80_4550h	INTFR1	Регистр флагов прерываний GPIO_1
80_4560h – 80_457Ch	MAPR1	Регистр(ы) карты прерываний GPIO_1; не используется (не используются)

На рисунке 16.2 приведена структура регистра входных данных блока GPIO, а в таблице 16.3 – описание разрядов этого регистра.

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
15	14	13	12	11	10	9	8
IN15	IN14	IN13	IN12	IN11	IN10	IN9	IN8
R	R	R	R	R	R	R	R
7	6	5	4	3	2	1	0
IN7	IN6	IN5	IN4	IN3	IN2	IN1	IN0
R	R	R	R	R	R	R	R

Примечание – Принятые условные обозначения: R – чтение, X – значение не определено.

Рисунок 16.2 – Структура регистра входных данных блока GPIO IOPDATINR***n*** /адрес **80_44F0**h – GPIO_0; адрес **80_4540**h – GPIO_1/

Таблица 16.3 – Описание битов регистра входных данных (уровней входных сигналов)

Бит	Обозначение	Тип	Описание
31 – 16	RSV	R	Зарезервировано.
15	IN15	R	- IN15 = 0 – на соответствующем входе низкий уровень; - IN15 = 1 – на соответствующем входе высокий уровень.
14	IN14	R	- IN14 = 0 – на соответствующем входе низкий уровень; - IN14 = 1 – на соответствующем входе высокий уровень.
13	IN13	R	- IN13 = 0 – на соответствующем входе низкий уровень; - IN13 = 1 – на соответствующем входе высокий уровень.
12	IN12	R	- IN12 = 0 – на соответствующем входе низкий уровень; - IN12 = 1 – на соответствующем входе высокий уровень.
11	IN11	R	- IN11 = 0 – на соответствующем входе низкий уровень; - IN11 = 1 – на соответствующем входе высокий уровень.
10	IN10	R	- IN10 = 0 – на соответствующем входе низкий уровень; - IN10 = 1 – на соответствующем входе высокий уровень.
9	IN9	R	- IN9 = 0 – на соответствующем входе низкий уровень; - IN9 = 1 – на соответствующем входе высокий уровень.
8	IN8	R	- IN8 = 0 – на соответствующем входе низкий уровень; - IN8 = 1 – на соответствующем входе высокий уровень.
7	IN7	R	- IN7 = 0 – на соответствующем входе низкий уровень; - IN7 = 1 – на соответствующем входе высокий уровень.
6	IN6	R	- IN6 = 0 – на соответствующем входе низкий уровень; - IN6 = 1 – на соответствующем входе высокий уровень.
5	IN5	R	- IN5 = 0 – на соответствующем входе низкий уровень; - IN5 = 1 – на соответствующем входе высокий уровень.
4	IN4	R	- IN4 = 0 – на соответствующем входе низкий уровень; - IN4 = 1 – на соответствующем входе высокий уровень.
3	IN3	R	- IN3 = 0 – на соответствующем входе низкий уровень; - IN3 = 1 – на соответствующем входе высокий уровень.
2	IN2	R	- IN2 = 0 – на соответствующем входе низкий уровень; - IN2 = 1 – на соответствующем входе высокий уровень.
1	IN1	R	- IN1 = 0 – на соответствующем входе низкий уровень; - IN1 = 1 – на соответствующем входе высокий уровень.
0	IN0	R	- IN0 = 0 – на соответствующем входе низкий уровень; - IN0 = 1 – на соответствующем входе высокий уровень.
Примечание – Принятое условное обозначение: R – только чтение.			

На рисунке 16.3 приведена структура регистра выходных данных блока GPIO, а в таблице 16.4 – описание разрядов этого регистра.

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
15	14	13	12	11	10	9	8
OUT15	OUT14	OUT13	OUT12	OUT11	OUT10	OUT9	OUT8
WC-0	WC-0	WC-0	WC-0	WC-0	WC-0	WC-0	WC-0
7	6	5	4	3	2	1	0
OUT7	OUT6	OUT5	OUT4	OUT3	OUT2	OUT1	OUT0
WC-0	WC-0	WC-0	WC-0	WC-0	WC-0	WC-0	WC-0

Примечание – Принятые условные обозначения: R – чтение, W – запись, C – бит сбрасывается, X – значение не определено.

Рисунок 16.3 – Структура регистра выходных данных блока GPIO IOPDATOUTR***n*** /адрес **80_44F1**h – GPIO_0; адрес **80_4541**h – GPIO_1/

Таблица 16.4 – Описание битов регистра выходных данных (уровней выходных сигналов)

Бит	Обозначение	Тип	Описание
31 – 16	RSV	R	Зарезервировано.
15	OUT15	WC	- OUT15 = 0 – на соответствующем выходе низкий уровень; - OUT15 = 1 – на соответствующем выходе высокий уровень.
14	OUT14	WC	- OUT14 = 0 – на соответствующем выходе низкий уровень; - OUT14 = 1 – на соответствующем выходе высокий уровень.
13	OUT13	WC	- OUT13 = 0 – на соответствующем выходе низкий уровень; - OUT13 = 1 – на соответствующем выходе высокий уровень.
12	OUT12	WC	- OUT12 = 0 – на соответствующем выходе низкий уровень; - OUT12 = 1 – на соответствующем выходе высокий уровень.
11	OUT11	WC	- OUT11 = 0 – на соответствующем выходе низкий уровень; - OUT11 = 1 – на соответствующем выходе высокий уровень.
10	OUT10	WC	- OUT10 = 0 – на соответствующем выходе низкий уровень; - OUT10 = 1 – на соответствующем выходе высокий уровень.
9	OUT9	WC	- OUT9 = 0 – на соответствующем выходе низкий уровень; - OUT9 = 1 – на соответствующем выходе высокий уровень.
8	OUT8	WC	- OUT8 = 0 – на соответствующем выходе низкий уровень; - OUT8 = 1 – на соответствующем выходе высокий уровень.
7	OUT7	WC	- OUT7 = 0 – на соответствующем выходе низкий уровень; - OUT7 = 1 – на соответствующем выходе высокий уровень.
6	OUT6	WC	- OUT6 = 0 – на соответствующем выходе низкий уровень; - OUT6 = 1 – на соответствующем выходе высокий уровень.
5	OUT5	WC	- OUT5 = 0 – на соответствующем выходе низкий уровень; - OUT5 = 1 – на соответствующем выходе высокий уровень.
4	OUT4	WC	- OUT4 = 0 – на соответствующем выходе низкий уровень; - OUT4 = 1 – на соответствующем выходе высокий уровень.
3	OUT3	WC	- OUT3 = 0 – на соответствующем выходе низкий уровень; - OUT3 = 1 – на соответствующем выходе высокий уровень.
2	OUT2	WC	- OUT2 = 0 – на соответствующем выходе низкий уровень; - OUT2 = 1 – на соответствующем выходе высокий уровень.
1	OUT1	WC	- OUT1 = 0 – на соответствующем выходе низкий уровень; - OUT1 = 1 – на соответствующем выходе высокий уровень.
0	OUT0	WC	- OUT0 = 0 – на соответствующем выходе низкий уровень; - OUT0 = 1 – на соответствующем выходе высокий уровень.
Примечание – Принятые условные обозначения: R –чтение, W – запись, C – сброс.			

На рисунке 16.4 приведена структура регистра направления выводов блока GPIO, а в таблице 16.5 – описание разрядов этого регистра.

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
15	14	13	12	11	10	9	8
DIR15	DIR14	DIR13	DIR12	DIR11	DIR10	DIR9	DIR8
RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0
7	6	5	4	3	2	1	0
DIR7	DIR6	DIR5	DIR4	DIR3	DIR2	DIR1	DIR0
RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0

Примечание – Принятые условные обозначения: R – чтение, W – запись, C – бит сбрасывается, X – значение не определено.

Рисунок 16.4 – Структура регистра направления выводов блока GPIO IOPDIRR***n*** /адрес **80_44F2**h – GPIO_0; адрес **80_4542**h – GPIO_1/

Таблица 16.5 – Описание битов регистра направления выводов

Бит	Обозначение	Тип	Описание
31 – 16	RSV	R	Зарезервировано.
15	DIR15	RWC	- DIR15 = 0 – соответствующий вывод работает как вход; - DIR15 = 1 – соответствующий вывод работает как выход.
14	DIR14	RWC	- DIR14 = 0 – соответствующий вывод работает как вход; - DIR14 = 1 – соответствующий вывод работает как выход.
13	DIR13	RWC	- DIR13 = 0 – соответствующий вывод работает как вход; - DIR13 = 1 – соответствующий вывод работает как выход.
12	DIR12	RWC	- DIR12 = 0 – соответствующий вывод работает как вход; - DIR12 = 1 – соответствующий вывод работает как выход.
11	DIR11	RWC	- DIR11 = 0 – соответствующий вывод работает как вход; - DIR11 = 1 – соответствующий вывод работает как выход.
10	DIR10	RWC	- DIR10 = 0 – соответствующий вывод работает как вход; - DIR10 = 1 – соответствующий вывод работает как выход.
9	DIR9	RWC	- DIR9 = 0 – соответствующий вывод работает как вход; - DIR9 = 1 – соответствующий вывод работает как выход.
8	DIR8	RWC	- DIR8 = 0 – соответствующий вывод работает как вход; - DIR8 = 1 – соответствующий вывод работает как выход.
7	DIR7	RWC	- DIR7 = 0 – соответствующий вывод работает как вход; - DIR7 = 1 – соответствующий вывод работает как выход.
6	DIR6	RWC	- DIR6 = 0 – соответствующий вывод работает как вход; - DIR6 = 1 – соответствующий вывод работает как выход.
5	DIR5	RWC	- DIR5 = 0 – соответствующий вывод работает как вход; - DIR5 = 1 – соответствующий вывод работает как выход.
4	DIR4	RWC	- DIR4 = 0 – соответствующий вывод работает как вход; - DIR4 = 1 – соответствующий вывод работает как выход.
3	DIR3	RWC	- DIR3 = 0 – соответствующий вывод работает как вход; - DIR3 = 1 – соответствующий вывод работает как выход.
2	DIR2	RWC	- DIR2 = 0 – соответствующий вывод работает как вход; - DIR2 = 1 – соответствующий вывод работает как выход.
1	DIR1	RWC	- DIR1 = 0 – соответствующий вывод работает как вход; - DIR1 = 1 – соответствующий вывод работает как выход.
0	DIR0	RWC	- DIR0 = 0 – соответствующий вывод работает как вход; - DIR0 = 1 – соответствующий вывод работает как выход.
Примечание – Принятые условные обозначения: R –чтение, W – запись, C – сброс.			

На рисунке 16.5 приведена структура регистра маски прерывания блока GPIO, а в таблице 16.6 – описание разрядов этого регистра.

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
15	14	13	12	11	10	9	8
EINT15	EINT14	EINT13	EINT12	EINT11	EINT10	EINT9	EINT8
RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0
7	6	5	4	3	2	1	0
EINT7	EINT6	EINT5	EINT4	EINT3	EINT2	EINT1	EINT0
RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0

Примечание – Принятые условные обозначения: R – чтение, W – запись, C – бит сбрасывается, X – значение не определено.

Рисунок 16.5 – Структура регистра маски прерывания блока GPIO INTMSKR***n*** /адрес **80_44F3**h – GPIO_0; адрес **80_4543**h – GPIO_1/

Таблица 16.6 – Описание битов регистра маски прерывания

Бит	Обозначение	Тип	Описание
31–16	RSV	R	Зарезервировано.
15	EINT15	RWC	- EINT15 = 0 – прерывание от соответствующего вывода запрещено; - EINT15 = 1 – прерывание от соответствующего вывода разрешено.
14	EINT14	RWC	- EINT14 = 0 – прерывание от соответствующего вывода запрещено; - EINT14 = 1 – прерывание от соответствующего вывода разрешено.
13	EINT13	RWC	- EINT13 = 0 – прерывание от соответствующего вывода запрещено; - EINT13 = 1 – прерывание от соответствующего вывода разрешено.
12	EINT12	RWC	- EINT12 = 0 – прерывание от соответствующего вывода запрещено; - EINT12 = 1 – прерывание от соответствующего вывода разрешено.
11	EINT11	RWC	- EINT11 = 0 – прерывание от соответствующего вывода запрещено; - EINT11 = 1 – прерывание от соответствующего вывода разрешено.
10	EINT10	RWC	- EINT10 = 0 – прерывание от соответствующего вывода запрещено; - EINT10 = 1 – прерывание от соответствующего вывода разрешено.
9	EINT9	RWC	- EINT9 = 0 – прерывание от соответствующего вывода запрещено; - EINT9 = 1 – прерывание от соответствующего вывода разрешено.
8	EINT8	RWC	- EINT8 = 0 – прерывание от соответствующего вывода запрещено; - EINT8 = 1 – прерывание от соответствующего вывода разрешено.
7	EINT7	RWC	- EINT7 = 0 – прерывание от соответствующего вывода запрещено; - EINT7 = 1 – прерывание от соответствующего вывода разрешено.
6	EINT6	RWC	- EINT6 = 0 – прерывание от соответствующего вывода запрещено; - EINT6 = 1 – прерывание от соответствующего вывода разрешено.
5	EINT5	RWC	- EINT5 = 0 – прерывание от соответствующего вывода запрещено; - EINT5 = 1 – прерывание от соответствующего вывода разрешено.
4	EINT4	RWC	- EINT4 = 0 – прерывание от соответствующего вывода запрещено; - EINT4 = 1 – прерывание от соответствующего вывода разрешено.
3	EINT3	RWC	- EINT3 = 0 – прерывание от соответствующего вывода запрещено; - EINT3 = 1 – прерывание от соответствующего вывода разрешено.
2	EINT2	RWC	- EINT2 = 0 – прерывание от соответствующего вывода запрещено; - EINT2 = 1 – прерывание от соответствующего вывода разрешено.
1	EINT1	RWC	- EINT1 = 0 – прерывание от соответствующего вывода запрещено; - EINT1 = 1 – прерывание от соответствующего вывода разрешено.
0	EINT0	RWC	- EINT0 = 0 – прерывание от соответствующего вывода запрещено; - EINT0 = 1 – прерывание от соответствующего вывода разрешено.
Примечание – Принятые условные обозначения: R –чтение, W – запись, C – сброс.			

На рисунке 16.6 приведена структура регистра полярности прерываний блока GPIO, а в таблице 16.7 – описание разрядов этого регистра.

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
15	14	13	12	11	10	9	8
PLR15	PLR14	PLR13	PLR12	PLR11	PLR10	PLR9	PLR8
RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0
7	6	5	4	3	2	1	0
PLR7	PLR6	PLR5	PLR4	PLR3	PLR2	PLR1	PLR0
RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0

Примечание – Принятые условные обозначения: R – чтение, W – запись, C – бит сбрасывается, X – значение не определено.

Рисунок 16.6 – Структура регистра полярности прерываний блока GPIO INTPOLR***n*** /адрес **80_44F4**h – GPIO_0; адрес **80_4544**h – GPIO_1/

Таблица 16.7 – Описание битов регистра полярности прерываний

Бит	Обозначение	Тип	Описание
31–16	RSV	R	Зарезервировано.
15	PLR15	RWC	- PLR15 = 0 – прерывание по низкому уровню/заднему фронту; - PLR15 = 1 – прерывание по высокому уровню/переднему фронту.
14	PLR14	RWC	- PLR14 = 0 – прерывание по низкому уровню/заднему фронту; - PLR14 = 1 – прерывание по высокому уровню/переднему фронту.
13	PLR13	RWC	- PLR13 = 0 – прерывание по низкому уровню/заднему фронту; - PLR13 = 1 – прерывание по высокому уровню/переднему фронту.
12	PLR12	RWC	- PLR12 = 0 – прерывание по низкому уровню/заднему фронту; - PLR12 = 1 – прерывание по высокому уровню/переднему фронту.
11	PLR11	RWC	- PLR11 = 0 – прерывание по низкому уровню/заднему фронту; - PLR11 = 1 – прерывание по высокому уровню/переднему фронту.
10	PLR10	RWC	- PLR10 = 0 – прерывание по низкому уровню/заднему фронту; - PLR10 = 1 – прерывание по высокому уровню/переднему фронту
9	PLR9	RWC	- PLR9 = 0 – прерывание по низкому уровню/заднему фронту; - PLR9 = 1 – прерывание по высокому уровню/переднему фронту.
8	PLR8	RWC	- PLR8 = 0 – прерывание по низкому уровню/заднему фронту; - PLR8 = 1 – прерывание по высокому уровню/переднему фронту.
7	PLR7	RWC	- PLR7 = 0 – прерывание по низкому уровню/заднему фронту; - PLR7 = 1 – прерывание по высокому уровню/переднему фронту.
6	PLR6	RWC	- PLR6 = 0 – прерывание по низкому уровню/заднему фронту; - PLR6 = 1 – прерывание по высокому уровню/переднему фронту.
5	PLR5	RWC	- PLR5 = 0 – прерывание по низкому уровню/заднему фронту; - PLR5 = 1 – прерывание по высокому уровню/переднему фронту.
4	PLR4	RWC	- PLR4 = 0 – прерывание по низкому уровню/заднему фронту; - PLR4 = 1 – прерывание по высокому уровню/переднему фронту.
3	PLR3	RWC	- PLR3 = 0 – прерывание по низкому уровню/заднему фронту; - PLR3 = 1 – прерывание по высокому уровню/переднему фронту.
2	PLR2	RWC	- PLR2 = 0 – прерывание по низкому уровню/заднему фронту; - PLR2 = 1 – прерывание по высокому уровню/переднему фронту.
1	PLR1	RWC	- PLR1 = 0 – прерывание по низкому уровню/заднему фронту; - PLR1 = 1 – прерывание по высокому уровню/переднему фронту.
0	PLR0	RWC	- PLR0 = 0 – прерывание по низкому уровню/заднему фронту; - PLR0 = 1 – прерывание по высокому уровню/переднему фронту.
Примечание – Принятые условные обозначения: R –чтение, W – запись, C – сброс.			

На рисунке 16.7 приведена структура регистра выбора типа (уровень, фронт) входного сигнала генерации прерывания GPIO, а в таблице 16.8 описание его разрядов.

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
15	14	13	12	11	10	9	8
EDG15	EDG14	EDG13	EDG12	EDG11	EDG10	EDG9	EDG8
RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0
7	6	5	4	3	2	1	0
EDG7	EDG6	EDG5	EDG4	EDG3	EDG2	EDG1	EDG0
RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0

Примечание – Принятые условные обозначения: R – чтение, W – запись, C – бит сбрасывается, X – значение не определено.

Рисунок 16.7 – Структура регистра выбора типа сигнала (уровень, фронт) генерации прерывания GPIO INTEDGR***n*** /адрес **80_44F2**h – GPIO_0; адрес **80_4542**h – GPIO_1/

Таблица 16.8 – Описание битов регистра типа сигнала генерации прерывания

Бит	Обозначение	Тип	Описание
31 – 16	RSV	R	Зарезервировано.
15	EDG15	RWC	- EDG15 = 0 – прерывание генерируется по уровню; - EDG15 = 1 – прерывание генерируется по фронту.
14	EDG14	RWC	- EDG14 = 0 – прерывание генерируется по уровню; - EDG14 = 1 – прерывание генерируется по фронту.
13	EDG13	RWC	- EDG13 = 0 – прерывание генерируется по уровню; - EDG13 = 1 – прерывание генерируется по фронту.
12	EDG12	RWC	- EDG12 = 0 – прерывание генерируется по уровню; - EDG12 = 1 – прерывание генерируется по фронту.
11	EDG11	RWC	- EDG11 = 0 – прерывание генерируется по уровню; - EDG11 = 1 – прерывание генерируется по фронту.
10	EDG10	RWC	- EDG10 = 0 – прерывание генерируется по уровню; - EDG10 = 1 – прерывание генерируется по фронту.
9	EDG9	RWC	- EDG9 = 0 – прерывание генерируется по уровню; - EDG9 = 1 – прерывание генерируется по фронту.
8	EDG8	RWC	- EDG8 = 0 – прерывание генерируется по уровню; - EDG8 = 1 – прерывание генерируется по фронту.
7	EDG7	RWC	- EDG7 = 0 – прерывание генерируется по уровню; - EDG7 = 1 – прерывание генерируется по фронту.
6	EDG6	RWC	- EDG6 = 0 – прерывание генерируется по уровню; - EDG6 = 1 – прерывание генерируется по фронту.
5	EDG5	RWC	- EDG5 = 0 – прерывание генерируется по уровню; - EDG5 = 1 – прерывание генерируется по фронту.
4	EDG4	RWC	- EDG4 = 0 – прерывание генерируется по уровню; - EDG4 = 1 – прерывание генерируется по фронту.
3	EDG3	RWC	- EDG3 = 0 – прерывание генерируется по уровню; - EDG3 = 1 – прерывание генерируется по фронту.
2	EDG2	RWC	- EDG2 = 0 – прерывание генерируется по уровню; - EDG2 = 1 – прерывание генерируется по фронту.
1	EDG1	RWC	- EDG1 = 0 – прерывание генерируется по уровню; - EDG1 = 1 – прерывание генерируется по фронту.
0	EDG0	RWC	- EDG0 = 0 – прерывание генерируется по уровню; - EDG0 = 1 – прерывание генерируется по фронту.
Примечание – Принятые условные обозначения: R –чтение, W – запись, C – сброс.			

Независимо от того, какая из 16 ассоциированных с каждым из двух блоков GPIO индивидуально программируемых (фронт/уровень, полярность, маска) линий стала причиной инициации прерывания, вектор прерывания для каждого блока GPIO является одним и тем же: 1Dh для GPIO_0 и 1Eh для GPIO_1. Для определения, какой именно вывод инициировал прерывание от блока GPIO, служит регистр флагов прерываний.

На рисунке 16.8 приведена структура регистра флагов прерываний блока GPIO, а в таблице 16.9 – описание разрядов этого регистра.

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
15	14	13	12	11	10	9	8
INTF15	INTF14	INTF13	INTF12	INTF11	INTF10	INTF9	INTF8
RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0
7	6	5	4	3	2	1	0
INTF7	INTF6	INTF5	INTF4	INTF3	INTF2	INTF1	INTF0
RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0	RWC-0

Примечание – Принятые условные обозначения: R – чтение, W – запись, C – бит сбрасывается, X – значение не определено.

Рисунок 16.8 – Структура регистра флагов прерываний GPIO INTFR***n*** /адрес **80_4500**h – GPIO_0; адрес **80_4550**h – GPIO_1/

Детектирование вывода, инициировавшего прерывание для активации соответствующей (индивидуальной) подпрограммы его обработки удобно осуществлять, используя логическую функцию поразрядного Исключающего ИЛИ (XOR) между текущим и предыдущим содержимым регистра INTFR***n***.

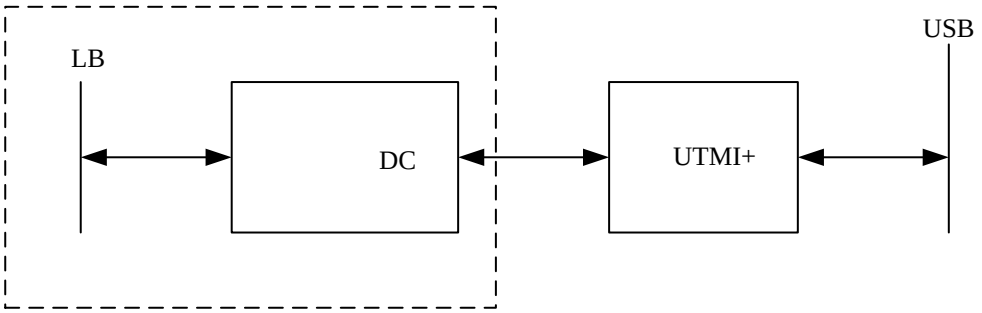
Таблица 16.9 – Описание битов регистра флагов прерываний

Бит	Обозначение	Тип	Описание
31 – 16	RSV	R	Зарезервировано.
15	INTF15	RWC	- INTF15 = 0 – не было прерывания по соответствующему выводу; - INTF15 = 1 – прерывание по соответствующему выводу произошло. Бит INTF15 очищается записью в него 1.
14	INTF14	RWC	- INTF14 = 0 – не было прерывания по соответствующему выводу; - INTF14 = 1 – прерывание по соответствующему выводу произошло. Бит INTF14 очищается записью в него 1.
13	INTF13	RWC	- INTF13 = 0 – не было прерывания по соответствующему выводу; - INTF13 = 1 – прерывание по соответствующему выводу произошло. Бит INTF13 очищается записью в него 1.
12	INTF12	RWC	- INTF12 = 0 – не было прерывания по соответствующему выводу; - INTF12 = 1 – прерывание по соответствующему выводу произошло. Бит INTF12 очищается записью в него 1.
11	INTF11	RWC	- INTF11 = 0 – не было прерывания по соответствующему выводу; - INTF11 = 1 – прерывание по соответствующему выводу произошло. Бит INTF11 очищается записью в него 1.
10	INTF10	RWC	- INTF10 = 0 – не было прерывания по соответствующему выводу; - INTF10 = 1 – прерывание по соответствующему выводу произошло. Бит INTF10 очищается записью в него 1.
9	INTF9	RWC	- INTF9 = 0 – не было прерывания по соответствующему выводу; - INTF9 = 1 – прерывание по соответствующему выводу произошло. Бит INTF9 очищается записью в него 1.
8	INTF8	RWC	- INTF8 = 0 – не было прерывания по соответствующему выводу; - INTF8 = 1 – прерывание по соответствующему выводу произошло. Бит INTF8 очищается записью в него 1.
7	INTF7	RWC	- INTF7 = 0 – не было прерывания по соответствующему выводу; - INTF7 = 1 – прерывание по соответствующему выводу произошло. Бит INTF7 очищается записью в него 1.
6	INTF6	RWC	- INTF6 = 0 – не было прерывания по соответствующему выводу; - INTF6 = 1 – прерывание по соответствующему выводу произошло. Бит INTF6 очищается записью в него 1.
5	INTF5	RWC	- INTF5 = 0 – не было прерывания по соответствующему выводу; - INTF5 = 1 – прерывание по соответствующему выводу произошло. Бит INTF5 очищается записью в него 1.
4	INTF4	RWC	- INTF4 = 0 – не было прерывания по соответствующему выводу; - INTF4 = 1 – прерывание по соответствующему выводу произошло. Бит INTF4 очищается записью в него 1.
3	INTF3	RWC	- INTF3 = 0 – не было прерывания по соответствующему выводу; - INTF3 = 1 – прерывание по соответствующему выводу произошло. Бит INTF3 очищается записью в него 1.
2	INTF2	RWC	- INTF2 = 0 – не было прерывания по соответствующему выводу; - INTF2 = 1 – прерывание по соответствующему выводу произошло. Бит INTF2 очищается записью в него 1.
1	INTF1	RWC	- INTF1 = 0 – не было прерывания по соответствующему выводу; - INTF1 = 1 – прерывание по соответствующему выводу произошло. Бит INTF1 очищается записью в него 1.
0	INTF0	RWC	- INTF0 = 0 – не было прерывания по соответствующему выводу; - INTF0 = 1 – прерывание по соответствующему выводу произошло. Бит INTF0 очищается записью в него 1.
Примечание – Принятые условные обозначения: R –чтение, W – запись, C – сброс.			

17 Периферийное устройство USB 2.0

Периферийное устройство Universal Serial Bus Device Controller (далее – USB 2.0 или DC контроллер) обеспечивает интерфейс между процессором (ПЦОС_0 или ПЦОС_1) и универсальной последовательной шиной (Universal Serial Bus). USB 2.0 поддерживает на протокольном уровне стандарт высокоскоростного – 480 Мбит/с High Speed (HS) и полноскоростного – 12 Мбит/с Full Speed (FS) обмена данными в режиме управляющих передач, в режиме передачи массивов данных, в режиме передачи по прерываниям или в изохронном режиме.

DC контроллер должен соединяться с шиной через внешнюю микросхему (далее – PHY), обеспечивающую физический уровень доступа к шине, см. рисунок 17.1, которая должна быть совместима с интерфейсом UTMI+.



Примечание – Принятые условные обозначения:
- LB – локальная шина;
- DC – Device контроллер;
- UTMI+ – интерфейс с микросхемой физического уровня;
- USB – универсальная последовательная шина.

Рисунок 17.1 – Подключение DC контроллера к внешнему интерфейсу

Для подключения к внешнему физическому интерфейсу (PHY) устройство USB 2.0 управляет интерфейсом UTMI+ для восьмибитного режима обмена данными. Тактовая частота интерфейса UTMI+ должна быть равной 60 МГц для обеспечения 480 Мбит/с (60 Мбайт/с) скорости передачи данных.

Периферийное устройство USB 2.0 поддерживает четыре конечные точки (End Point), которые программируются через регистры конечных точек EPn_CSR, EPn_INT, EPn_OBA, EPn_IBA. Каждая конечная точка имеет свой уникальный номер.

Каждая из точек может быть определена как точка типа IN, OUT или CONTROL для осуществления управляющих передач, передач массивов данных, передач по прерываниям или изохронных передач. Конечная точка типа CONTROL использует два буфера EPn_IBA и EPn_OBA, конечные точки IN и OUT используют один тип буфера.

Регистры EPn_IBA и EPn_OBA позволяют задать буфер с данными для отправки хосту и приёма данных от хоста соответственно. В них же задаётся размер буфера. Каждый буфер размещается в диапазоне адресов 80_6000h – 80_6FFFh. Таким образом, максимальный размер всех буферов составляет 4K 32-разрядных слов.

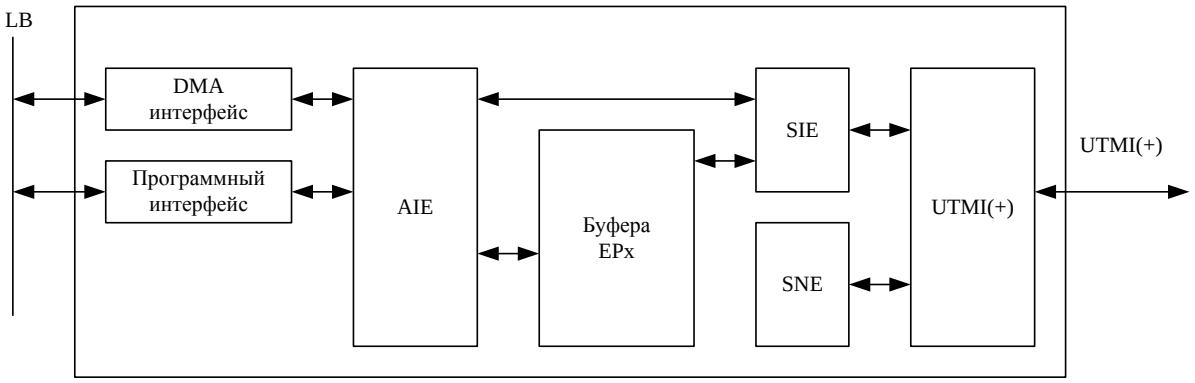
Линия прерывания от USB 2.0 коммутируется к ядрам ПЦОС_0 или ПЦОС_1. Далее приведена внутренняя структура USB 2.0 (рисунок 17.2) и коротко описаны функции его основных блоков:

- блок детектирования подключения к USB шине;
- блок последовательного интерфейса;
- блок DMA.

Блок детектирования подключения к USB шине (далее – Speed Negotiation Engine или SNE) обнаруживает подключение к шине VBus путём опроса сигнала VBusValid на разъёме.

Когда напряжение достигнет 5 В, уровень сигнала VBusValid будет равен логической 1. Блок SNE ждёт сигнал RESET от HOST (устройство, управляющее USB шиной) и начинает обмен данными на скорости HS (480 Мбит/с). Когда процедура сброса и первоначального обмена заканчивается, то выбирается скорость обмена данными FS (12 Мбит/с) или HS (480 Мбит/с). Об этом сообщается блоку последовательного интерфейса (далее – Serial Interface Engine или SIE), и могут начинаться операции приёма/передачи. Блок SNE также может определять и управлять состояниями приостановки (SUSPEND) и восстановления (RESUME) работы USB 2.0.

Блок SIE ждёт прихода пакетов и обрабатывает их в соответствии со спецификацией USB 2.0. Данные записываются во внутренний буфер, принадлежащий конкретной конечной точке EPx. Блок DMA, используя дескрипторы, передаёт данные из внутреннего буфера в память, которая доступна процессору. Когда необходимо передать пакет, пользовательская программа должна через регистры запрограммировать EPx и инициировать DMA передачу.



Примечание – Принятые условные обозначения:
- SNE – машина, реализующая протокол USB 2.0;
- SIE – машина последовательного канала USB 2.0;
- UTMI+ – интерфейс с микросхемой физического уровня;
- EPx – конечная точка с номером x = 0, 1, 2, 3;
- AIE – машина, реализующая протокол процессорной шины LB.

Рисунок 17.2 – Структурная схема DC контроллера

17.1 Конечные точки ENDPOINT

Конечная точка – это часть USB-устройства, которая имеет уникальный идентификатор, является получателем или отправителем информации, передаваемой по шине USB. Конфигурация процессорной шины включает операции DMA, описанные в последующих разделах. Конфигурация USB включает: разрешение ENDPOINT для USB транзакции, установку типа передачи (CONTROL, BULK, ISOCHRONOUS, INTERRUPT), максимальный объём данных для обеспечения высокой пропускной способности. Конфигурирование выполняется через регистры доступные со стороны процессора через программный интерфейс.

Когда выбор конфигурации выполнен, должен быть установлен бит готовности ENDPOINT. Это позволит разрешить передачи данных. Если ENDPOINT реконфигурируется, то бит готовности должен быть первым установлен в 0, иначе ENDPOINT будет инициализирован некорректно. Когда ENDPOINT разрешается, то схема сбрасывается, буфер очищается, и селектор буфера устанавливается в 0. Поле максимальной загрузки, количество дополнительных транзакций и тип обмена могут быть изменены только тогда, когда бит разрешения ENDPOINT установлен в 0. Другие биты регистра управления ENDPOINT могут быть изменены в любое время.

Конфигурация ENDPOINT не должна меняться, когда бит разрешения ENDPOINT установлен в 1 за исключением состояний «ОСТАНОВ», «УПРАВЛЕНИЕ ОСТАНОВОМ» и запрещения ENDPOINT. ENDPOINT может быть также остановлен путём установки бита HALT в 1. После этого все транзакции получат в ответ STALL HANDSHAKE. Когда режим HALT прекращает действовать, схема сбрасывается в соответствие со стандартом USB.

Когда ENDPOINT будет переведён в состояние «ГОТОВ», то передача данных из и в ENDPOINT могут быть возобновлены. Не существует различий между передачами данных по процессорной шине в зависимости от выбранного типа передач. Для управления ENDPOINT некоторое дополнительное управление потребуется во время обработки ошибок.

При получении пакета (независимо от типа ENDPOINT) с большим числом байт, чем сконфигурированное максимально допустимое значение для ENDPOINT, будет вызывать STALL HANDSHAKE и режим HALT ENDPOINT.

Когда детектируется сброс USB, то биты CS, ED регистра управления ENDPOINT будут сбрасываться для всех ENDPOINT. Бит EV будет также очищаться для всех ENDPOINT кроме управляющей ENDPOINT (CTRL EP).

Бит CB в управляющем регистре ENDPOINT очищает содержимое внутренних буферов ENDPOINT. При его установке в 1 данные в буферах будут потеряны, биты готовности данных для соответствующих буферов будут очищаться, и бит CB также установится в ноль. Если есть активная транзакция для этой EP, то ничего перечисленного не произойдёт, поэтому данная функция должна использоваться с осторожностью.

17.1.1 Конечные точки OUT ENDPOINT

Операции DMA для OUT ENDPOINT в целом соответствуют описанию выше. Имеются небольшие отличия в индивидуальных битах и значениях полей. Описание слов дескрипторов приведено в таблицах ниже.

После разрешения дескриптора он выбирается аппаратурой DC контроллера. Как только буфер для соответствующего ENDPOINT будет содержать данные, полученные от USB, эти данные будут записаны в память, начиная с адреса, указанного в слове указателя буфера дескриптора. Содержимое одного внутреннего буфера памяти всегда записывается в буфер одного дескриптора. Оно обычно соответствует одному USB пакету, за исключением конечных точек типа ISOHRONOUS EPx (изохронные передачи) и INTRRUPT EPx (передачи по прерыванию).

После завершения записи данных в буфер, число записанных байт сохраняется в поле длины. Это поле содержит верные данные, если бит разрешения EPx равен нулю. После сброса бита готовности, пространство памяти может быть использовано снова.

Прерывания генерируются при условии завершения записи в память. ENDPOINT может также быть сконфигурирована для генерации прерывания немедленно, когда пакет был принят во внутренний буфер.

Это прерывание разрешается в регистре управления ENDPOINT. Когда данные выбираются из внутренней памяти, бит сбрасывается и может быть использован снова SIE для приёма нового пакета. На рисунках 17.3, 17.4 и 17.5 приведена структура слов OUT дескриптора, а в таблицах 17.1, 17.2 и 17.3 – приведено их описание.

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R	R	R	R	R	R	R	R
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	SE	RSV
R	R	R	R	R	R	RW	RW
15	14	13	12	11	10	9	8
IE	NX	EN	LENGTH12	LENGTH11	LENGTH10	LENGTH9	LENGTH8
RW	RW	RW	RW	RW	RW	RW	RW
7	6	5	4	3	2	1	0
LENGTH7	LENGTH6	LENGTH5	LENGTH4	LENGTH3	LENGTH2	LENGTH1	LENGTH0
RW	RW	RW	RW	RW	RW	RW	RW

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

Рисунок 17.3 – Структура слова 0 OUT дескриптора – управляющее слово

Таблица 17.1 – Описание битов слова 0 OUT дескриптора

Бит	Обозначение	Тип	Описание
31 – 18	RSV	R	Зарезервировано
17	SE	RW	Пакет SETUP. Данные были получены из пакета SETUP вместо пакета OUT.
16	RSV	RW	Зарезервировано.
15	IE	RW	Разрешение прерывания. Прерывание будет генерироваться, когда пакет от этого дескриптора был принят во внутренний буфер и передан SIE. Это не значит, что пакет был передан.
14	NX	RW	Следующий дескриптор готов. Поле следующего дескриптора действительно и указывает на следующий дескриптор.
13	EN	RW	Бит разрешения дескриптора. Установка этого бита в 1 разрешает использовать дескриптор. Он должен быть всегда установлен последним из всех полей дескриптора.
12 – 0	LENGTH12 – LENGTH0	RW	Число полученных байт. Действителен, когда бит EN сброшен DC контроллером.
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

31	30	29	28	27	26	25	24
ADDRESS 29	ADDRESS 28	ADDRESS 27	ADDRESS 26	ADDRESS 25	ADDRESS 24	ADDRESS 23	ADDRESS 22
RW	RW	RW	RW	RW	RW	RW	RW
23	22	21	20	19	18	17	16
ADDRESS 21	ADDRESS 20	ADDRESS 19	ADDRESS 18	ADDRESS 17	ADDRESS 16	ADDRESS 15	ADDRESS 14
RW	RW	RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8
ADDRESS 13	ADDRESS 12	ADDRESS 11	ADDRESS 10	ADDRESS 9	ADDRESS 8	ADDRESS 7	ADDRESS 6
RW	RW	RW	RW	RW	RW	RW	RW
7	6	5	4	3	2	1	0
ADDRESS 5	ADDRESS 4	ADDRESS 3	ADDRESS 2	ADDRESS 1	ADDRESS 0	RSV	RSV
RW	RW	RW	RW	RW	RW	R	R

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

Рисунок 17.4 – Структура слова 1 OUT дескриптора

Таблица 17.2 – Описание битов слова 1 OUT дескриптора

Бит	Обозначение	Тип	Описание
31 – 2	ADDRESS29 – ADDRESS0	RW	Указатель на ячейку памяти, куда будет загружаться принимаемый пакет данных
1 – 0	RSV	R	Зарезервировано
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

31	30	29	28	27	26	25	24
NDP29	NDP28	NDP27	NDP26	NDP25	NDP24	NDP23	NDP22
RW	RW	RW	RW	RW	RW	RW	RW
23	22	21	20	19	18	17	16
NDP21	NDP20	NDP19	NDP18	NDP17	NDP16	NDP15	NDP14
RW	RW	RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8
NDP13	NDP12	NDP11	NDP10	NDP9	NDP8	NDP7	NDP6
RW	RW	RW	RW	RW	RW	RW	RW
7	6	5	4	3	2	1	0
NDP5	NDP4	NDP3	NDP2	NDP1	NDP0	RSV	RSV
RW	RW	RW	RW	RW	RW	R	R

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

Рисунок 17.5 – Структура слова 2 OUT дескриптора (смещение адреса 0x8).
Указатель на следующий дескриптор

Таблица 17.3 – Описание битов слова 2 OUT дескриптора

Бит	Обозначение	Тип	Описание
31 – 2	NDP29 – NDP0	RW	Указатель на следующий дескриптор
1 – 0	RSV	R	Зарезервировано
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

17.1.2 Конечные точки IN ENDPOINT

Операции DMA в IN ENDPOINT соответствуют описанным выше основным операциям DMA. Имеются небольшие различия в индивидуальных битах и значениях полей.

На рисунках 17.6, 17.7 и 17.8 приведена структура слов IN дескриптора, а в таблицах 17.4, 17.5 и 17.6 – приведено описание их разрядов.

DC контроллер начнёт обработку дескриптора, когда установлен в 1 бит разрешения дескриптора. Как только буфер будет готов, DC контроллер заносит во внутренний буфер число байт, указанное в поле длины пакета данной ENDPOINT. Соответствующее прерывание будет сгенерировано (если оно было разрешено), когда данные были записаны во внутренний буфер и статус был обновлён в дескрипторе. При этом пакет может быть ещё не передан на USB шину.

Отдельное прерывание может быть сгенерировано, когда пакет действительно был передан. Для этого нужно установить в регистре управления ENDPOINT соответствующий бит, а также в дескрипторе установить бит PI для каждого пакета.

Для дескриптора с нулевой длиной, как следствие, будет передан пакет нулевой длины. Если длина больше, чем максимально возможное значение для ENDPOINT, то будет передано два и более пакета. Размер последней передачи данных может быть равным или меньше максимально возможного значения, указанного в регистре управления ENDPOINT. Если это значение превышает размер буфера, то данные не будут записаны в буфер, и в статусном бите дескриптора будет записан код ошибки. Когда установлен бит MORE, данные текущего дескриптора будут записаны в буфер. После этого DC контроллер продолжит обработку следующего дескриптора без разрешения отправки данных из буфера. Данные следующего дескриптора читаются из того же буфера и это будет продолжаться до тех пор, пока не встретится дескриптор с битом MORE = 0.

Если общее количество байт становится большим, чем размер внутреннего буфера, то пакет не отсылается (данные из внутреннего буфера теряются) и бит ML устанавливается в 1 в последнем дескрипторе. После этого выборка дескриптора начинается снова.

Если бит NEXT не установлен, когда установлен бит MORE, то DC контроллер будет ждать дескриптор без передачи доступа к процессорной шине другим конечным точкам.

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R	R	R	R	R	R	R	R
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	MO	PI	ML
R	R	R	R	R	RW	RW	RW
15	14	13	12	11	10	9	8
IE	NX	EN	LENGTH 12	LENGTH 11	LENGTH 10	LENGTH 9	LENGTH 8
RW	RW	RW	RW	RW	RW	RW	RW
7	6	5	4	3	2	1	0
LENGTH 7	LENGTH 6	LENGTH 5	LENGTH 4	LENGTH 3	LENGTH 2	LENGTH 1	LENGTH 0
RW	RW	RW	RW	RW	RW	RW	RW

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

Рисунок 17.6 – Структура слова 0 IN дескриптора (смещение адреса 0x0) – управляющее слово

Таблица 17.4 – Описание битов слова 0 IN дескриптора

Бит	Обозначение	Тип	Описание
31 – 19	RSV	R	Зарезервировано
18	MO	RW	MORE. Данные следующего дескриптора должны читаться в тот же буфер.
17	PI	RW	Генерирует прерывание, когда пакет был передан на USB шину.
16	ML	RW	ML = 1, если была попытка передать данные, объём которых больше, чем размер буфера.
15	IE	RW	Разрешение прерывания. Прерывание будет генерироваться, когда пакет для этого дескриптора был прочитан во внутренний буфер и передан SIE. Но это не означает, что пакет был передан.
14	NX	RW	Поле следующего дескриптора. Когда NX = 1, то нужно обрабатывать следующий дескриптор.
13	EN	RW	Бит разрешения дескриптора. Устанавливается в 1 для разрешения дескриптора. Этот бит всегда устанавливается последним из всех полей дескриптора.
12 – 0	LENGTH12 – LENGTH0	RW	Длина. Число байт, которые должны быть переданы.
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

31	30	29	28	27	26	25	24
ADDRESS 29	ADDRESS 28	ADDRESS 27	ADDRESS 26	ADDRESS 25	ADDRESS 24	ADDRESS 23	ADDRESS 22
RW	RW	RW	RW	RW	RW	RW	RW
23	22	21	20	19	18	17	16
ADDRESS 21	ADDRESS 20	ADDRESS 19	ADDRESS 18	ADDRESS 17	ADDRESS 16	ADDRESS 15	ADDRESS 14
RW	RW	RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8
ADDRESS 13	ADDRESS 12	ADDRESS 11	ADDRESS 10	ADDRESS 9	ADDRESS 8	ADDRESS 7	ADDRESS 6
RW	RW	RW	RW	RW	RW	RW	RW
7	6	5	4	3	2	1	0
ADDRESS 5	ADDRESS 4	ADDRESS 3	ADDRESS 2	ADDRESS 1	ADDRESS 0	RSV	RSV
RW	RW	RW	RW	RW	RW	R	R

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

Рисунок 17.7 – Структура слова 1 IN дескриптора – указатель на буфер данных (смещение адреса 0x4)

Таблица 17.5 – Описание битов слова 1 IN дескриптора

Бит	Обозначение	Тип	Описание
31 – 2	ADDRESS29 – ADDRESS0	RW	Указатель на область буфера, откуда должны загружаться данные пакета
1 – 0	RSV	R	Зарезервировано
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

31	30	29	28	27	26	25	24
NDP29	NDP28	NDP27	NDP26	NDP25	NDP24	NDP23	NDP22
RW	RW	RW	RW	RW	RW	RW	RW
23	22	21	20	19	18	17	16
NDP21	NDP20	NDP19	NDP18	NDP17	NDP16	NDP15	NDP14
RW	RW	RW	RW	RW	RW	RW	RW
15	14	13	12	11	10	9	8
NDP13	NDP12	NDP11	NDP10	NDP9	NDP8	NDP7	NDP6
RW	RW	RW	RW	RW	RW	RW	RW
7	6	5	4	3	2	1	0
NDP5	NDP4	NDP3	NDP2	NDP1	NDP0	RSV	RSV
RW	RW	RW	RW	RW	RW	R	R

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

Рисунок 17.8 – Структура слова 2 IN дескриптора (смещение адреса 0x8) – указатель на следующий дескриптор

Таблица 17.6 – Описание битов слова 2 IN дескриптора

Бит	Обозначение	Тип	Описание
31 – 2	NDP29 – NDP0	RW	Указатель на следующий дескриптор
1 – 0	RSV	R	Зарезервировано
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

17.1.3 Управляющие конечные точки CTRL EP

В соответствии с USB стандартом ENDPOINT с номером 0 (CTRL EP0) всегда является управляющей конечной точкой и должна быть доступной для взаимодействия с хостом и осуществления RESET USB. DC контроллер не подтверждает любую транзакцию к нему до тех пор, пока сброс USB не будет получен. Таким образом, CTRL EP должна быть разрешена после системного сброса. Другие управляющие точки могут быть разрешены с теми же ограничениями, что и обязательная CTRL EP0, только после конфигурирования. Программа, выполняемая процессором 1867ВН016, может не успеть сконфигурировать CTRL EP0 должным образом перед процедурой сброса USB. Эта проблема устраняется в DC контроллере. Его резистор PULL_UP на линии D+ запрещается после системного сброса, что даёт полный функциональный контроль до того момента, пока DC контроллер не будет виден на USB шине.

CTRL EP0 является потоковой для сообщений, и, следовательно, передача данных может осуществляться в направлении IN и OUT. Поэтому управляющие конечные точки должны использоваться в обоих направлениях с одним и тем же номером в DC контроллере. Таким образом, обе управляющие EPx должны быть сконфигурированы в одном режиме (один и тот же тип передачи, объём принимаемых данных и т. п.). Иначе поведение DC контроллера будет не определено.

Управляющая передача всегда начинается с SETUP транзакции, которую получает OUT EP. Если управляющая передача является записью в фазе DATA и осуществляется в OUT направлении, то эти данные также будут получены в OUT EPx. Программа для процессора 1867ВН016 должна читать данные конфигурационных запросов и отвечать соответственно. Если запрос был корректным, то программа должна разрешить ответный пакет для IN EP с соответствующими данными. В случае обнаружения ошибки необходимо остановить ENDPOINT. Для этого существуют два альтернативных способа: «Несбрасываемый» HALT, который будет сохраняться при получении следующего пакета SETUP и «сбрасываемый» HALT, который будет удаляться при получении следующего пакета SETUP. Ниже приведено рекомендуемое USB стандартом поведение для такого случая. Другие устройства будут требовать сброса DC контроллера для продолжения операции, если «несбрасываемый» HALT появляется на CTRL EP0.

DC контроллер устанавливает режим HALT автоматически, когда детектирует ошибку в одиночных транзакциях для конкретной EPx. Если запрос SETUP требует считать данные, то в фазе DATA будет направление IN. В этом случае пользователь DC контроллера должен разрешить отправку данных для EPx в IN направлении, если принимается такой запрос, в противном случае должен быть установлен режим HALT. Передача заканчивается, когда HOST посылает пакет нулевой длины в OUT EP.

Каждый раз, когда CTRL EP0 получает SETUP маркер, буферы в IN направлении опустошаются. Это делается для предотвращения ситуации, когда фаза DATA и фаза STATUS отсутствовали или не корректны и, таким образом, данные не могут быть извлечены. Старые данные будут в буфере, и следующая SETUP транзакция получит ошибочные данные. Стандарт USB утверждает, что это может произойти во время состояния ошибки, когда новый

пакет SETUP передаётся до окончания предыдущей передачи. Пользователь DC контроллера может очистить буфер через регистр управления IN EP или попытаться это сделать при обнаружении нового пакета SETUP до окончания предыдущей передачи. Это должно быть сделано, так как пользователь может включить буферы после того, как сброшен DC контроллер при получении нового пакета SETUP.

Источник полученных данных для дескриптора OUT EP (SETUP транзакция или OUT транзакция) указывается в статусном бите дескриптора.

17.1.4 Конечные точки передачи массивов данных BULK

Конечные точки в режиме BULK (BULK EPx) являются потоковыми магистралями и, следовательно, используются только в одном направлении (или IN или OUT). EPx с одним номером могут использоваться независимо.

17.1.5 Конечные точки передачи данных по прерываниям INTERRUPT

Данные от конечных точек, находящиеся в режиме INTERRUPT, обрабатываются процессором таким же образом, как и в режиме BULK. Отличия появляются на шине USB.

ENDPOINT в этом режиме также является потоковой конечной точкой, поддерживает широкополосный (с высокой пропускной способностью) статус, который означает, что выполняется одна транзакция на микрофрейм. Это требует большого размера буфера. Конечная точка должна конфигурироваться с буфером большим или эквивалентным максимальному количеству байт в транзакции. Все транзакции будут использовать один буфер для приёма и передач. Конечные точки конфигурируются как широкополосные EP установкой ненулевого количества дополнительных транзакций в управляющем регистре ENDPOINT.

17.1.6 Конечные точки в режиме ISOCHRONOUS

Конечные точки в режиме ISOCHRONOUS также являются потоковыми и передают данные по процессорной шине таким же образом, как и BULK EP и INTERRUPT EP. Между ISOCHRONOUS EP и другими типами конечных точек имеется существенное различие: в этом режиме отсутствует HANDSHAKE.

Если нет готовых данных для передачи, когда поступает IN маркер, то ISOCHRONOUS EP передаёт пакет нулевой длины. Это указывает хосту на то, что данные не готовы, и не произошла ошибка. Если не посылается пакет, то HOST не будет знать о причине: был ли повреждён пакет или нет. Когда EP работает не в высокоскоростном режиме, то только одна транзакция в OUT направлении будет сохраняться в буфере. В высокоскоростном режиме все транзакции во время микрофрейма будут сохранены в одном буфере. В IN направлении данные всегда передаются из того же буфера, в котором находятся и OUT данные. Для высокоскоростных EPx буфер должен конфигурироваться для количества транзакций с максимальной нагрузкой. Высокоскоростные ISOCHRONOUS EP используют последовательность PID. Когда возникает ошибка в PID последовательности в OUT направлении, то по процессорной шине передаются неверные данные для всего микрофрейма.

17.1.7 Буферы конечных точек

Состояние буферов влияет на посылку HANDSHAKE к HOST в конце транзакции. В HS режиме конечные точки BULK OUT и CONTROL OUT EPx, которые не находятся в фазе конфигурирования поддерживают PING протокол. Это значит, что в конце передачи данных одному из этих EPx DC контроллер должен вернуть ACK, если принял текущие данные и есть место для следующего пакета. Это происходит, если второй буфер конечной точки EPx пуст по окончании транзакции.

Если второй буфер не пустой, то DC контроллер посылает NYET. Если текущие данные не принимаются (оба буфера не пусты, когда приходит пакет), то возвращается NAK.

Для других типов EPx в режиме HS и всех EPx в режиме FS ACK всегда возвращается, если данные приняты, и NAK, если данные не приняты.

Размер EPx буфера может быть задан большим, чем значение MAXPL для EPx. Для IN EPx при чтении данных большего размера, чем максимальное значение размера буфера, из буфера будет прочитан пакет, который равен максимальному размеру пакета или пакет с меньшим размером, чем максимальный.

Большие буферы используются только для OUT транзакций или широкополосных ENDPOINT, где более одной транзакции на микрофрейм может передаваться для этой EPx. В этом случае данные из всех пакетов одного микрофрейма запоминаются в одном буфере в том порядке, в котором они поступают и после этого передаются на процессорную шину.

Все не широкополосные (высокоскоростные) ENDPOINT всегда хранят один пакет данных в буфере. Буферы ENDPOINT в DC контроллере не разделены на отдельные физические блоки, вместо этого они располагаются последовательно в пространстве памяти.

17.2 Интерфейс с микросхемой физического уровня

DC контроллер соединён с микросхемой физического уровня (далее – PHY) через интерфейс UTMI+. В таблице 17.7 приведено функциональное описание сигналов UTMI+ интерфейса.

Таблица 17.7 – Функциональное описание сигналов UTMI+ интерфейса

Обозначение сигнала	Описание	Тип
1	2	3
SuspendM	Установка трансивера в режим ожидания	O
Xcvr_Select	Выбор режима трансивера: - 0 – Full speed; - 0 – High speed	O
Term_Select	Разрешение окончания режима: - 1 – окончание режима Full speed; - 0 – окончание режима High speed	O
LineState_1	Сигнал 1 состояния линии	I
LineState_0	Сигнал 0 состояния линии	I
Op_Mode_1	Сигнал 1 выбора режима работы	O
Op_Mode_0	Сигнал 0 выбора режима работы	O

Окончание таблицы 17.7

1	2	3
DataOut7	Выходные данные, бит 7	O
DataOut6	Выходные данные, бит 6	O
DataOut5	Выходные данные, бит 5	O
DataOut4	Выходные данные, бит 4	O
DataOut3	Выходные данные, бит 3	O
DataOut2	Выходные данные, бит 2	O
DataOut1	Выходные данные, бит 1	O
DataOut0	Выходные данные, бит 0	O
TXValid	Передаваемые данные правильные	O
TXReady	Готовность передачи	I
DataIn7	Входные данные, бит 7	I
DataIn6	Входные данные, бит 6	I
DataIn5	Входные данные, бит 5	I
DataIn4	Входные данные, бит 4	I
DataIn3	Входные данные, бит 3	I
DataIn2	Входные данные, бит 2	I
DataIn1	Входные данные, бит 1	I
DataIn0	Входные данные, бит 0	I
RXValid	Принимаемые данные правильные	I
RXActive	Готовность приёма данных	I
RXError	Ошибка приёма	I
CLK_Phy	Тактовый сигнал от трансивера	I
Reset_phy	Сигнал сброса трансивера	O
VBusValid	Сигнал готовности USB шины	I
Примечание – Принятые условные обозначения: I – вход, O – выход.		

17.3 Блок SNE

Блок SNE детектирует подключение, обеспечивает управление сбросом, высокоскоростным (High Speed) HANDSHAKE (HS OTBET) и операциями SUSPEND/RESUME. Он также поддерживает различные тестовые режимы, стандартные для всех USB устройств.

Состояние подключения обнаруживается, когда сигнал VBusValid становится активным (высокий уровень). После этого DC контроллер ждёт сброса от HOST и начинает процедуру HS HANDSHAKE, после которой определяется режим работы HS или FS. DC контроллер не будет осуществлять передачи данных, и не будет подтверждать пакеты, отправленные ему, пока не выполнится процедура сброса.

DC контроллер поддерживает программное подключение/отключение. Это означает, что резистор PULL UP на линии D+ может управляться из доступного пользователю регистра DC контроллера. PULL UP резистор запрещается после сброса, и поэтому процедура подключения осуществляется, когда DC контроллер виден HOST.

SNE продолжает опрашивать условие, когда устанавливается состояние SUSPEND (3 мс приостановки IDLE на USB шине). Состояние SUSPEND отменяется через сброс USB или сигналом возобновления (resume).

Сигнал возобновления (resume) может идти или от порта (HUB узла или HOST контроллера) или от самого устройства «УДАЛЁННОЕ ВОССТАНОВЛЕНИЕ» (REMOUTE WAKEUP). DC контроллер может генерировать сигнал REMOUTE WAKEUP, который программируется через регистр, доступный пользователю. В возвращаемом дескрипторе на запрос GetStatus к DC контроллеру должно быть указано о поддержке функции удалённого восстановления. Транзакции управляются SIE только в том случае, если SNE находится в режиме FS или HS.

Все транзакции будут отклонены, пока DC контроллер находится в состояниях: SUSPEND, не подключено, подключено или во время процесса RESET. Все текущие состояния SNE: VBusValid, активный SUSPEND, USB RESET и текущий режим скорости могут быть определены через регистр состояния DC контроллера. Каждый из статусных битов, соответствующий одному из состояний DC контроллера, имеет бит разрешения прерывания. С помощью него можно разрешить прерывание при изменении конкретного статусного бита. Если используется только режим FS, то желательно не устанавливать выполнение HS HANDSHAKE через регистр.

Различные тестовые режимы, требуемые стандартом USB – Test_SE0_NAK, Test_J, Test_K и Test_Packet, также доступны через пользовательские регистры:

- Test_SE0_NAK HS. В этом режиме приёмник включается и отвечает NAK только на верные IN транзакции (правильный CRC, адрес DC и номер ENDPOINT совпадают, PID не повреждён).
- Test_J постоянно управляет состоянием HS J.
- Test_K постоянно управляет состоянием HS K.
- Test_Packet повторно посылает тестовые пакеты. Смотрите стандарт USB 2.0 о содержании пакета. Минимальная задержка между пакетами, когда DC контроллер посылает два или больше последовательных пакетов, которые не специфицируются стандартом. DC контроллер использует 192-битное время, как минимальную задержку, которая равна максимальному значению различных минимальных задержек в стандарте для любой последовательности пакетов и должна быть совместима с требованиями USB 2.0 стандарта.

17.4 Блок SIE

Блок SIE управляет приёмом и передачей USB пакетов. DC не будет отвечать для любой транзакции, пока не получен RESET, либо устанавливается режим FS или HS.

SIE всегда ждёт маркер (далее – TOKEN) пакета. В зависимости от типа TOKEN данные либо передаются из DC контроллера, либо наоборот принимаются в DC контроллер.

Специальные TOKEN передаются без каких-либо данных. Специальные TOKEN, такие как PING и SOF, вызывают отправку HANDSHAKE или номер фрейма. IN TOKEN инициируют передачу данных от DC контроллера к HOST, в то время как SETUP TOKEN и OUT TOKEN передают данные от HOST к DC контроллеру. Полученные пакеты с другим PID отбрасываются на этапе приёма TOKEN.

Пакеты данных передаются после того, как на этапе TOKEN определено, что данные должны передаваться от DC контроллера к HOST. Когда передача данных завершена, DC контроллер ждёт HANDSHAKE перед возвратом к этапу TOKEN.

Если на этапе TOKEN определено, что данные должны передаваться от HOST к DC контроллеру, то DC ждёт пакета с данными и после приёма пакета посылает HANDSHAKE перед возвратом в этап TOKEN.

17.5 Блок интерфейса с процессорной шиной AIE

Блок AIE (AHB Interface Engine) может конфигурироваться либо в режиме SLAVE (программный режим), либо в режиме MASTER (режим прямого доступа к памяти). Оба режима не могут присутствовать в одно и то же время. Оба интерфейса описаны далее.

17.5.1 MASTER режим

В режиме MASTER DC контроллер управляет процессорной шиной, и все данные передаёт и принимает в буферы EPx, используя DMA операции. Существуют две отдельные DMA машины для IN и OUT направлений. Они мультиплексируются в один мастер интерфейс на AHB шине.

Если оба DMA блока запрашивают шину, то она будет предоставляться обоим DMA по очереди. Если DMA OUT блок управляет процессорной шиной и после завершения обмена снова выставит запрос, но в это же время будет запрашивать процессорную шину DMA IN блок, то арбитр предоставит процессорную шину DMA IN блоку. DMA IN блок только читает данные (обратите внимание, что это применимо только к фазе DATA, статусный дескриптор записывается) с шины и всегда выполняет передачу слов.

DMA IN блок записывает данные и выполняет передачу слов. Если началась передача не с начала слова, то будет передаваться байт, пока не будет достигнута граница слова. С этого времени передаются слова до тех пор, пока для передачи останется меньше, чем 4 байта.

Если остались байты для передачи, то выполняется последняя отправка. Запись байта всегда требует одного доступа к шине.

DC контроллер может работать только в режиме обратного порядка байт (big-endian), при котором младший адрес в слове является старшим байтом. Что соответствует битам 31–24. Первый байт, полученный по USB шине, будет запоминаться в старшем байте. В одном байте младший бит соответствует первому биту, переданному по USB шине.

17.5.2 Функциональный тестовый режим

Функциональный тестовый режим может быть разрешён в DC контроллере, используя доступный программно бит FT в регистре управления. Функциональный тестовый режим предназначен для уменьшения количества требуемых тест-векторов во время функционального тестирования DC контроллера на кристалле. Для нормальной работы DC контроллера, перед тем как начать USB транзакцию, требуется осуществить процедуру последовательного определения скорости приёма/передачи по шине USB. Поскольку определение скорости приёма/передачи по шине USB занимает относительно длительное время, то использование тест-режима DC контроллера даёт возможность сократить процедуру детектирования скорости. Тестовый режим может быть запрещён/разрешён с помощью бита FT в регистре управления.

17.5.3 Операции DMA

Операции DMA похожи для обоих направлений обмена данными. Различия описаны ниже.

Операции DMA основываются на связанном списке дескрипторов, расположенных в памяти. Каждый ENDPOINT имеет свой собственный связанный список дескрипторов. Первое слово в дескрипторе является управляющим словом, которое содержит бит разрешения, определяющий, является ли дескриптор активным или нет и содержит другие управляющие биты. Следующее слово является указателем на буфер, расположенный в памяти, в котором находятся данные для отправки или в который нужно писать данные. Последнее слово является указателем на первое слово следующего дескриптора. Один бит в управляющем слове определяет, актуален или нет следующий дескриптор. Если следующий дескриптор не актуален, то происходит останов после обработки текущего дескриптора, канал DMA останавливается.

Вначале следует записать список дескрипторов в память, после этого следует записать указатель на первый дескриптор в регистр указателя дескриптора ENDPOINT и установить бит разрешения. Регистр указателя модифицируется при переходе к следующему дескриптору и может читаться процессором. Список заканчивается дескриптором, который содержит бит разрешения, равный нулю. Список не должен меняться, пока DC контроллер не закончит обработку списка, и DMA канал не будет отключён. Иначе может произойти непредвиденная ситуация, и поведение будет не определено.

Другой путь использования связанного списка – это всегда устанавливать бит разрешения в следующем дескрипторе и при этом быть уверенным, что он равен нулю у последнего дескриптора. Можно добавлять новые дескрипторы в конец списка и разрешать их «на лету», при условии, что бит разрешения дескриптора всегда устанавливается после записи нового дескриптора в память. Таким образом, не произойдёт тупика, и новые дескрипторы не будут пропущены.

На рисунке 17.9 показана структура связанного списка дескрипторов.

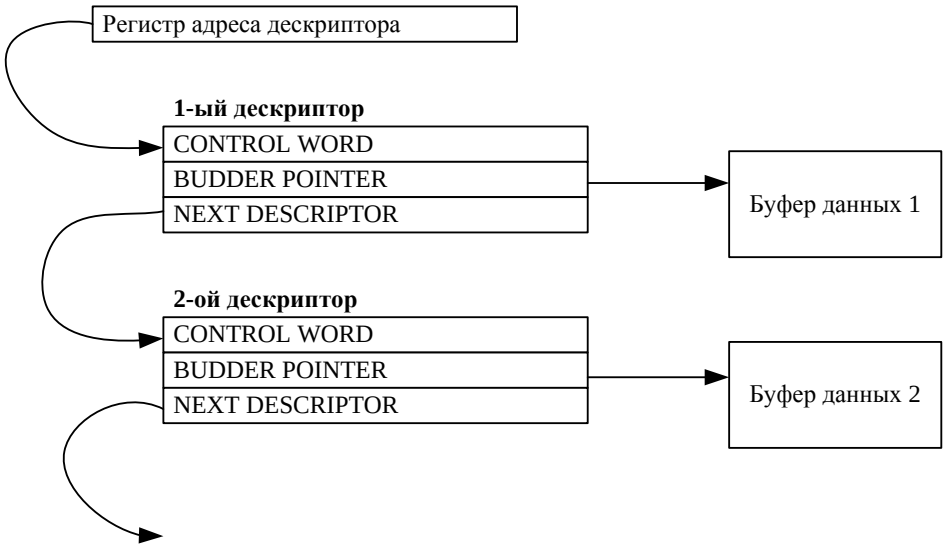


Рисунок 17.9 – Пример структуры связанного списка DMA дескрипторов

17.6 Пример реализации DC контроллера в режиме MASTER

В данном подразделе коротко описано, как DC контроллер может использоваться в режиме MASTER. Функциональное управление DC контроллером и реализация конкретного устройства для фактического применения требует управления DC контроллером через процессорную шину.

Первое, что нужно сделать – это правильно сконфигурировать PHY. Это автоматически делает DC контроллер. После этого DC контроллер, анализируя сигнал VBusValid, ждёт подключения к шине USB, которое фиксируется или опросом флага, или по прерыванию. Подключением можно управлять через бит EP (Pull-up Enable/Disable) регистра GLBLCTRLDC (регистр глобального управления DC контроллером, см. рисунок 17.19 и таблицу 17.17). В состоянии Disable Pull-up (запрет резистора) HOST «не замечает» DC контроллер, даже если тот вставлен в разъём.

После подключения и перед началом обмена данными требуется сброс USB от HOST, что также определяется опросом флага или прерыванием.

Только CTRL EP0 должен быть доступен после сброса. Процессор отвечает за включение и настройку DC контроллера в нужное время. Процессор может ждать, пока DC контроллер получит сброс USB, но лучше DC включить сразу после включения питания. Это можно сделать, поскольку DC контроллер не будет принимать никакую транзакцию, пока не будет получен сброс USB. При разрешении дескрипторов EPx должны быть также разрешены оба направления IN и OUT, и также должен быть установлен бит разрешения дескриптора. После этого ENDPOINT готов принимать пакеты, а процессор должен ждать прибытие пакетов SETUP. Он может уведомляться о полученных пакетах или через опрос флага или по прерыванию. DC контроллер должен обработать запросы и возвращаемые дескрипторы, как требуется USB 2.0 стандартом. Когда приходит запрос SET ADDRESS, программа DC контроллера должна записать новый адрес в регистр управления DC контроллером. Этот адрес начинает действовать после следующей успешной IN транзакции для CTRL EP. Это процедура должна соответствовать стадии установки адреса SET ADDRESS.

Когда получен запрос SET CONFIGURATION, программа DC контроллера должна включить соответствующую конфигурацию и EPx, соответствующие этой конфигурации. Это делается записью в различные регистры управления EPx. Программа отвечает за передачу дескрипторов конфигураций и интерфейсов, требуемых SETUP транзакциями. При разрешении программой EPx, должны быть также разрешены операции DMA. После этого DC контроллер готов для передачи и приёма данных через конечные точки, определённые приложением. Прерывания могут использоваться для уведомления о новых переданных пакетах. Через опрос флагов можно будет определить источник прерывания.

17.7 Описание регистров DC контроллера

В таблице 17.8 приведены регистры периферийного устройства USB 2.0.

Таблица 17.8 – Регистры DC контроллера

Адрес	Название	Функциональное назначение
Конечная точка OUT EP0		
80_4060h	CTRLOUTEP0	Регистр управления конечной точкой 0 OUT EP0
80_4064h	DMACTRLOUTEP0	Регистр управления DMA конечной точки 0 OUT EP0
80_4068h	DMADAOUTEP0	Регистр адреса дескриптора DMA конечной точки 0 OUT EP0
80_406Ch	STOUTEP0	Статусный регистр конечной точки 0 OUT EP0
Конечная точка OUT EP1		
80_4070h	CTRLOUTEP1	Регистр управления конечной точкой 1 OUT EP1
80_4074h	DMACTRLOUTEP1	Регистр управления DMA конечной точки 1 OUT EP1
80_4078h	DMADAOUTEP1	Регистр адреса дескриптора DMA конечной точки 1 OUT EP1
80_407Ch	STOUTEP1	Статусный регистр конечной точки 1 OUT EP1
Конечная точка OUT EP2		
80_4080h	CTRLOUTEP2	Регистр управления конечной точкой 2 OUT EP2
80_4084h	DMACTRLOUTEP2	Регистр управления DMA конечной точки 2 OUT EP2
80_4088h	DMADAOUTEP2	Регистр адреса дескриптора DMA конечной точки 2 OUT EP2
80_408Ch	STOUTEP2	Статусный регистр конечной точки 2 OUT EP2
Конечная точка OUT EP3		
80_4090h	CTRLOUTEP3	Регистр управления конечной точкой 3 OUT EP3
80_4094h	DMACTRLOUTEP3	Регистр управления DMA конечной точки 3 OUT EP3
80_4098h	DMADAOUTEP3	Регистр адреса дескриптора DMA конечной точки 3 OUT EP3
80_409Ch	STOUTEP3	Статусный регистр конечной точки 3 OUT EP3
Конечная точка IN EP0		
80_40A0h	CTRLINEP0	Регистр управления конечной точкой 0 IN EP0
80_40A4h	CTRLDMAINEP0	Регистр управления DMA конечной точки 0 IN EP0
80_40A8h	DMADAINEP0	Регистр адреса дескриптора DMA конечной точки 0 IN EP0
80_40Ach	STINEP0	Статусный регистр конечной точки 0 IN EP0
Конечная точка IN EP1		
80_40B0h	CTRLINEP1	Регистр управления конечной точкой 1 IN EP1
80_40B4h	CTRLDMAINEP1	Регистр управления DMA конечной точки 1 IN EP1
80_40B8h	DMADAINEP1	Регистр адреса дескриптора DMA конечной точки 1 IN EP1
80_40BCh	STINEP1	Статусный регистр конечной точки 1 IN EP1
Конечная точка IN EP2		
80_40C0h	CTRLINEP2	Регистр управления конечной точкой 2 IN EP2
80_40C4h	CTRLDMAINEP2	Регистр управления DMA конечной точки 2 IN EP2
80_40C8h	DMADAINEP2	Регистр адреса дескриптора DMA конечной точки 2 IN EP2
80_40CCh	STINEP2	Статусный регистр конечной точки 2 IN EP2
Конечная точка IN EP3		
80_40D0h	CTRLINEP3	Регистр управления конечной точкой 3 IN EP3
80_40D4h	CTRLDMAINEP3	Регистр управления DMA конечной точки 3 IN EP3
80_40D8h	DMADAINEP3	Регистр адреса дескриптора DMA конечной точки 3 IN EP3
80_40DCh	STINEP3	Статусный регистр конечной точки 3 IN EP3
Общие регистры DC контроллера		
80_40E0h	GLBLCTRLDC	Регистр управления DC контроллера
80_40E4h	GLBLSTDC	Регистр состояния DC контроллера
RAM DC контроллера		
80_6000h–80_6FFFh	DCRAM	ОЗУ DC контроллера 4K × 32

31	30	29	28	27	26	25	24
BUFSZ10	BUFSZ9	BUFSZ8	BUFSZ7	BUFSZ6	BUFSZ5	BUFSZ4	BUFSZ3
RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X
23	22	21	20	19	18	17	16
BUFSZ2	BUFSZ1	BUFSZ0	PI	CB	CS	MAXPL10	MAXPL9
RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X
15	14	13	12	11	10	9	8
MAXPL8	MAXPL7	MAXPL6	MAXPL5	MAXPL4	MAXPL3	MAXPL2	MAXPL1
RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X
7	6	5	4	3	2	1	0
MAXPL0	NT1	NT0	TT1	TT0	EH	ED	EV
RW-X	RW-X	RW-X	RW-X	RW-X	RWC-0	RWC-0	RWC-0

Примечание – Принятые условные обозначения: R – чтение, W – запись, C – бит сбрасывается, X – значение не определено.

Здесь и далее в названиях рисунков /адрес регистра/(***n***) – номер ENDPOINT (**EP*n***)

Например:

адрес регистра CTRLOUTEP0 для OUT EP**0** – 80_40(6+***n***)0h = 80_40**6**0h;

адрес регистра DMADAOUTEP3 для OUT EP**3** – 80_40(6+***n***)8h = 80_40**9**8h.

Рисунок 17.10 – Структура регистра управления конечной точки OUT CTRLOUTEP /адрес **80_40(6+*n*)0**h/

Таблица 17.9 – Описание битов регистра управления конечной точки OUT

Бит	Обозначение	Тип	Описание
1	2	3	4
31 – 21	BUFSZ10 – BUFSZ0	R	Размер/8 в байтах одного аппаратного буферного слота для этой ENDPOINT. Два слота доступны для каждого ENDPOINT
20	PI	RC	Прерывание после приёма пакета. Генерирует прерывание для каждого пакета, который получен по USB шине для этого ENDPOINT (пакет запоминается во внутреннем буфере). Значение после сброса – 0
19	CB	RW	Очищает любые буферы ENDPOINT, которые содержат данные, если буфер не активен
18	CS	RW	Управление STALL. Возвращает STALL для пакета данных и статуса в управляющей транзакции. Автоматически сбрасывается, когда принимается следующий маркер SETUP. Используется только в управляющей контрольной точке CNTRL EP
17 – 7	MAXPL10 – MAXPL0	RW	Максимальная длина одного пакета для ENDPOINT. Определяет максимальный размер одного пакета, принимаемого конечной точкой. Максимально допустимый размер пакета составляет 1024 байта. Поле не сбрасывается
6 – 5	NT1, NT0	RW	Количество транзакций. Устанавливает число дополнительных транзакций на микрофрейм для HS ENDPOINT и на фрейм для FS ENDPOINT. Действителен только для ISOCHRONOUS EP. Поле не сбрасывается

Окончание таблицы 17.9

1	2	3	4
4 – 3	TT1, TT0	RW	Тип передачи. Устанавливает тип передачи для ENDPOINT: - 00 – CTRL; - 01 – ISOCH; - 10 – BULK; - 11 – INTERRUPT. Только для OUT EP можно установить тип CTRL, после этого IN EP с тем же номером будет автоматически типа CTRL. Важно не использовать OUT EP, которые не имеют соответствующих IN EP в качестве CTRL EP. Поле не сбрасывается.
2	EH	RWC	Останов EP. Запись 1 останавливает конечную точку. В этом случае все передачи к этой EP будут получать ответ STALL HANDSHAKE. Значение после сброса – 0.
1	ED	RWC	Запрет EP. Если этот бит установлен в 1, то все передачи к этой точке будут получать NAK HANDSHAKE. Значение после сброса – 0.
0	EV	RWC	Конечная точка актуальна. Разрешает работу конечной точки. Если бит не установлен, то все передачи EP будут игнорироваться и не посылаться какой-либо ответ (HANDSHAKE). Значение после сброса – 0.
Примечание – Принятые условные обозначения: R –чтение, W – запись, C – сброс.			

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
15	14	13	12	11	10	9	8
RSV	RSV	RSV	RSV	RSV	AE	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-0	R-X	R-X
7	6	5	4	3	2	1	0
RSV	RSV	RSV	RSV	AD	AI	IE	DA
R-X	R-X	R-X	R-X	RWC-0	RW-0	RW-0	RW-0

Примечание – Принятые условные обозначения: R – чтение, W – запись, C – бит сбрасывается, X – значение не определено.

Рисунок 17.11 – Структура регистра управления OUT DMA DMACTRLOUTEP /адрес **80_40(6+*n*)4**h/

Таблица 17.10 – Описание битов регистра управления OUT DMA

Бит	Обозначение	Тип	Описание
1	2	3	4
31 – 11	RSV	R	Зарезервировано.
10	AE	R	Ошибка процессорной шины для этой EP.
9 – 4	RSV	R	Зарезервировано.
3	AD	RWC	Запрет обработки дескриптора (установка DA бита в 0) и прекращение текущей передачи DMA, если она была активной. Значение после сброса – 0.

Окончание таблицы 17.10

1	2	3	4
2	AI	RW	Разрешение прерывания по ошибке процессорной шины; генерирует прерывание, когда произошла ошибка процессорной шины для этой EP.
1	IE	RW	Разрешение прерывания DMA; каждый раз будет генерироваться прерывание, когда данные получены или переданы для дескриптора, если его бит разрешения прерывания установлен.
0	DA	RW	Разрешение дескриптора; установка этого бита в 1 укажет DC контроллеру, что один или больше дескрипторов готовы для обработки.
Примечание – Принятые условные обозначения: R –чтение, W – запись, C – сброс.			

31	30	29	28	27	26	25	24
DESC ADDR29	DESC ADDR28	DESC ADDR27	DESC ADDR26	DESC ADDR25	DESC ADDR24	DESC ADDR23	DESC ADDR22
RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X
23	22	21	20	19	18	17	16
DESC ADDR21	DESC ADDR20	DESC ADDR19	DESC ADDR18	DESC ADDR17	DESC ADDR16	DESC ADDR15	DESC ADDR14
RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X
15	14	13	12	11	10	9	8
DESC ADDR13	DESC ADDR12	DESC ADDR11	DESC ADDR10	DESC ADDR9	DESC ADDR8	DESC ADDR7	DESC ADDR6
RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X
7	6	5	4	3	2	1	0
DESC ADDR5	DESC ADDR4	DESC ADDR3	DESC ADDR2	DESC ADDR1	DESC ADDR0	RSV	RSV
RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	R-X	R-X

Примечание – Принятые условные обозначения: R – чтение, W – запись, C – бит сбрасывается, X – значение не определено.

Рисунок 17.12 – Структура регистра адреса OUT дескриптора DMADAOUTEP /адрес **80_40(6+*n*)8**h/

Таблица 17.11 – Описание битов регистра адреса OUT дескриптора

Бит	Обозначение	Тип	Описание
31 – 2	DESCADDR29 – DESCADDR0	RW	Адрес таблицы OUT дескрипторов; адрес к следующему дескриптору. Не сбрасывается.
1 – 0	RSV	R	Зарезервировано.
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

31	30	29	28	27	26	25	24
RSV	RSV	PR	B1CNT12	B1CNT11	B1CNT10	B1CNT9	B1CNT8
R-X	R-X	RWC-0	R-X	R-X	R-X	R-X	R-X
23	22	21	20	19	18	17	16
B1CNT7	B1CNT6	B1CNT5	B1CNT4	B1CNT3	B1CNT2	B1CNT1	B1CNT0
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
15	14	13	12	11	10	9	8
B0CNT12	B0CNT11	B0CNT10	B0CNT9	B0CNT8	B0CNT7	B0CNT6	B0CNT5
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
7	6	5	4	3	2	1	0
B0CNT4	B0CNT3	B0CNT2	B0CNT1	B0CNT0	B1	B0	BS
R-X	R-X	R-X	R-X	R-X	RC-0	RC-0	RW-0

Примечание – Принятые условные обозначения: R – чтение, W – запись, C – бит сбрасывается, X – значение не определено.

Рисунок 17.13 – Структура статусного регистра конечной точки OUT STOUTEP /адрес **80_40(6+*n*)C**h/

Таблица 17.12 – Описание битов статусного регистра конечной точки OUT

Бит	Обозначение	Тип	Описание
31 – 30	RSV	R	Зарезервировано.
29	PR	RWC	Пакет получен: - устанавливается каждый раз, когда пакет (OUT направление) был принят и сохранен во внутреннем буфере. Бит PR очищается записью в него 1.
28 – 16	B1CNT12 – B1CNT0	R	Счётчик байтов буфера 1: - количество байт в буфере 1.
15 – 3	B0CNT12 – B0CNT0	R	Счётчик байтов буфера 0: - количество байт в буфере 0.
2	B1	RC	Данные в буфере 1 актуальны; устанавливается в 1, когда в буфере 1 находятся актуальные данные.
1	B0	RC	Данные в буфере 0 актуальны; устанавливается в 1, когда в буфере 0 находятся актуальные данные.
0	BS	RW	Выбор буфера: - 0 – текущий буфер 0; - 1 – текущий буфер 1.
Примечание – Принятые условные обозначения: R – чтение, W – запись, C – бит сбрасывается.			

31	30	29	28	27	26	25	24
BUFSZ10	BUFSZ9	BUFSZ8	BUFSZ7	BUFSZ6	BUFSZ5	BUFSZ4	BUFSZ3
RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X
23	22	21	20	19	18	17	16
BUFSZ2	BUFSZ1	BUFSZ0	PI	CB	CS	MAXPL10	MAXPL9
RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X
15	14	13	12	11	10	9	8
MAXPL8	MAXPL7	MAXPL6	MAXPL5	MAXPL4	MAXPL3	MAXPL2	MAXPL1
RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X
7	6	5	4	3	2	1	0
MAXPL0	NT1	NT0	TT1	TT0	EH	ED	EV
RW-X	RW-X	RW-X	RW-X	RW-X	RWC-0	RWC-0	RWC-0

Примечание – Принятые условные обозначения: R – чтение, W – запись, C – бит сбрасывается, X – значение не определено.

Здесь и далее в названиях рисунков /адрес регистра/, (***n***) – номер ENDPOINT (**EP*n***)

Например:

адрес регистра CTRLINEP0 для IN EP**0** – 80_40(A+***n***)0h = 80_40**A**0h;
адрес регистра DMADAINEP3 для IN EP**3** – 80_40(A+***n***)8h = 80_40**D**8h.

Рисунок 17.14 – Структура регистра управления конечной точки IN CTRLINEP /адрес **80_40(A+*n*)0**h/

Таблица 17.13 – Описание битов регистра управления конечной точки IN

Бит	Обозначение	Тип	Описание
1	2	3	4
31 – 21	BUFSZ10 – BUFSZ0	RW	Размер буфера. Размер/8 в байтах одного аппаратного буферного слота для EP. Два слота доступны для каждой EP
20	PI	RW	Прерывание после передачи пакета. Генерирует прерывание каждый раз, когда пакет был передан по USB шине, и внутренний буфер очищен. Значение после сброса – 0
19	CB	RW	Очистка буферов. Очищает все буферы EP, которые содержат данные, если буфер сейчас не активен
18	CS	RW	Управление STALL. Возвращает STALL для данных и запроса статуса в управляющей передаче. Автоматически сбрасывается, когда получен следующий маркер SETUP. Используется только в управляющей контрольной точке CNTRL EP
17 – 7	MAXPL10 – MAXPL0	RW	Максимальная длина одного пакета для ENDPOINT. Определяет максимальный размер одного пакета, принимаемого конечной точки. Максимально допустимый размер пакета составляет 1024 байта. Поле не сбрасывается
6 – 5	NT1, NT0	RW	Количество транзакций. Устанавливает дополнительное число транзакций на микрофрейм для HS EP и на фрейм для FS EP. Актуален только для ISOCHRONOUS EP. Поле не сбрасывается

Окончание таблицы 17.13

1	2	3	4
4 – 3	TT1, TT0	RW	Тип передачи. Устанавливает тип передачи для ENDPOINT: - 00 – CTRL; - 01 – ISOCH; - 10 – BULK; - 11 – INTERRUPT. Только для OUT EP можно установить тип CTRL, после этого IN EP с тем же номером будет автоматически типа CTRL. Важно не использовать OUT EP, которые не имеют соответствующих IN EP в качестве CTRL EP. Поле не сбрасывается.
2	EH	RWC	Останов EP. Запись 1 останавливает конечную точку. В этом случае все передачи для этой EP будут получать ответ STALL HANDSHAKE. Значение после сброса – 0.
1	ED	RWC	Запрет EP. Если этот бит установлен в 1, то все передачи для этой точки будут получать NAK HANDSHAKE. Значение после сброса – 0.
0	EV	RWC	Конечная точка актуальна. Разрешает работу конечной точки. Если бит не установлен, то все передачи EP будут игнорироваться и не посылаться какой-либо ответ (HANDSHAKE). Значение после сброса – 0.
Примечание – Принятые условные обозначения: R –чтение, W – запись, C – сброс.			

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
15	14	13	12	11	10	9	8
RSV	RSV	RSV	RSV	RSV	AE	RSV	RSV
R-X	R-X	R-X	R-X	R-X	RC-0	R-X	R-X
7	6	5	4	3	2	1	0
RSV	RSV	RSV	RSV	AD	AI	IE	DA
R-X	R-X	R-X	R-X	RWC-0	RW-0	RW-0	RW-0

Примечание – Принятые условные обозначения: R – чтение, W – запись, C – бит сбрасывается, X – значение не определено.

Рисунок 17.15 – Структура регистра управления IN DMA CTRLDMAINEP /адрес **80_40(A+*n*)4**h/

Таблица 17.14 – Описание битов регистра управления IN DMA

Бит	Обозначение	Тип	Описание
1	2	3	4
31 – 11	RSV	R	Зарезервировано.
10	AE	RC	Ошибка процессорной шины для этой EP.
9 – 4	RSV	R	Зарезервировано.
3	AD	RWC	Запрет обработки дескриптора (установка DA бита в 0) и прекращение текущей передачи DMA, если она была активной. Значение после сброса – 0.

Окончание таблицы 17.14

1	2	3	4
2	AI	RW	Разрешение прерывания по ошибке процессорной шины; генерирует прерывание, когда произошла ошибка процессорной шины для этой EP.
1	IE	RW	Разрешение прерывания от DMA; каждый раз будет генерироваться прерывание, когда данные получены или переданы для дескриптора, если его бит разрешения прерывания установлен в 1.
0	DA	RW	Разрешение дескриптора; установка этого бита в 1 укажет DC контроллеру, что один или больше дескрипторов готовы для обработки.
Примечание – Принятые условные обозначения: R –чтение, W – запись, C – сброс.			

31	30	29	28	27	26	25	24
DESC ADDR29	DESC ADDR28	DESC ADDR27	DESC ADDR26	DESC ADDR25	DESC ADDR24	DESC ADDR23	DESC ADDR22
RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X
23	22	21	20	19	18	17	16
DESC ADDR21	DESC ADDR20	DESC ADDR19	DESC ADDR18	DESC ADDR17	DESC ADDR16	DESC ADDR15	DESC ADDR14
RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X
15	14	13	12	11	10	9	8
DESC ADDR13	DESC ADDR12	DESC ADDR11	DESC ADDR10	DESC ADDR9	DESC ADDR8	DESC ADDR7	DESC ADDR6
RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X
7	6	5	4	3	2	1	0
DESC ADDR5	DESC ADDR4	DESC ADDR3	DESC ADDR2	DESC ADDR1	DESC ADDR0	RSV	RSV
RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	R-X	R-X

Примечание – Принятые условные обозначения: R – чтение, W – запись, X – значение не определено.

Рисунок 17.16 – Структура регистра адреса IN дескриптора DMADAINEP
/адрес **80_40(A+*n*)8**h/

Таблица 17.15 – Описание битов регистра адреса IN дескриптора

Бит	Обозначение	Тип	Описание
31 – 2	DESCADDR29 – DESCADDR0	RW	Адрес таблицы IN дескрипторов; адрес следующего дескриптора. Не сбрасывается.
1 – 0	RSV	R	Зарезервировано.
Примечание – Принятые условные обозначения: R – только чтение, RW – чтение и запись.			

31	30	29	28	27	26	25	24
RSV	RSV	PT	B1CNT12	B1CNT11	B1CNT10	B1CNT9	B1CNT8
R-X	R-X	RWC-0	R-X	R-X	R-X	R-X	R-X
23	22	21	20	19	18	17	16
B1CNT7	B1CNT6	B1CNT5	B1CNT4	B1CNT3	B1CNT2	B1CNT1	B1CNT0
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
15	14	13	12	11	10	9	8
B0CNT12	B0CNT11	B0CNT10	B0CNT9	B0CNT8	B0CNT7	B0CNT6	B0CNT5
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
7	6	5	4	3	2	1	0
B0CNT4	B0CNT3	B0CNT2	B0CNT1	B0CNT0	B1	B0	BS
R-X	R-X	R-X	R-X	R-X	RC-0	RC-0	RW-0

Примечание – Принятые условные обозначения: R – чтение, W – запись, X – значение не определено.

Рисунок 17.17 – Структура статусного регистра конечной точки IN STINEP /адрес **80_40(A+*n*)C**h/

Таблица 17.16 – Описание битов статусного регистра конечной точки IN

Бит	Обозначение	Тип	Описание
31 – 30	RSV	R	Зарезервировано.
29	PT	RWC	Пакет передан: - устанавливается всякий раз по завершению передачи пакета и очистке внутреннего буфера. Бит PT очищается записью в него 1.
28 – 16	B1CNT12 – B1CNT0	R	Счётчик байтов буфера 1: - количество байт в буфере 1.
15 – 3	B0CNT12 – B0CNT0	R	Счётчик байтов буфера 0: - количество байт в буфере 0.
2	B1	RC	Данные в буфере 1 актуальны; устанавливается в 1, когда в буфере 1 находятся актуальные данные.
1	B0	RC	Данные в буфере 0 актуальны; устанавливается в 1, когда в буфере 0 находятся актуальные данные.
0	BS	RW	Выбор буфера: - 0 – текущий буфер 0; - 1 – текущий буфер 1.
Примечание – Принятые условные обозначения: R –чтение, W – запись, C – сброс.			

31	30	29	28	27	26	25	24
SI	UI	VI	SP	FI	RSV	RSV	RSV
R-0	RC-0	RC-0	RC-0	RWC-0	R-X	R-X	R-X
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
15	14	13	12	11	10	9	8
FT	EP	DH	RW	TS2	TS1	TS0	TM
RW-0	RW-0	RW-0	RW-0	RW-X	RW-X	RW-X	RWC-0
7	6	5	4	3	2	1	0
UA6	UA5	UA4	UA3	UA2	UA1	UA0	SU
RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RWC-0

Примечание – Принятые условные обозначения: R – чтение, W – запись, X – значение не определено.

Рисунок 17.18 – Структура регистра управления DC контроллера GLBLCTRLDC /адрес **80_40E0**h/

Таблица 17.17 – Описание битов регистра управления DC контроллера

Бит	Обозначение	Тип	Описание
1	2	3	4
31	SI	RC	Прерывание при приостановке; генерирует прерывание, при смене статуса состояния SUSPEND. Значение после сброса – 0
30	UI	RC	Сброс USB; генерирует прерывание, при обнаружении сброса USB. Значение после сброса – 0
29	VI	RC	Прерывание по сигналу VBusValid UTMII+ интерфейса; генерирует прерывание при изменении состояния сигнала VBusValid. Значение после сброса – 0
28	SP	RC	Прерывание при изменении режима скорости; генерирует прерывание при изменении режима скорости приёма/передачи. Значение после сброса – 0
27	FI	RWC	Прерывание при получении фрейма; генерирует прерывание при получении нового маркера SOF (Start Of Frame). Значение после сброса – 0
26 – 16	RSV	R	Зарезервировано
15	FT	RW	Режим функционального тестирования; разрешение функционального тест-режима, который сокращает все временные соотношения, такие как RESET и CHIRP (специальная последовательность импульсов, используемая для реализации протокола на шине USB) до восьми тактовых циклов
14	EP	RW	Разрешение резистора Pull-up; разрешает подключение резистора Pull-up к линии D+, сигнализирующей о подключении USB шины к HOST. Значение после сброса – 0
13	DH	RW	Запрет HS; запрещает HS (High Speed) HANDCSHAKE, чтобы сконфигурировать DC контроллер только в режим FS (Full Speed)

Окончание таблицы 17.17

1	2	3	4
12	RW	RW	Удалённый старт; запускает удалённый старт. Этот бит очищает сам себя: - при нахождении DC контроллера в режиме SUSPEND бит очищается по окончании передачи «удалённый старт»; - если DC контроллер не находился в режиме SUSPEND, то будучи установленным, бит сбрасывает себя немедленно. Когда бит установлен, запись в него игнорируется. Значение после сброса – 0
11 – 9	TS2 – TS0	RW	Выбор тест-режима: - 001 – Test_J; - 010 – Test _K; - 011 – Test_SE0_NAK; - 100 – Test _Packet
8	TM	RWC	Разрешение тест-режима: - установка в 1 разрешает тест-режим. Примечание – Тест-режим не может быть выключен без сброса и не может устанавливаться, если бит запрета High Speed (см. выше бит 13) установлен в 1. Значение после сброса – 0
7 – 1	UA6 –UA0	RW	Адрес на шине USB: адрес, назначенный DC контроллеру на USB шине хостом
0	SU	RWC	Устанавливает USB адрес: запись 1 в этот бит назначит адрес DC контроллеру из поля адреса шины USB
Примечание – Принятые условные обозначения: R – чтение, W – запись, C – сброс.			

31	30	29	28	27	26	25	24
NEPI3	NEPI2	NEPI1	NEPI0	NEPO3	NEPO2	NEPO1	NEPO0
R-nmbI	R-nmbI	R-nmbI	R-nmbI	R-nmbO	R-nmbO	R-nmbO	R-nmbO
23	22	21	20	19	18	17	16
DM	RSV	RSV	RSV	RSV	RSV	SU	UR
R-1	R-X	R-X	R-X	R-X	R-X	R-1	RC-0
15	14	13	12	11	10	9	8
VB	SP	AF2	AF1	AF0	FN10	FN9	FN8
R-0	R-0	R-X	R-X	R-X	R-X	R-X	R-X
7	6	5	4	3	2	1	0
FN7	FN6	FN5	FN4	FN3	FN2	FN1	FN0
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X

Примечание – Принятые условные обозначения: R – чтение, X – значение не определено, nmbI – число входных точек, nmbO – число выходных точек.

Рисунок 17.19 – Структура регистра состояния DC контроллера GLBLSTDC /адрес **80_40E4**h/

Таблица 17.18 – Описание битов регистра состояния DC контроллера

Бит	Обозначение	Тип	Описание
31 – 28	NEPI3 – NEPI0	R	Число реализованных конечных IN точек (3); поле отражает количество конфигурируемых IN ENDPOINT, доступных в DC контроллере (включая EP0) минус 1.
27 – 24	NEPO3 – NEPO0	R	Число реализованных конечных OUT точек (3); поле отражает количество конфигурируемых OUT ENDPOINT, доступных в DC контроллере (включая EP0) минус 1.
23	DM	R	Режим данных. Примечание – В данной реализации используется только DM = 1: - 1 – DC контроллер использует DMA режим передачи данных.
22 – 18	RSV	R	Зарезервировано.
17	SU	R	Приостановка работы DC контроллера (SUSPEND): - 0 – DC контроллер находится в состоянии SUSPEND; - 1 – DC контроллер не находится в состоянии SUSPEND.
16	UR	RC	Сброс USB: - устанавливается всякий раз, когда на USB шине детектируется сброс. Бит UR очищается записью в него 1.
15	VB	R	Сигнал VBusValid: - 1 – на линии USB VBus детектируется правильное напряжение.
14	SP	R	Скорость (актуальный режим скорости USB шины): - 0 – режим High Speed (HS) – 480 Мбит/с; - 1 – режим Full Speed (FS) – 12 Мбит/с.
13 – 11	AF2 – AF0	R	Дополнительные фреймы. Количество дополнительных фреймов, принятых с текущим номером фрейма.
10 – 0	FN10 – FN0	R	Номер фрейма; значение последнего полученного SOF маркера.
Примечание – Принятые условные обозначения: R – чтение, C – начальный сброс.			

18 Периферийное устройство TechBus (технологическая шина)

Технологическая шина (ТВ) представляет собой пользовательский параллельный интерфейс, обеспечивающий доступ к специальной периферии со стороны ИС 1867ВН016. Основная особенность ТВ – мультиплексная передача адресов/данных по одной и той же шине. Список сигналов интерфейса ТВ с описанием их назначения приведён в таблице 18.1. Для организации обмена через технологическую шину используются четыре картированных в памяти регистра, составляющие основу архитектуры ТВ. Название, функциональное назначение и адреса этих регистров приведены в таблице 18.2, а структура и форматы полей и отдельных разрядов – на рисунках 18.1 – 18.4 и в таблицах 18.3 – 18.6.

Таблица 18.1 – Сигналы технологической шины и их функциональное назначение

Обозначение	Описание	Тип
AD(15–00)_TB	16 мультиплексированных бит адреса/данных технологической шины.	I/O/Z
INIT#_TB	Начальный запуск (выходная разовая команда, по включению питания). Вывод отражает состояние бита «INIT» (Нулевой разряд в регистре RK).	O
WA#_TB	Строб адреса. Запись адреса.	O
RD#_TB	Строб чтения данных. Адресное чтение данных.	O
WD#_TB	Строб записи данных. Адресная запись данных.	O
PKO#_TB	Выходная разовая команда. Второй бит регистра команд, только запись.	O
PKI_TB	Входная разовая команда. Первый бит регистра команд, только чтение.	I
ENLPT#_TB	Технологический вход начального запуска.	I
WPK1#_TB	Сигнал безадресной записи данных 1.	O
RRK#_TB	Сигнал безадресного чтения данных.	O
ZPR_TB	Запрос на прерывание процессора.	I
ERDYn	Внешний сигнал готовности периферии.	I
Примечание – Принятые условные обозначения: I/O/Z – вход/выход (двунаправленный)/третье состояние, I – вход, O – выход.		

Таблица 18.2 – Список регистров TechBus и их функциональное назначение

Номер по порядку	Адрес	Обозначение	Доступ	Описание
1	80_4613h	PRMR	RWC	Регистр параметров записи/чтения.
2	80_4614h	ADDR	RW	Регистр адреса.
3	80_4615h	DSWR	RW	Регистр данных и сигналов записи данных.
4	80_4616h	RK	RW	Регистр команд.
Примечание – Принятые условные обозначения: R – чтение, W –запись, C – начальный сброс.				

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	DVDH1	DVDH1
R-X	R-X	R-X	R-X	R-X	R-X	RWC-0	RWC-0
15	14	13	12	11	10	9	8
DVDH1	DVDH1	RD/RRK	NR3	NR2	NR1	NR0	NW3
RWC-1	RWC-0	RWC-0	RWC-0	RWC-0	RWC-1	RWC0	RWC-0
7	6	5	4	3	2	1	0
NW2	NW1	NW0	NA3	NA2	NA1	NA0	RSV
RWC-0	RWC-1	RWC-0	RWC-0	RWC-0	RWC-1	RWC-0	RWC-0

Примечание – Принятые условные обозначения: R – чтение, W – запись, C – бит сбрасывается, X – значение не определено.

Рисунок 18.1 – Структура регистра параметров записи/чтения PRMR /адрес **80_4613**h/

Таблица 18.3 – Формат регистра параметров записи/чтения PRMR

Бит	Обозначение	Доступ	Описание
1	2	3	4
31 – 18	RSV	R	Зарезервировано.
17 – 14	DVDH1	RWC	Количество тактов процессорного сигнала H1 (N_{H1}), через которое формируется внутренний сигнал DVDH1, где диапазон значений поля DVDH1: 0x0 – 0xF. Пример – после системного сброса поле DVDH1 = 0x2, что соответствует числу тактов N_{H1} равному 3.
13	RD/RRK	RWC	Выбор формирования сигналов RD#_TB или RRK#_TB: - если RD/RRK = 0, формируется RD#_TB; - если RD/RRK = 1, формируется RRK#_TB.
12 – 9	NR	RWC	Количество тактов внутреннего сигнала DVDH1 (N_{DVDH1}), через которые при чтении из регистра DSWR формируется сигнал XRDY, где диапазон значений поля NR: 0x0 – 0xF. Примеры – если: - NR = 0x0, XRDY формируется от внешнего ERDYn; - NR = 0x1, XRDY формируется через 2 такта DVDH1; - NR = 0xF, XRDY формируется через 17 тактов DVDH1.
8 – 5	NW	RWC	Количество тактов внутреннего сигнала DVDH1 (N_{DVDH1}), через которые при записи в регистр DSWR формируется сигнал XRDY, где диапазон значений поля NW: 0x0 – 0xF. Примеры – если: - NW = 0x0, XRDY формируется от внешнего ERDYn; - NW = 0x1, XRDY формируется через 2 такта DVDH1; - NW = 0xF, XRDY формируется через 17 тактов DVDH1.
4 – 1	NA	RWC	Количество тактов внутреннего сигнала DVDH1 (N_{DVDH1}), через которые при записи в регистр ADDR формируется сигнал XRDY, где диапазон значений поля NA: 0x0 – 0xF. Примеры – если: - NA = 0x0, XRDY формируется от внешнего ERDYn; - NA = 0x1, XRDY формируется через 2 такта DVDH1; - NA = 0xF, XRDY формируется через 17 тактов DVDH1.

Окончание таблицы 18.3

1	2	3	4
0	RSV	RWC	Зарезервировано.

Примечания

1 Принятые условные обозначения: R – чтение, W –запись, C – начальный сброс.

2 Число тактов процессорного сигнала H1 (N_{H1}), необходимых для формирования WA#_TB, RD#_TB, WD#_TB, WPK1#_TB, RRK#_TB определяется формулой:

$$N_{H1} = N_X \times (DVDH1 + 2) + DVDH1, \qquad (18.1)$$

где N_{H1} – число тактов H1,

N_X – значение поля NR, NW, NA регистра PRMR (см. рисунок 18.1),

DVDH1 – значение поля DVDH1 регистра PRMR (см. рисунок 18.1).

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
15	14	13	12	11	10	9	8
ADDR15	ADDR14	ADDR13	ADDR12	ADDR11	ADDR10	ADDR9	ADDR8
RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X
7	6	5	4	3	2	1	0
ADDR7	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	ADDR0
RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X

Примечание – Принятые условные обозначения: R – чтение, W – запись, X – значение не определено.

Рисунок 18.2 – Структура регистра адреса ADDR /адрес **80_4614**h/

Таблица 18.4 – Описание битов регистра адреса ADDR

Бит	Обозначение	Доступ	Описание
31 – 16	RSV	R	Зарезервировано.
15 – 0	ADDR	RW	Поле адреса. Содержит текущее значение адреса, передаваемого на шину AD(15–00)_TB. При записи в регистр ADDR формируется сигнал записи адреса WA#_TB.
Примечание – Принятые условные обозначения: R – чтение, W – запись.			

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	WD	WRK1	RSV	RSV
R-X	R-X	R-X	R-X	Wr*-X	Wr*-X	R-X	R-X
15	14	13	12	11	10	9	8
DATA15	DATA14	DATA13	DATA12	DATA11	DATA10	DATA9	DATA8
RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X
7	6	5	4	3	2	1	0
DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0
RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X	RW-X

Примечание – Принятые условные обозначения: R – чтение, W – запись, X – значение не определено, r* – чтение возвращает невалидное значение.

Рисунок 18.3 – Структура регистра записи/чтения данных DSWR /адрес **80_4615**h/

Таблица 18.5 – Описание битов регистра записи/чтения данных DSWR

Бит	Обозначение	Доступ	Описание
31 – 20	RSV	R	Зарезервировано
19	WD	Wr*	Бит разрешения формирования сигнала WD#_TB. Если: - WD = 0, то сигнал WD#_TB не формируется; - WD = 1, то сигнал WD#_TB формируется.
18	WRK1	Wr*	Бит разрешения формирования сигнала WPK1#_TB. Если: - WRK1 = 0, то сигнал WPK1#_TB не формируется; - WRK1 = 1, то сигнал WPK1#_TB формируется.
17 – 16	RSV	R	Зарезервировано
15 – 0	DATA15 – DATA0	RW	Данные записи/чтения. Это поле содержит данные на шине AD(15–00)_TB. Формирование сигналов записи WD#_TB, WPK1#_TB определяется состоянием битов WD, WRK1 этого регистра. При чтении из регистра DSWR формируется сигнал чтения RD#_TB или RRK#_TB в зависимости от бита RD/RRK в регистре PRMR.
Примечание – Принятые условные обозначения: R – чтение, W – запись, r* – чтение возвращает невалидное значение.			

31	30	29	28	27	26	25	24
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
15	14	13	12	11	10	9	8
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
7	6	5	4	3	2	1	0
RSV	RSV	RSV	RSV	RSV	RKO	RKI	INIT
R-X	R-X	R-X	R-X	R-X	RWC-1	RKI#_TB	RWC-0

Примечание – Принятые условные обозначения: R – чтение, W – запись, C – бит сбрасывается, X – значение не определено.

Рисунок 18.4 – Структура регистра команд RK /адрес **80_4616**h/

Таблица 18.6 – Описание битов регистра команд RK

Бит	Обозначение	Доступ	Описание
31 – 3	RSV	R	Зарезервировано.
2	RKO	RWC	Запись 0 в этот разряд формирует на выходе PKO#_TB низкий уровень. Запись 1 в этот разряд формирует на выходе PKO#_TB высокий уровень. После системного сброса бит RKO = 1.
1	RKI	R	Чтение этого бита возвращает состояние вывода PKI#_TB. Если на выводе PKI#_TB: - высокий уровень, то бит RKI = 1; - низкий уровень, то бит RKI = 0. Задержка изменения значения этого бита при изменении состояния на выводе PKI#_TB составляет 2 – 3 такта H1.
0	INIT	RWC	Бит начального запуска. Если: - INIT = 0, то на выводе INIT#_TB будет низкий уровень; - INIT = 1, то на выводе INIT#_TB будет высокий уровень. После системного сброса значение бита INIT#_TB = 0.
Примечание – Принятые условные обозначения: R – чтение, W –запись, C – начальный сброс.			

Временная диаграмма записи/чтения Адреса/Данных по шине TechBus приведена на рисунке 18.5.

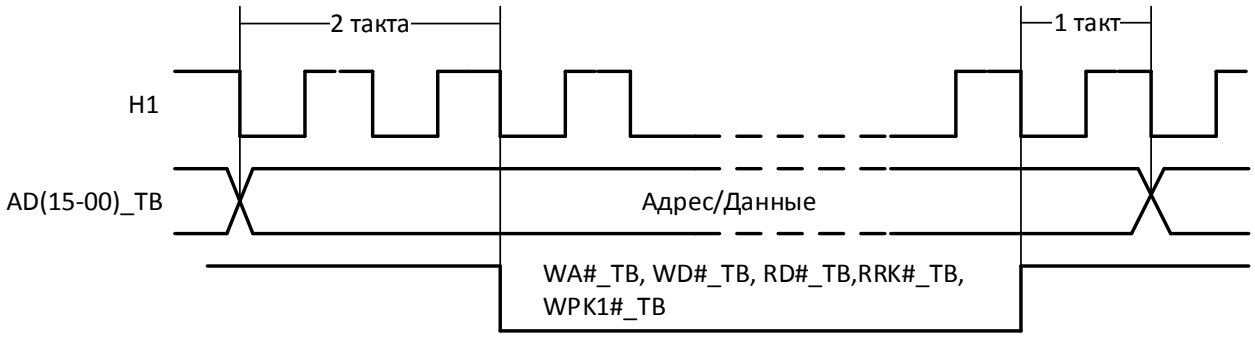


Рисунок 18.5 – Временная диаграмма записи/чтения Адреса/Данных по шине ТВ

На рисунке 18.6 приведена диаграмма адресного/безадресного чтения/записи данных по шине TechBus.

Рисунок 18.6 – Адресная/безадресная запись/чтение данных по шине TechBus

Примечания

1 При выполнении операции «ЗАПИСЬ ДАННЫХ» в регистре DSWR необходимо записать 1 в один из битов WD, WRK1 в зависимости от необходимости формирования на шине TechBus сигналов WD#_TB, WPK1#_TB соответственно.

2 При выполнении операции «ЧТЕНИЕ ДАННЫХ» в регистре PRMR необходимо установить бит RD/RRK в 0 при формировании сигнала RD#_TB или установить в 1 при формировании сигнала RRK#_TB.

3 Две инструкции NOP для формирования промежутка между адресом и данными.

19 Периферийное устройство преобразователь частоты в код (ПЧК)

Периферийное устройство преобразователь частоты в код (далее – ПЧК) предназначен для подсчёта сигналов, поступающих на входы UPXp, UPYp, UPZp, UMXp, UMYp, UMZp, PPXn, PPYn, PPZn, PMXn, PMYn, PMZn, в зависимости от входа MI (инкрементирования или декрементирования полей VALUE соответствующих регистров).

Таблица 19.1 – Алгоритм формирования значения полей SGN и VALUE

Регистр	Алгоритм формирования значения полей SGN и VALUE
SUX	Если MI = 1, то: VALUE = VALUE + UPX – UMX, иначе: VALUE = VALUE – UPX + UMX
SUY	Если MI = 1, то: VALUE = VALUE + UPY – UMY, иначе: VALUE = VALUE – UPY + UMY
SUZ	Если MI = 1, то: VALUE = VALUE + UPZ – UMZ, иначе: VALUE = VALUE – UPZ + UMZ
DUX	VALUE = VALUE + UPX – UMX
DUY	VALUE = VALUE + UPY – UMY
DUZ	VALUE = VALUE + UPZ – UMZ
DPX	VALUE = VALUE + PPX – PMX
DPY	VALUE = VALUE + PPY – PMY
DPZ	VALUE = VALUE + PPZ – PMZ
Примечания 1 Префикс «+» перед названием сигнала означает, что приходящий импульс по входу инкрементирует счётчик соответствующего регистра. 2 Префикс «–» перед названием сигнала означает, что приходящий импульс по входу декрементирует счётчик соответствующего регистра.	

Таблица 19.2 – Алгоритм формирования выходных сигналов

Сигнал	Алгоритм формирования выходных сигналов
T1	T1 = CTT[3] & CTT[4]. Синхронный к FI
T2	T2 = CTT[3] &~ CTT[4] & CTT[5]. Синхронный к FI
T3	T3 = MI & T2. Синхронный к FI
T4	T4 = CTP & MII. Синхронный к FI
C1	C1 = CTT[3]. Синхронный к FI
C2	C2 = (CT[5] & RU[2] &~ RU[3] | (CT[4] &RU[2]) | (CT[3] & RU[3]). Синхронный к FI
C3	C3 = INV0 & CTT[3]. Синхронный к FI
C4	C4 = INV1 & CTT[3]. Синхронный к FI

Таблица 19.3 – Параметры входных сигналов ПЧК

Номер по порядку	Имя сигнала	Длительность импульса, нс	Частота, МГц	Период, нс
1	FI	31,25	16,000	125
2	UPi	500 – 1500	0,500	2000
3	UMi	500 – 1500	0,500	2000
4	PPi	1000	0,064	15625
5	PMi	1000	0,064	15625
6	MI	500000	0,001	1000000

Регистры ПЧК представлены в таблице 19.4.

Таблица 19.4 – Список регистров ПЧК

Адрес, hex	Обозначение	Описание
80_4680	SUX	Суммы по UX.
80_4681	SUY	Суммы по UY.
80_4682	SUZ	Суммы по UZ.
80_4683	DUX	Дельты по UX.
80_4684	DUY	Дельты по UY.
80_4685	DUZ	Дельты по UZ.
80_4686	RSV	Зарезервировано. Запись не изменяет значения, читаются нули.
80_4687	RSV	Зарезервировано. Запись не изменяет значения, читаются нули.
80_4688	DPX	Дельты по PX.
80_4689	DPY	Дельты по PY.
80_468A	DPZ	Дельты по PZ.
80_468B	RSV	Зарезервировано. Запись не изменяет значения, читаются нули.
80_468C	TP	Значение времени на интервале 12 мс.
80_468D	VC	Вектор.

Структура регистров SUX, SUY, SUZ, DUX, DUY, DUZ, DPX, DPY, DPZ приведена на рисунке 19.1, а в таблице 19.5 – описание битов этих регистров.

Структура регистров TP и VC приведена на рисунках 19.2 и 19.3, а в таблицах 19.6, 19.7 – описание битов их регистров соответственно.

31	30	29	28	27	26	25	24
RDYXX	ERRXX	RSV	RSV	RSV	RSV	RSV	RSV
RC-0	RWC-0	R-X	R-X	R-X	R-X	R-X	R-X
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
15	14	13	12	11	10	9	8
RSV	RSV	EnIntXX	SGN	VALUE11	VALUE10	VALUE9	VALUE8
R-X	R-X	RWC-0	R-X	R-X	R-X	R-X	R-X
7	6	5	4	3	2	1	0
VALUE7	VALUE6	VALUE5	VALUE4	VALUE3	VALUE2	VALUE1	VALUE0
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X

Примечание – Принятые условные обозначения: R – чтение, W – запись, C – бит сбрасывается, X – значение не определено.

Рисунок 19.1 – Структура регистров SUX, SUY, SUZ, DUX, DUY, DUZ, DPX, DPY, DPZ

Таблица 19.5 – Описание битов регистров SUX, SUY, SUZ, DUX, DUY, DUZ, DPX, DPY, DPZ

Бит	Обозначение	Доступ	Описание
1	2	3	4
31	RDY*XX*	RC	Бит готовности: - RDY*XX* = 1 – поле VALUE и бит SGN регистра *XX* содержат актуальные данные. Если RDY*XX* = 1, то изменение регистра *XX* заблокировано; - RDY*XX* = 0 – в соответствующий регистр осуществляется перезапись данных из реверсивного счётчика по сигналу С2. Бит RDY*XX* сбрасывается в «0» системным сбросом и чтением из регистра *XX*. Установка бита RDY*XX* в «1» вызывает прерывание, если оно разрешено. Бит устанавливается через каждые 4, 2, 1 мс, в зависимости от состояния поля RU32 регистра ТР.
30	ERR*XX*	RC	Бит ошибки ERR*XX* устанавливается в «1», если в течение 4, 2, 1 мс, в зависимости от состояния поля RU32 регистра ТР, не был прочитан регистр *XX*, т. е. бит готовности RDY*XX* не был сброшен чтением из регистра. Установка бита ERR*XX* в «1» вызывает прерывание, если оно разрешено. Бит ошибки ERR*XX* устанавливается в «0» системным сбросом и записью «0» в этот бит.
29 – 14	RSV	R	Зарезервировано
13	EnInt*XX*	RWC	Разрешение прерывания при установке бита RDY*XX* в регистре *XX*. Прерывание происходит по вектору 0x0D: - EnInt*XX* = 1 – прерывание разрешено; - EnInt*XX* = 0 – прерывание запрещено. Бит EnInt*XX* сбрасывается в «0» системным сбросом.
12	SGN	R	Знаковый бит. Бит SGN актуален в том случае, если бит RDY*XX* = 1 и бит ERR*XX* = 0.
11 – 0	VALUE11-VALUE0	R	Значение реверсивного счётчика. Поле VALUE является актуальным в том случае, если бит RDY*XX* = 1 и бит ERR*XX* = 0.

Примечания

1 Суффикс *XX* в названии бит кодирует имя регистра и может принимать значение: SUX, SUY, SUZ, DUX, DUY, DUZ, DPX, DPY, DPZ.

2 Принятые условные обозначения: R – чтение, W – запись, C – бит сбрасывается.

31	30	29	28	27	26	25	24
RDYRTM	ERR	RSV	RSV	RSV	RSV	RSV	RSV
RC-0	RWC-0	R-X	R-X	R-X	R-X	R-X	R-X
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
15	14	13	12	11	10	9	8
RSV	RSV	EnInt	RSV	RTM7	RTM6	RTM5	RTM4
R-X	R-X	R-0	R-X	RC-0	RC-0	RC-0	RC-0
7	6	5	4	3	2	1	0
RTM3	RTM2	RTM1	RTM0	RU32	RU32	RU10	RU10
RC-0	RC-0	RC-0	RC-0	RWC-0	RWC-0	RWC-0	RWC-0

Примечание – Принятые условные обозначения: R – чтение, W – запись, C – бит сбрасывается, X – значение не определено.

Рисунок 19.2 – Структура регистра ТР

Таблица 19.6 – Описание битов регистра TP

Бит	Обозначение	Доступ	Описание
31	RDYRTM	RC	Бит готовности поля данных поля RTM: - RDYRTM = 1 – поле RTM содержит актуальные данные. Если бит RDYRTM = 1, то изменение поля RTM заблокировано; - RDYRTM = 0 – данные в регистрах недействительны и в это поле может осуществляться перезапись данных из счётчика времени. Бит RDYRTM сбрасывается в «0» системным сбросом и чтением из регистра TP.
30	ERR	RWC	Бит ошибки ERR устанавливается в «1», если в течение 30 мкс не был прочитан регистр TP, т. е. бит готовности RDYRTM не был сброшен чтением из регистра. Установка бита ERRTP в «1» вызывает прерывание, если оно разрешено. Бит ошибки ERR устанавливается в «0» системным сбросом и записью «0» в этот бит.
29 – 14	RSV	R	Зарезервировано.
13	EnInt	RWC	Разрешение прерывания при установке битов RDYRTM и ERR. Прерывание происходит по вектору 0x0D: - EnInt = 1 – прерывание разрешено; - EnInt = 0 – прерывание запрещено. Бит EnInt сбрасывается в «0» системным сбросом.
12	RSV	R	Зарезервировано.
11 – 4	RTM7-RTM0	R	Значение времени на интервале 12 мс.
3 – 2	RU32	RWC	Интервал прерывания и формирования сигнала прерывания C2: - RU32 = 00 – C2 = 4 мс; - RU32 = 01 – C2 = 2 мс; - RU32 = 1x – C2 = 1 мс.
1 – 0	RU10	RWC	Управление частотой выходных сигналов C3, C4: - RU10 = 00 – частота сигналов C3, C4 = 8 кГц, P32 = CTT[7], P64 = CT[0], P125 = CT[1]; - RU10 = 01 – частота сигналов C3, C4 = 16 кГц, P32 = CTT[7], P64 = CT[0], P125 = «1;» - RU10 = 10 – частота сигналов C3, C4 = 32 кГц, P32 = CTT[7], P64 = VCC, P125 = «1»; - RU10 = 11 – частота сигналов C3, C4 = 64 кГц, P32 = P64 = P125 = «1».

Примечание – Принятые условные обозначения: R – чтение, W – запись, C – бит сбрасывается.

31	30	29	28	27	26	25	24
RDYRTM	ERR	RSV	RSV	RSV	RSV	RSV	RSV
RC-0	RWC-0	R-X	R-X	R-X	R-X	R-X	R-X
23	22	21	20	19	18	17	16
RSV	RSV	RSV	RSV	RSV	RSV	RSV	RSV
R-X	R-X	R-X	R-X	R-X	R-X	R-X	R-X
15	14	13	12	11	10	9	8
RSV	RSV	RSV	ZERO	ZERO	ZERO	ZERO	ZERO
R-X	R-X	R-X	RC-0	RC-0	RC-0	RC-0	RC-0
7	6	5	4	3	2	1	0
ZERO	ZERO	ZERO	ST	D	ONE	T8	T4
RC-0	RC-0	RC-0	R-X	R-X	RC-1	RC-0	RC-0

Примечание – Принятые условные обозначения: R – чтение, W – запись, C – бит сбрасывается, X – значение не определено.

Рисунок 19.3 – Структура регистра VC

Таблица 19.7 – Описание битов регистра VC

Бит	Обозначение	Доступ	Описание
31	RDYRTM	RC	Копия бита RDYRTM в регистре TP.
30	ERR	RWC	Копия бита ERR в регистре TP.
29 – 13	RSV	R	Зарезервировано.
12 – 5	ZERO	R	Ноль.
4	ST	R	Сигнал «СТОП»: - ST = 1 – разрешена работа ПЧК (на блоки ПЧК поступает тактовый сигнал FI); - ST = 0 – работа ПЧК запрещена. Бит ST синхронизирован с передним фронтом тактового сигнала процессора H1.
3	D	R	Признак появления сигнала CTR в течение 4 мс. Бит D синхронизирован с передним фронтом тактового сигнала процессора H1.
2	ONE	RC	Единица.
1	T8	RC	T8 = TP[11] или RTM[7]. Бит T8 синхронизирован с передним фронтом тактового сигнала процессора H1. Значение этого бита действительно, когда бит RDYRTM равен «1».
0	T4	RC	T4 = TP[10] или RTM[6]. Бит T4 синхронизирован с передним фронтом тактового сигнала процессора H1. Значение этого бита действительно в том случае, если бит RDYRTM равен «1».
Примечание – Принятые условные обозначения: - X – значение не определено, - R – только чтение, - RC – чтение/сброс, - RWC – чтение/запись/сброс.			

Примечания

1 Сигнал IRQn формируется по переднему фронту H1. Сигнал IRQn формируется на 2, 3 такта H1 позже, чем приходит высокий уровень сигнала C2.

2 При чтении из регистров ПЧК поле SWW Bit Field регистра управления расширенной шиной EBCR (Expansion Bus Control Register; адрес – 80_8060h) должно быть установлено в 00b.

20 Отладочные средства ИС 1867ВН016

ИС 1867ВН016 является однородной вычислительной системой и содержит два одинаковых процессорных ядра, каждое из которых является полным функциональным аналогом ИС 1867ВЦ6Ф (TMS320C30).

Существует обширная номенклатура программных и аппаратных средств для разработки и отладки программ для ИС 1867ВЦ6Ф (TMS320C30), поставляемых как фирмой Texas Instruments, так и большим количеством других специализированных фирм, разрабатывающих подобные средства. Процесс отладки осуществляется под управлением комплекса средств, состоящего из программной и аппаратной частей, обеспечивающих подключение и взаимодействие с процессором.

К программным средствам относятся:

- интегрированная среда разработки Code Composer от фирмы Texas Instruments;
- компиляторы C, C++;
- ассемблеры, дизассемблеры, линкеры и отладчики-симуляторы;
- операционные системы реального времени.

К аппаратным средствам относятся одноканальные эмуляторы типа EMU510PCI и MIRAGE-NE1, двухканальные типа MIRAGE-NC2, другие внутрисхемные эмуляторы и оценочные модули.

Интерактивная среда разработки Code Composer даёт возможность разрабатывать и выполнять отладку программ, как в режиме симулятора, так и непосредственно в составе проектируемой аппаратуры в режиме внутрисхемной эмуляции с обменом данными через специализированный тестовый порт MPSD (специализированный последовательный тестовый порт).

Интегрированная программная среда обеспечивает доступ к внутренним ресурсам микросхемы: функциональным регистрам процессора, регистрам периферийных устройств, внутренней памяти и внешней памяти.

Основные функции интегрированной среды проектирования пользовательского ПО:

- определение статуса выполнения пользовательской программы: ход или останов;
- запуск на выполнение программы пользователя в режиме реального времени;
- останов программы пользователя в произвольный момент времени;
- определение текущего адреса выполнения программы пользователя при останове;
- изменение текущего адреса выполнения программы пользователя в останове;
- управление тактированием периферийных устройств в останове;
- определение статуса подключения внутренней памяти;
- определение статуса нахождения микропроцессора в режимах энергосбережения;
- установка и сброс точек останова по адресу выполнения программы;
- шаг низкого уровня (выполнение инструкции) с входом в подпрограммы;
- шаг низкого уровня (выполнение инструкции) без входа в подпрограммы;
- шаг высокого уровня (выполнение программы по строкам исходного кода на языке C) с входом в подпрограммы;
- шаг высокого уровня без входа в подпрограммы;
- чтение и запись памяти программ, памяти данных и регистров.

Доступ к внутренним аппаратным ресурсам процессора осуществляется в режиме отладки (debug), вход в который осуществляется по следующим событиям:

- выполнение breakpoint (точка останова) инструкции;
- установкой break-now (останова «сейчас») бита в управляющем регистре;
- после операции single step («одиночный шаг»).

После входа в режим отладки значение счётчиков команд сохраняется во временных регистрах, выставляется активный уровень внешнего сигнала, таймерный модуль прекращает счёт, сторожевой таймер останавливается. В режиме отладки обеспечивается поддержка следующих функций:

- загрузка программы в адресуемую память (File → Load Program);
- перезагрузка программы в адресуемую память (File → Reload Program);
- загрузка данных в страницу Data (File → Load Data);
- сохранение данных из страницы Data (Save Data → File);
- чтение/редактирование памяти/регистров (View/Edit Memory/Registers);
- установка аппаратных/программных точек останова (Debug → HW/SW Breakpoints);
- установка временных точек останова (Debug → Watch Breakpoints);
- установка точек останова по событию/сигналу (Debug → Event/Probe Point);
- установка пошагового исполнения программы внутри функции Step Info;
- установка пошагового исполнения программы на функциях Step Over (Debug → Step Over);
- установка пошагового исполнения программы при выходе из функции Step Out (Debug → Step Out);
- запуск процессора на выполнение программы с остановкой на курсоре в окне Disassembler (Debug → Run to Cursor);
- продолжение исполнения программы после останова (Debug → Animate);
- запуск/останов исполнения программы (Debug → Run/Halt);
- запуск процессора на выполнение программы в реальном времени с блокировкой Breakpoints (Debug → Run Free Real Time Mode);
- сброс/перезапуск процессора (Debug → Reset/Restart DSP).

21 Заключение

В настоящем руководстве пользователя приведены архитектура, функциональное построение, система команд и особенности применения радиационно-стойкой двухъядерной «системы в корпусе» ИС 1867ВН016. Процессорное ядро подробно описано в приложении А к техническому описанию.

Все значения электрических параметров ИС приведены в технических условиях АЕНВ.431280.365ТУ, в настоящем документе представлены справочные значения параметров.

Руководство пользователя может служить практическим руководством по применению высокопроизводительного двухъядерного процессора цифровой обработки сигналов с фиксированной и плавающей запятой для разработчиков систем на основе микросхемы 1867ВН016.

Приложение А

(обязательное)

«Система в корпусе», процессорное ядро

Содержание

А.1 Введение

В приложении А технического описания КФДЛ.431299.034ТО подробно рассматриваются вопросы реализации и функционирования ИС 1867ВН016 – радиационно-стойкого двухъядерного высокопроизводительного процессора цифровой обработки сигналов, как «системы в корпусе» в целом, включая аспекты межъядерного взаимодействия, взаимодействия каждого ядра с совместно используемыми (разделяемыми) периферийными устройствами. Кроме этого, детально описывается функционирование каждого из разделяемых периферийных устройств и общего для обоих ядер двухпортового ОЗУ.

ИС 1867ВН016 является однородной двухъядерной вычислительной системой, каждое процессорное ядро которой в полном объёме реализует архитектуру и систему команд ПЦОС с плавающей и фиксированной запятой 1867ВЦ6Ф (TMS320C30). Настоящий документ представляет собой описание процессорного ядра ИС 1867ВН016, которое имеет следующие функциональные характеристики:

Разрядность данных, бит:	
- плавающая запятая (ПЗ)	40
- фиксированная запятая (ФЗ)	32
Аппаратный умножитель, бит:	
- ПЗ	32 × 32 = 40
- ФЗ	24 × 24 = 32
Объём ПЗУ, бит	4K × 32
Объём ОЗУ, бит	2K × 32
Объём кэш ОЗУ команд, бит	64 × 32
Объём внешней адресуемой памяти, бит	16M × 32
Таймеры	2
Контроллер ПДП	1
Количество команд	113

Информация о процессорном ядре, приводимая в настоящем документе, в основном применима к серийно выпускаемым в виде отдельных микросхем его функциональным аналогам. Изменения и дополнения в этом ядре, связанные с добавлением новых периферийных устройств и интерфейсов, а также с организацией межъядерного обмена данными, выделены в тексте бледно-зелёным фоном (см. пример далее).

Пример – Информация о дополнениях в центральном процессоре

Далее подробно рассматриваются архитектура, регистры и память, периферийные устройства, адресация, форматы данных, система команд и другие вопросы, связанные с функционированием высокопроизводительного ядра цифровой обработки сигналов с плавающей и фиксированной запятой двухъядерной радиационно-стойкой «системы в корпусе» ИС 1867ВН016. Кроме этого, детально описывается функционирование неразделяемых (индивидуально ассоциированных с каждым из ядер) периферийных устройств.

А.2 Обзор архитектуры ядра

Сочетание аппаратных и программных решений для реализации арифметических алгоритмов нашло своё отражение в архитектуре процессорного ядра. Высокая производительность достигнута благодаря увеличенной точности и большому динамическому диапазону операций с плавающей запятой, большому объёму собственной памяти ядра, высокой степени параллелизма и наличию контроллера прямого доступа к памяти ПДП (DMA).

Основные темы, представленные в разделе А.2:

- центральное процессорное устройство ЦПУ (CPU) (подраздел А.2.1);
- умножитель чисел с фиксированной и плавающей запятой (Multiplier) (А.2.1.1);
- арифметико-логическое устройство АЛУ (ALU) для операций с фиксированной и плавающей запятой, целочисленных и логических операций (А.2.1.2);
- 32-разрядный циклический сдвигатель (Shifter);
- внутренние шины (CPU1/CPU2 и REG1/REG2);
- арифметические устройства вспомогательных регистров (ARAU0, ARAU1) (А.2.1.3);
- регистровый файл ЦПУ (R0-R7, AR0-AR7, IR0, IR1, Others registers) (А.2.1.4);
- организация памяти (подраздел А.2.2);
- ОЗУ, ПЗУ и кэш (А.2.2.1);
- карты памяти (А.2.2.2);
- режимы адресации памяти (А.2.2.3);
- система команд (А.2.2.4);
- операции внутренней шины (подраздел А.2.3);
- операции внешней шины (подраздел А.2.4);
- периферия (подраздел А.2.5);
- модули таймера (А.2.5.1);
- последовательные порты (А.2.5.2);
- прямой доступ к памяти ПДП (DMA) (подраздел А.2.6);

А.2.1 Центральное процессорное устройство (ЦПУ)

ЦПУ ПЦОС имеет регистровую архитектуру и состоит из следующих компонентов:

- умножителя чисел с фиксированной (24 бит) и плавающей (32 бит) запятой (Multiplier);
- АЛУ, выполняющего одноцикловые арифметические операции над числами с плавающей (40 бит) и фиксированной (32 бит) запятой, а также 32-битные логические операции (ALU);
- 32-разрядного циклического сдвигателя (Shifter), внутренней шины (CPU1/CPU2 и REG1/REG2);
- двух выделенных, работающих независимо от АЛУ арифметических устройств вспомогательных регистров (ARAU0, ARAU1);
- регистрового блока ЦПУ (R0-R7, AR0-AR7, IR0, IR1, Others registers).

Структурная схема процессорного ядра ИС 1867ВН016 приведена на рисунке А.2.1.

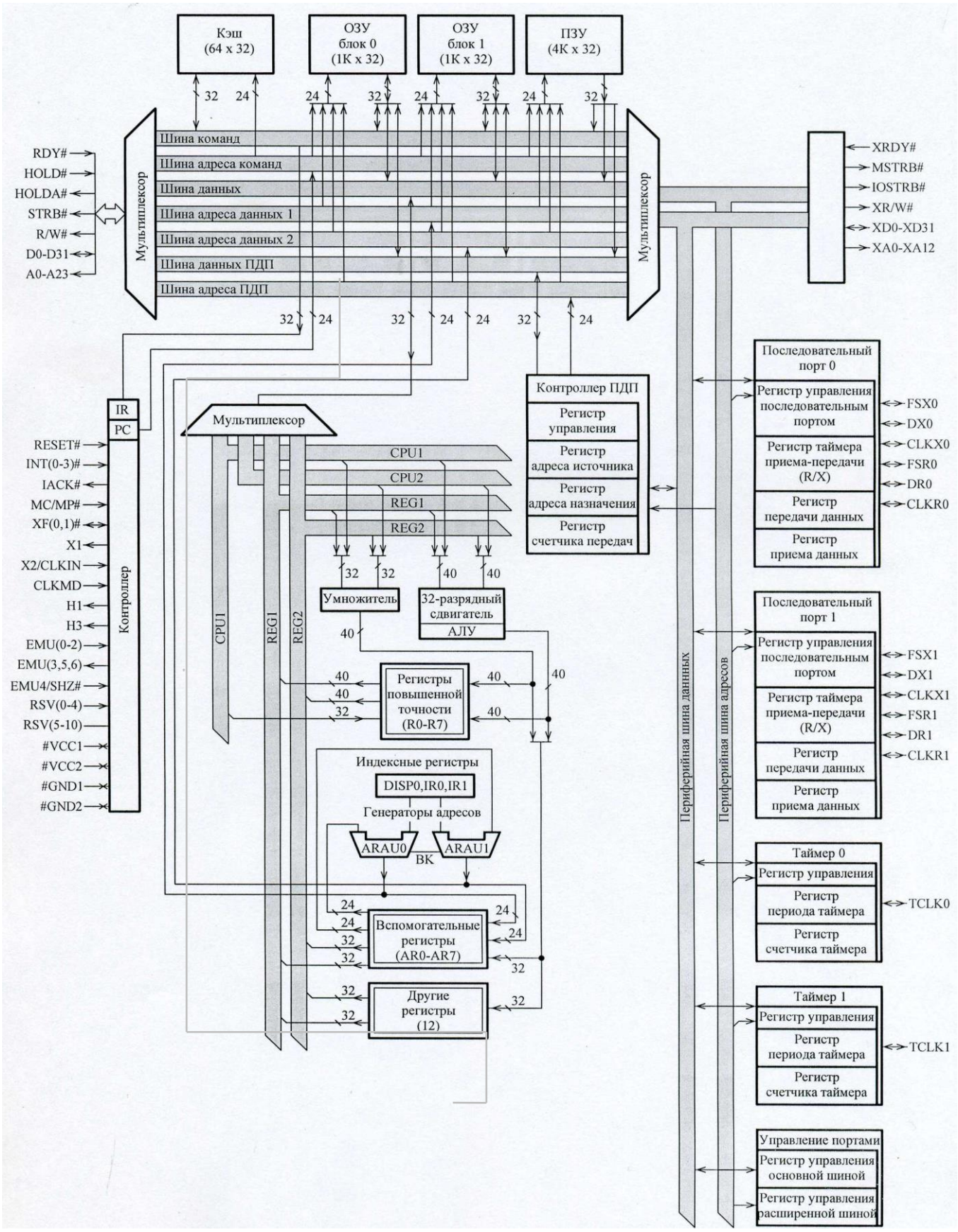


Рисунок А.2.1 – Структурная схема процессорного ядра ИС 1867ВН016

A.2.1.1 Умножитель (Multiplier)

Умножитель выполняет одноцикловые умножения над 24-разрядными целыми значениями и 32-разрядными значениями с плавающей запятой. Использование параллельных команд позволяет выполнять умножение одновременно (в одном цикле) с операциями АЛУ.

При выполнении умножения с плавающей запятой на вход подаются 32-разрядные значения с плавающей запятой, а на выходе получатся 40-разрядные значения с плавающей запятой. При выполнении целочисленного умножения на вход подаются 24-разрядные значения, а на выходе получается 32-разрядный результат. Более подробно информация о форматах данных и операциях с плавающей запятой изложена в разделе A.4 «Форматы данных и операции с плавающей запятой».

A.2.1.2 Арифметико-логическое устройство (АЛУ)

АЛУ выполняет одноцикловые операции с 32-разрядными целыми, 32-разрядными логическими и 40-разрядными значениями с плавающей запятой, включая одноцикловые целочисленные преобразования и преобразования значений с плавающей запятой. Результаты операции АЛУ всегда сохраняются в 32-разрядном целом или 40-разрядном формате с плавающей запятой. Циклический сдвигатель используется для сдвига до 32 бит влево или вправо за один цикл. Внутренние шины CPU1/CPU2 и REG1/REG2 обеспечивают выборку двух операндов из памяти и двух операндов из регистрового файла, таким образом, обеспечивая параллельное умножение и сложение/вычитание четырёх целочисленных или с плавающей запятой операндов в одном цикле.

A.2.1.3 Арифметические устройства вспомогательных регистров (ARAUn)

Два арифметических устройства вспомогательных регистров (ARAU0 и ARAU1) могут генерировать два адреса в одном цикле. ARAUn функционируют параллельно с умножителем и АЛУ. Они обеспечивают адресацию со смещением, индексную регистровую адресацию (с использованием IR0 и IR1), а также циклический и битреверсный режимы адресации. Все поддерживаемые режимы адресации подробно представлены в разделе A.5 «Адресация».

A.2.1.4 Регистровый файл ЦПУ

ПЦОС имеет 28 регистров в многопортовом регистровом файле, который непосредственно связан с ЦПУ. Все эти регистры могут хранить операнды для умножителя и АЛУ, а также могут быть использованы как регистры общего назначения. Однако регистры также имеют некоторые специальные функции. Например, восемь 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 целочисленные и логические регистры общего назначения. Оставшиеся регистры обеспечивают системные функции, такие как адресация, управление стеком, состояние процессора, прерывания и повторения блоков команд. В разделе A.6 «Управление выполнением программы» приведена более детальная информация и примеры управления стеком и использования регистров.

Имена регистров и их назначение приведены в таблице A.2.1 с кратким описанием функции каждого регистра или группы регистров. В разделе A.3 «Регистры ЦПУ, память и кэш» представлена более детальная информация о регистрах ЦПУ.

Таблица А.2.1 – Регистры ЦПУ

Имя регистра	Функциональное назначение регистра
R0	Регистр повышенной точности 0
R1	Регистр повышенной точности 1
R2	Регистр повышенной точности 2
R3	Регистр повышенной точности 3
R4	Регистр повышенной точности 4
R5	Регистр повышенной точности 5
R6	Регистр повышенной точности 6
R7	Регистр повышенной точности 7
AR0	Вспомогательный регистр 0
AR1	Вспомогательный регистр 1
AR2	Вспомогательный регистр 2
AR3	Вспомогательный регистр 3
AR4	Вспомогательный регистр 4
AR5	Вспомогательный регистр 5
AR6	Вспомогательный регистр 6
AR7	Вспомогательный регистр 7
DP	Регистр указателя страницы данных
IR0	Индексный регистр 0
IR1	Индексный регистр 1
BK	Регистр размера блока
SP	Регистр указателя системного стека
ST	Регистр состояния
IE	Регистр разрешения прерываний ЦПУ/ПДП
IF	Регистр флагов прерываний ЦПУ
IOF	Регистр флагов ввода-вывода
RS	Регистр начального адреса повторений
RE	Регистр конечного адреса повторений
RC	Регистр счётчика повторений
PC	Регистр счётчика команд

Регистры повышенной точности (R0-R7) имеют возможность хранения и выполнения операций с 32-разрядными целыми и 40-разрядными значениями с плавающей запятой. Любая инструкция, содержащая значения с плавающей запятой, использует разряды 39 – 0. Если операнды являются целыми со знаком или беззнаковыми, используются только разряды 31-0, разряды 39-32 остаются без изменений. Это относится ко всем операциям сдвига. В разделе А.4 «Форматы данных и операции с плавающей запятой» указаны форматы регистров повышенной точности для значений с плавающей запятой и целочисленных значений.

32-разрядные вспомогательные регистры (AR0-AR7) могут быть доступны из ЦПУ и модифицированы двумя арифметическими устройствами вспомогательных регистров (ARAUn). Основной функцией вспомогательных регистров является генерация 24-разрядных адресов. Они также могут быть использованы для выполнения различных функций, таких как циклические счётчики или как 32-разрядные регистры общего пользования, которые могут быть модифицированы умножителем и АЛУ. В разделе А.5 «Адресация» дана подробная информация и примеры использования вспомогательных регистров для адресации.

Указатель страницы данных (DP) является 32-разрядным регистром, восемь младших разрядов значащих разрядов (LSBs) которого, используются в режиме прямой адресации для выбора страницы данных. Размер страницы данных – 64K слов, всего имеется 256 страниц.

32-разрядные индексные регистры (IR0 и IR1) используются арифметическими устройствами вспомогательных регистров (ARAUn) для вычисления индексированного адреса. В разделе A.5 «Адресация» приведены примеры использования индексных регистров в различных режимах адресации.

32-разрядный регистр размера блока (BK) используется ARAUn при циклической адресации для указания 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. Для получения подробной информации об управлении системным стеком необходимо посмотреть подраздел A.5.5 «Управление системным стеком и стеком пользователя».

Регистр состояния (ST) содержит общую 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. При загрузке регистра состояния, поразрядная замена выполняется над всем его текущим содержимым, независимо от состояния каких-либо разрядов в операнде источника. Поэтому, после загрузки содержимое регистра состояния тождественно равно содержимому операнда источника. Это позволяет легко сохранять и восстанавливать содержимое регистра состояния. В таблице A.3.2 приведён список разрядов регистра состояния, их наименование и функции.

Регистр разрешения прерываний ЦПУ/ПДП – это 32-разрядный регистр. Битами разрешения прерывания ЦПУ являются разряды 10-0. Битами разрешения прерывания ПДП являются разряды 26-16. Единица в бите регистра разрешения прерываний разрешает соответствующие прерывания. Ноль запрещает соответствующие прерывания. В A.3.1.8 приведено описание разрядов этого регистра. Регистр флагов прерываний ЦПУ (IF) также является 32-разрядным регистром (см. A.3.1.9). Единица в бите регистра флага прерываний показывает, что соответствующее прерывание установлено. Ноль показывает отсутствие прерывания.

Регистр флагов ввода-вывода (IOF) управляет функцией специализированных внешних выводов XF0 и XF1. Эти выводы могут быть установлены в качестве входа или выхода, из них также может быть считана и записана информация. См. A.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) – 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.

A.2.2 Организация памяти

Общее адресное пространство ПЦОС – 16M (миллионов) 32-разрядных слов. Программа, данные и область ввода-вывода содержатся внутри этого 16M 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.

A.2.2.1 ОЗУ, ПЗУ и кэш

Блок ПЗУ – 4K × 32 бит. Каждый из блоков ОЗУ и ПЗУ поддерживает два обращения в одном цикле. Наличие раздельных шин команд, шин данных и шин ПДП обеспечивает параллельно: выборку программы, чтение и запись данных, и работу ПДП. Например, ЦПУ может обращаться к двум значениям данных в одном блоке ОЗУ и выполнять выборку внешней программы параллельно с загрузкой ПДП другого блока ОЗУ – все внутри одного цикла. 64×32-разрядный кэш команд обеспечивает хранение часто повторяющихся фрагментов кода, что значительно уменьшает число необходимых внешних обращений. Эта операция допускается для кодов, сохраняемых во внешней, как правило, менее быстродействующей памяти. Внешние шины в этом случае свободны для использования ПДП, выборки внешней памяти или для использования другими устройствами в системе. В разделе A.3 «Регистры ЦПУ, память и кэш» приведена более подробная информация о памяти и командах кэш.

A.2.2.2 Карты памяти

Карта памяти зависит от того, в каком режиме работает процессор, в режиме микропроцессора (MCBL/MP# = 0) или микрокомпьютера (MCBL/MP# = 1). Карты памяти для этих режимов очень похожи. В режиме микропроцессора 4K внутреннего ПЗУ не картированы в карту памяти ядра. Область от 0h до 0BFh содержит векторы прерываний, системные прерывания (векторы переходов) и зарезервированные адреса, каждый из которых доступен через порт внешней памяти (STRB# активен). Область от 0C0h до 7F_FFFFh также доступна через порт внешней памяти. В режиме микрокомпьютера 4K внутреннего ПЗУ картированы в адресах 0h - 0FFFh. 192 адреса (от 0h до BFh) внутри этого блока предназначены для векторов прерываний, векторов системных прерываний и зарезервированной области. Адреса от 1000h до 7F_FFFFh доступны через порт внешней памяти (STRB# активен).

В подразделе A.3.2 «Память» дана более подробная информация о картах памяти.

Будьте внимательны! Обращение к зарезервированной области памяти приведёт к непредсказуемым результатам.

А.2.2.3 Режимы адресации памяти

ПЦОС содержит базовый набор команд общего назначения, таких как арифметические команды, которые предназначены для цифровой обработки сигналов и других интенсивных вычислительных применений. В разделе А.5 «Адресация» дана более подробная информация об адресации.

ПЦОС имеет пять групп режимов адресации. Внутри группы могут быть использованы шесть типов адресации:

1) основные режимы адресации:
   - регистровый. Операнд хранится в регистре ЦПУ;
   - короткая непосредственная. Операнд – 16-разрядное непосредственное значение;
   - прямая. Операнд – содержимое 24-разрядного адреса;
   - косвенная. Вспомогательный регистр указывает адрес операнда;
2) трёхоперандные режимы адресации:
   - регистровый. Такой же, как для основных режимов адресации;
   - косвенный. Такой же, как для основных режимов адресации;
3) режимы параллельной адресации:
   - регистровый. Операнд – регистр повышенной точности;
   - косвенный. Такой же, как для основных режимов адресации;
4) режим длинной непосредственной адресации:
   - длинная непосредственная. Операнд – 24-разрядная непосредственная величина;
5) режимы адресации условных переходов:
   - регистровый. Такой же, как для основных режимов адресации;
   - относительно счётчика команд PC. 16-разрядов смещения прибавляется к PC.

А.2.2.4 Система команд

В таблице А.2.2 приведён список команд ПЦОС, составленный в алфавитном порядке. Каждая строка таблицы содержит мнемонику, описание и работу команды. В разделе А.10 «Команды языка ассемблер» приведён функциональный список команд и описание каждой команды.

Таблица А.2.2 – Система команд

Мнемоника	Описание	Операция
1	2	3
ABSF	Абсолютное значение числа с плавающей запятой	|src| → Rn
ABSI	Абсолютное значение целого числа	|src| → Dreg
ADDC	Сложение целых с переносом	src + Dreg + C → Dreg
ADDC3	Сложение целых с переносом (3 операнда)	src + src2 + C → Dreg
ADDF	Сложение значений с плавающей запятой	src + Rn → Rn
ADDF3	Сложение значений с плавающей запятой (3 операнда)	src1 + src2 → Rn
ADDI	Сложение целых	src + Dreg → Dreg
ADDI3	Сложение целых (3 операнда)	src1 + src2 + → Dreg
AND	Поразрядное логическое И (AND)	Dreg AND src → Dreg

Продолжение таблицы А.2.2

1	2	3
AND3	Поразрядное логическое И (AND) (3 операнда)	src1 AND src2 → Dreg
ANDN	Поразрядное логическое И (AND) с дополнением	Dreg AND src → Dreg
ANDN3	Поразрядное логическое И (AND с дополнением (3 операнда))	src1 AND src2 → Dreg
ASH	Арифметический сдвиг	Если count ≥ 0: (Сдвиг Dreg влево по счётчику) → Dreg. Иначе: (Сдвиг Dreg вправо по счётчику) → Dreg.
ASH3	Арифметический сдвиг (3 операнда)	Если count ≥ 0: (Сдвиг src влево по счётчику) →Dreg. Иначе: (Сдвиг src вправо по счётчику) →Dreg.
Bcond	Условный переход (стандартный)	Если cond = «истина»: Если Csrc – регистр, то Csrc → PC. Если Csrc – значение, то Csrc+PC → PC. Иначе: PC+1 → PC.
BcondD	Условный переход (с задержкой)	Если cond = «истина»: Если Csrc – регистр, то Csrc → PC. Если Csrc – значение, то Csrc+PC+3 → PC. Иначе: PC+1 → PC.
BR	Безусловный переход (стандартный)	Значение → PC
BRD	Безусловный переход (задержанный)	Значение → PC
CALL	Вызов подпрограммы	PC + 1 → TOS. Значение → PC.
CALLcond	Условный вызов подпрограммы	Если cond = «истина»: PC + 1 → TOS. Если Csrc регистр, Csrc → PC. Если Csrc значение, Csrc+PC → PC. Иначе: PC+1 →PC.
CMPF	Сравнение значений с плавающей запятой	Устанавливает флаги в Rn – src
CMPF3	Сравнение значений с плавающей запятой (3 операнда)	Устанавливает флаги в src1 – src2
CMPI	Сравнение целых	Устанавливает флаги в Dreg – src
CMPI3	Сравнение целых	Устанавливает флаги в src1 – src2 (3 операнда)
DBcond	Декремент и условный переход (стандартный)	ARn – 1 → ARn Если cond = «истина» и ARn≥ 0: Если Csrc регистр, Csrc→PC. Если Csrc значение, Csrc+PC→ PC. Иначе: PC+1 → PC.
DBcondD	Декремент и условный переход (задержанный)	ARn – 1→ ARn Если cond = «истина» и ARn≥0: Если Csrc регистр, Csrc → PC. Если Csrc значение, Csrc+PC+3 → PC. Иначе: PC + 1 → PC.
FIX	Преобразование значений с плавающей запятой в целое	Fix (src) → Dreg
FLOAT	Преобразование целых значений в значения с плавающей запятой	Float (src) → Rn
IACK	Подтверждение прерывания	Пустое чтение src IACK# переключается в 0, затем в 1.
IDLE	Ожидание прерывания	PC + 1 → PC. Ожидание следующего прерывания.
LDE	Загрузка значения экспоненты с плавающей запятой	src (exponent) → Rn (exponent)

Продолжение таблицы А.2.2

1	2	3
LDF	Загрузка значения с плавающей запятой	src → Rn
LDFcond	Условная загрузка значения с плавающей запятой	Если cond = «истина», src → Rn. Иначе: Rn без изменений.
LDFI	Загрузка значения с плавающей запятой, блокировка. Сигнализирует об операции блокировки	src → Rn
LDI	Загрузка целого	src → Dreg
LDIcond	Условная загрузка целых	Если cond = «истина», src→ Dreg. Иначе: Dreg без изменений.
LDII	Загрузка целых, блокировка. Сигнализирует о команде блокировки	src → Dreg
LDM	Загрузка мантиссы с плавающей запятой	src (mantissa) → Rn (mantissa)
LSH	Логический сдвиг	Если count ≥ 0: (Dreg, сдвинутый влево по счётчику) → Dreg. Иначе: (Dreg, сдвинутый вправо по счётчику) → Dreg.
LSH3	Логический сдвиг (3 операнда)	Если count ≥ 0: (src, сдвинутый влево по счётчику) → Dreg. Иначе: (src, сдвинутый вправо по счётчику) → Dreg.
MPYF	Умножение значений с плавающей запятой	src × Rn → Rn
MPYF3	Умножение значений с плавающей запятой (3 операнда)	src1 × src2 → Rn
MPYI	Умножение целых	src × Dreg → Dreg
MPYI3	Умножение целых (3 операнда)	src1 × src2 → Dreg
NEGB	Отрицание целого с заёмом	0 – src – C → Dreg
NEGF	Отрицание значения с плавающей запятой	0 – src → Rn
NEGI	Отрицание целого	0 – src → Dreg
NOP	Нет операции	Изменяется ARn, если требуется
NORM	Нормирование значений с плавающей запятой	Нормализованный (src) → Rn
NOT	Поразрядное логическое дополнение	src → Dreg
OR	Поразрядное логическое OR (ИЛИ)	Dreg OR src → Dreg
OR3	Поразрядное логическое – OR (ИЛИ) (3 операнда)	src1 OR src2 →Dreg
POP	Выталкивание целого из стека	*SP – – → Dreg
POPF	Выталкивание значений с плавающей запятой из стека	*SP – – → Rn
PUSH	Загрузка в стек целого	Sreg → * + + SP
PUSHF	Загрузка значений с плавающей запятой в стек	Rn → * + + SP
RETIcond	Условный выход из прерывания	Если cond = «истина» или опущено: *SP – – → PC 1 →ST(GIE). Иначе: продолжать.
RETScond	Условный выход из подпрограммы	Если cond = «истина» или опущено: *SP – – → PC. Иначе: продолжать.
RND	Округление значения с плавающей запятой	Round (src) → Rn
ROL	Циклический сдвиг влево	Dreg, сдвинутый влево на 1 бит → Dreg
ROLC	Циклический сдвиг влево через перенос	Dreg, сдвинутый влево на 1 бит через перенос → Dreg
ROR	Циклический сдвиг вправо	Dreg, сдвинутый вправо на 1 бит → Dreg
RORC	Циклический сдвиг вправо	Dreg, сдвинутый вправо на 1 бит через перенос → Dreg

Окончание таблицы А.2.2

1	2	3
RPTB	Повторение блока команд	src → RE; 1 → ST (RM); следующий PC → RS.
RPTS	Повторение одной инструкции	src → RE; 1 → ST (RM); Следующий PC → RS; следующий PC → RE.
SIGI	Сигнал блокировки сигнализирует об операции блокировки	Ожидание подтверждения блокировки. Очистка блокировки.
STF	Сохранение значения с плавающей запятой	Rn → Daddr.
STFI	Сохранение значения с плавающей запятой, блокировка	Rn → Daddr. Сигнализирует о завершении операции блокировки.
STI	Сохранение целого	Sreg → Daddr
STII	Сохранение целых, блокировка	Sreg → Daddr. Сигнализирует о завершении операции блокировки.
SUBB	Вычитание целых с заёмом	Dreg – src – C → Dreg
SUBB3	Вычитание целых с заёмом (3 операнда)	src1 – src2 – C → Dreg
SUBC	Условное вычитание целых	Если Dreg – src ≥ 0: [(Dreg – src) << 1] или 1 → Dreg. Иначе: Dreg << 1 → Dreg.
SUBF	Вычитание значений с плавающей запятой	Rn – src → Rn
SUBF3	Вычитание значений с плавающей запятой (3 операнда)	src1 – src2 → Rn
SUBI	Вычитание целых	Dreg – src →Dreg
SUBI3	Вычитание целых (3 операнда)	src1 – src2 → Dreg
SUBRB	Вычитание целых в обратном порядке с заёмом	src – Dreg – C → Dreg
SUBRF	Вычитание значений с плавающей запятой в обратном порядке	src – Rn → Rn
SUBR	Вычитание целых в обратном порядке	src – Dreg → Dreg
SWI	Программное прерывание	Выполняет последовательно прерывания эмулятора
TRAPcond	Условное системное прерывание	Если cond = «истина» или опущено: Следующий PC → *++ SP; Вектор trap N → PC. Иначе: продолжать.
TSTB	Проверка битовых полей	Dreg AND src
TSTB3	Проверка битовых полей (3 операнда)	src1 AND src2
XOR	Поразрядное исключающее ИЛИ	Dreg XOR src →Dreg
XOR3	Поразрядное исключающее ИЛИ (3 операнда)	src1 XOR src2 →Dreg

Примечание – Применяемые условные обозначения:

- src – основные режимы адресации;
- src1 – трёхоперандные режимы адресации;
- src2 – трёхоперандные режимы адресации;
- Csrc – режимы адресации условных переходов;
- Sreg – адрес регистра (любой регистр);
- count – величина сдвига (основные режимы адресации);
- SP – указатель стека;
- GIE – регистр общего разрешения прерываний;
- RM – бит режима повторения;
- PC – счётчик команд;
- TOS – вершина стека.
- C – бит переноса;
- Dreg – адрес регистра (любой регистр);
- Rn – адрес регистра (R0-R7);
- Daddr – адрес назначения памяти;
- ARn – вспомогательный регистр n (AR0-AR7);
- Addr – 24-разрядный непосредственный адрес (метка);
- Cond – код условия;
- ST – регистр состояния;
- RS – регистр начального адреса повторений;
- RE – регистр конечного адреса повторений.

Таблица А.2.3 – Система параллельных команд

Мнемоника	Описание	Функция
1	2	3
Параллельные арифметические команды с сохранением		
ABSF ||STF	Абсолютное значение величины с плавающей запятой	|src2| → dst1 || src3 → dst2
ABSI ||STI	Абсолютное значение целого	|src2| → dst1 || src3 → dst2
ADDF3 ||STF	Сложение значений с плавающей запятой	src1 + src2 → dst1 || src3 → dst
ADDI3 ||STI	Сложение целых	src1 + src2 → dst1 || src3 → dst2
AND3 ||STI	Поразрядное логическое И (AND)	src1 AND src2 → dst1 || src → dst2
ASH3 ||STI	Арифметический сдвиг	Если count ≥ 0: src2 << count → dst1 || src3 → dst2 Иначе: src2 >> |count| → dst1 || src3 → dst2
FIX ||STI	Преобразование значений с плавающей запятой в целое	Fix (src2) → dst1 ||src3 – dst2
FLOAT ||STF	Преобразование целого в значение с плавающей запятой	Float (src2) – dst1 || src3 → dst2
LDF ||STF	Загрузка значений с плавающей запятой	src2 → dst1 || src3 → dst2
LDI ||STI	Загрузка целых значений	src2 → dst1 || src3 → dst2
LSH3 ||STI	Логический сдвиг	Если count ≥ 0: src2 << count → dst1 || src3 → dst2 Иначе: src2 >> |count| → dst1 || src3 → dst2
MPYF3 ||STF	Умножение значений с плавающей запятой	src1 × src2 → dst1 || src3 → dst2
MPYI3 ||STI	Умножение целых	src1 × src2 → dst1 || src3 → dst2
NEGF ||STF	Отрицание значения с плавающей запятой	0 – src2 → dst1 || src3 → dst2
NEGI ||STI	Отрицание целого	0 – src2 → dst1 || src3 → dst2
NOT ||STI	Дополнение	src1 → dst1 || src3 → dst2
OR3 ||STI	Поразрядное логическое ИЛИ (OR)	src1 OR src2 → dst1 || src3 → dst2
STF ||STF	Сохранение значений с плавающей запятой	src1 → dst1 || src3 → dst2
STI ||STI	Сохранение целых	src1 → dst1 || src3 → dst2
SUBF3 ||STF	Вычитание значений с плавающей запятой	src1 – src2 → dst1 || src3 → dst2
SUBI3 ||STI	Вычитание целых	src1 – src2 → dst1 || src3 → dst2
XOR3 ||STI	Поразрядное исключающее ИЛИ (XOR)	src1 XOR src2 → dst1 || src3 → dst2
Команды параллельной загрузки		
LDF ||LDF	Загрузка значений с плавающей запятой	src2 → dst1|| src4 → dst2
LDI ||LDI	Загрузка целых	src2 → dst1 || src4 → dst2

Окончание таблицы А.2.3

1	2	3
Команды параллельного умножения и сложения/вычитания		
MPYF3 ||ADDF3	Умножение и сложение значений с плавающей запятой	op1 × op2 → op3 || op4 + op5 → op6
MPYF3 ||SUBF3	Умножение и вычитание значений с плавающей запятой	op1 × op2 → op3 || op4 – op5 → op6
MPYI3 ||ADDI3	Умножение и сложение целых	op1 × op2 → op3 || op4 + op5 → op6

Примечания

1 Применяемые условные обозначения:

- src1 – регистровая адресация (R0-R7);
- src3 – регистровая адресация (R0-R7);
- dst1 – регистровая адресация (R0-R7);
- op3 – регистровая адресация (R0 или R1);
- src2 – косвенная адресация (disp = 0, 1, R0, IR1);
- src4 – косвенная адресация (disp = 0, 1, IR0, IR1);
- dst2 – косвенная адресация (disp = 0, 1, IR0, IR1);
- op6 – регистровая адресация (R2 или R3).

2 op1, op2, op4, op5 – два из этих операндов могут быть определены с использованием регистровой адресации, и два могут быть определены с использованием косвенной.

А.2.3 Работа внутренней шины

Основой высокой производительности ПЦОС служит организация внутренних шин и возможность параллельной работы. Разделение шин команд (PADDR и PDATA), шин данных (DADDR1, DADDR2 и DDATA) и шин ПДП (DMAADDR и DMADATA) обеспечивает параллельную выборку программ, доступ к данным и обращение ПДП. Эти шины связывают все физические области (внутреннюю память, внешнюю память и внутреннюю периферию), поддерживаемые ядром. Регистр команд (IR) связан с 32-разрядной шиной данных программы (PDATA). Эти шины могут выбирать однокомандное слово в каждом цикле.

24-разрядные адресные шины (DADDR1 и DADDR2) и 32-разрядная шина данных (DDATA) обеспечивают два обращения к данным памяти в каждом машинном цикле. Шина DDATA передаёт данные по шинам CPU1 и CPU2. Шины CPU1 и CPU2 могут передавать два операнда данных памяти в умножитель, АЛУ и в регистровый файл в каждом машинном цикле. Также в ЦПУ есть регистровые шины REG1 и REG2, которые могут передавать два значения данных из регистрового файла в умножитель и в АЛУ в каждом машинном цикле. Контроллер ПДП снабжён 24-разрядной адресной шиной (DMAADDR) и 32-разрядной шиной данных (DMADATA). Эти шины дают возможность ПДП выполнять обращение к памяти параллельно с обращением к памяти, имеющих место на шинах данных и команд.

А.2.4 Работа внешней шины

ПЦОС поддерживает два внешних интерфейса: основной и расширенный. Оба интерфейса состоят из 32-разрядной шины данных и набора управляющих сигналов. Основной интерфейс имеет 24-разрядную адресную шину, а расширенный – 13-разрядную адресную шину. Обе шины могут быть использованы для адресации внешней памяти программ/данных или области ввода-вывода. Шины также имеют внешний сигнал RDY# для формирования состояния ожидания. Дополнительно состояния ожидания могут быть включены программно. В разделе А.7 «Работа с внешней шиной» дана более подробная информация об операциях внешних шин.

A.2.4.1 Внешние прерывания

ПЦОС имеет четыре внешних прерывания (INT3# – INT0#), некоторое количество внутренних прерываний и немаскируемый внешний сигнал сброса RESET#. Они могут быть использованы для прерывания ЦПУ или ПДП. Когда ЦПУ обрабатывает прерывание, вывод IACK# может быть использован для сигнализации подтверждения внешнего прерывания. Подраздел A.6.5 «Операция сброса» посвящён обработке RESET# и прерываний.

A.2.4.2 Сигнализация об инструкциях блокировки

Два внешних флага ввода-вывода XF0 и XF1 могут быть использованы как контакты ввода или вывода в управляющем программном обеспечении. Эти выводы также используются операциями блокировки ПЦОС. Группа команд блокировки обеспечивает мультипроцессорную связь, см. подраздел A.6.4 «Операции блокировки» – примеры использования команд блокировки.

A.2.5 Периферия

Все периферийные устройства ПЦОС управляются регистрами, картированными в памяти, через специальную периферийную шину. Эта шина соединена с 32-разрядной шиной данных и 24-разрядной адресной шиной. Периферийная шина обеспечивает прямую связь с периферийными устройствами. Периферийные устройства ПЦОС включают два таймера и два последовательных порта. В разделе A.8 «Периферийные устройства» приводится более подробная информация о периферийных устройствах.

A.2.5.1 Модули таймера

Два модуля таймера являются 32-разрядными универсальными счётчиками времени/числа событий с двумя режимами сигнализации и внешней или внутренней синхронизацией. Каждый таймер имеет контакт ввода-вывода, который может быть использован как вход синхронизации таймера или как выходной сигнал, управляемый таймером. Контакт может также быть использован как контакт ввода-вывода общего назначения.

A.2.5.2 Последовательные порты

Два последовательных порта полностью независимы. Они одинаковы и имеют комплементарный набор регистров управления, контролирующих каждый из портов. Каждый последовательный порт может быть установлен для пересылок 8-, 16-, 24- или 32-разрядными словами данных. Синхронизация для каждого последовательного порта может быть внутренней или внешней. Обеспечивается внутренняя генерация синхросигнала с делением частоты. Выводы последовательного порта можно представить как контакты ввода-вывода общего назначения. Последовательные порты могут также быть сконфигурированы как таймеры. Специальный режим квитирования передачи позволяет связать несколько ПЦОС через последовательные порты с гарантированной синхронизацией.

A.2.6 Прямой доступ к памяти (ПДП)

Контроллер прямого доступа к памяти (ПДП) ядра может считывать или записывать любую область памяти без взаимодействия с работой ЦПУ. Поэтому ПЦОС может работать с медленным внешним интерфейсом памяти и периферией без уменьшения производительности ЦПУ. Контроллер ПДП содержит свой собственный генератор адреса, регистры исходного адреса и назначенного адреса и счётчик переданных слов. Соответствующие шины адреса и данных ПДП позволяют минимизировать конфликты между ЦПУ и контроллером ПДП. Работа ПДП состоит из пересылок блока или одного слова в память или из памяти. В подразделе A.8.3 «Контроллер ПДП» приведена более подробная информация о работе контроллера ПДП.

A.3 Регистры ЦПУ, память и кэш

Регистровый файл ЦПУ содержит 28 регистров, которые могут использоваться умножителем и АЛУ. В регистровый файл включены вспомогательные регистры, регистры повышенной точности и индексные регистры. Регистры в регистровом файле ЦПУ поддерживают адресацию, операции с плавающей запятой, управление стеком, состояние процессора, повторение блока и прерывания.

ПЦОС имеет общее адресное пространство памяти в 16M 32-разрядных слов, включая области программ, данных и ввода-вывода.

Два блока ОЗУ по 1K × 32 разрядов каждый и блок ПЗУ размером 4K × 32 разрядов обеспечивают два обращения ЦПУ в одном цикле.

Карты памяти в режиме микрокомпьютера и микропроцессора аналогичны, за исключением того, что внутреннее масочное ПЗУ ядра не используется в режиме микропроцессора.

64 × 32-разрядный кэш команд хранит часто повторяющиеся фрагменты программы. Это сильно уменьшает количество необходимых обращений к памяти вне ядра и позволяет хранить программу вовне в сравнительно медленной недорогой памяти. Три разряда регистра состояния ЦПУ обеспечивают управление очисткой, разрешением или «замораживанием» кэш.

В данном разделе подробно описаны каждый из регистров ЦПУ, карты памяти и команды кэш. Основные темы в данном разделе, следующие:

1) регистровый файл ЦПУ (подраздел A.3.1):
   - регистры повышенной точности (R0-R7):
   - вспомогательные регистры (AR0-AR7);
   - индексные регистры (IR0, IR1);
   - регистр размера блока (BK);
   - указатель страницы данных (DP);
   - указатель системного стека (SP);
   - регистр состояния (ST);
   - регистр разрешения прерываний ЦПУ/ПДП (IE);
   - регистр флага прерываний ЦПУ (IF);
   - регистр флага ввода-вывода (IOF);
   - счётчик повторений (RC) и регистры повторений блока (RS, RE);
   - счётчик команд (PC);

2) память (подраздел А.3.2):
- карта памяти;
- карта периферийной шины;
- карта сброса/прерывания/системного прерывания;

3) кэш команд (подраздел А.3.3):
- архитектура кэш;
- алгоритм кэш;
- управляющие разряды кэш.

А.3.1 Набор регистров ПЦОС

Процессорное ядро содержит 28 регистров в мультипортовом регистровом файле, который непосредственно связан с ЦПУ. Счётчик команд PC не включен в эти 28 регистров. Все эти регистры могут обрабатываться умножителем и АЛУ и могут использоваться как регистры общего назначения. Однако регистры имеют специальные функции, для которых они предназначены. Например, восемь регистров повышенной точности наиболее подходят для хранения результатов повышенной точности с плавающей запятой.

Восемь вспомогательных регистров обеспечивают различные режимы косвенной адресации и могут быть использованы как 32-разрядные целочисленные и логические и регистры общего назначения. Оставшиеся регистры обеспечивают такие системные функции, как адресация, управление стеком, состояние процессора, прерывания и повторение блока.

В разделе А.5 «Адресация» дана более подробная информация и примеры использования регистров ЦПУ для адресации.

Сводный список набора регистров ПЦОС приведён в таблице А.3.1

Таблица А.3.1 – Набор регистров ПЦОС

Мнемоника	Название регистра
R0	Регистр повышенной точности 0
R1	Регистр повышенной точности 1
R2	Регистр повышенной точности 2
R3	Регистр повышенной точности 3
R4	Регистр повышенной точности 4
R5	Регистр повышенной точности 5
R6	Регистр повышенной точности 6
R7	Регистр повышенной точности 7
AR0	Вспомогательный регистр 0
AR1	Вспомогательный регистр 1
AR2	Вспомогательный регистр 2
AR3	Вспомогательный регистр 3
AR4	Вспомогательный регистр 4
AR5	Вспомогательный регистр 5
AR6	Вспомогательный регистр 6
AR7	Вспомогательный регистр 7
DP	Указатель страницы данных
IR0	Индексный регистр 0
IR1	Индексный регистр 1
BK	Регистр размера блока
SP	Указатель системного стека
ST	Регистр состояния
IE	Регистр разрешения прерываний ЦПУ/ПДП
IF	Флаги прерываний ЦПУ
IOF	Флаги ввода-вывода
RS	Регистр начального адреса повторений
RE	Регистр конечного адреса повторений
RC	Счётчик повторений
PC	Программный счётчик

А.3.1.1 Регистры повышенной точности (R0-R7)

Восемь регистров повышенной точности (R0-R7) могут хранить и оперировать с 32-разрядными целыми и 40-разрядными значениями с плавающей запятой. Эти регистры состоят из двух различных областей. Разряды 39-32 регистров повышенной точности предназначены для хранения экспоненты (e) значения с плавающей запятой. Разряды 31-0 хранят мантиссу значения с плавающей запятой. Разряд 31 – это знаковый разряд (s), разряды 30-0 – это дробная часть (f). Любая команда, которая содержит операнд с плавающей запятой, использует разряды 0-39.

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, использует только разряды 31-0. Разряды 39-32 остаются без изменений. Это верно для любых операций сдвига.

A.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 ARAUn. Основная функция вспомогательных регистров – это генерация 24-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 разделе A.5 «Адресация» дана более подробная информация и примеры использования вспомогательных регистров для адресации.

A.3.1.3 Указатель страницы данных (DP)

Указатель страницы данных (DP) – это 32-разрядный регистр. Восемь младших значащих разрядов указателя страницы данных используются в режиме прямой адресации как указатель на страницу адресуемых данных. Страница данных имеет длину 64K слов, общее количество страниц – 256. Разряды 31-8 зарезервированы и будут всегда равны 0.

A.3.1.4 Индексные регистры (IR0, IR1)

32-разрядные индексные регистры (IR0, IR1) используются арифметическими устройствами вспомогательных регистров ARAUn для индексирования адресов. В разделе A.5 «Адресация» дана подробная информация и примеры использования индексных регистров при адресации.

A.3.1.5 Регистр размера блока (BK)

32-разрядный регистр размера блока (BK) используется ARAUn при циклической адресации для точного определения размера блока данных, см. подраздел A.5.3 «Циклическая адресация».

A.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 прерываниями, системными прерываниями, вызовами, возвратами и командами PUSH, PUSHF, POP, POPF. При выталкивании из стека и проталкивании данных в стек выполняется прединкремент и постдекремент указателя стека на всех 32 разрядах. Однако только 24 младших значащих разряда используются как адрес. В подразделе A.5.5 «Управление системным стеком и стеком пользователя» дана информация об управлении системным стеком.

А.3.1.7 Регистр состояния (ST)

Регистр состояния (ST) содержит общую информацию, относящуюся к состоянию 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. При загрузке регистра состояния, поразрядная замена выполняется над всем его текущим содержимым, независимо от состояния каких-либо разрядов в операнде источника. Поэтому, после загрузки содержимое регистра состояния тождественно равно содержимому операнда источника. Это позволяет легко сохранять и восстанавливать содержимое регистра состояния. При системном сбросе в этот регистр записываются нули.

В таблице А.3.2 приведён список разрядов регистра состояния, их имена и функции.

Таблица А.3.2 – Описание разрядов регистра состояния ST

Разряд	Наименование	Функция
0*	C	Флаг переноса.
1*	V	Флаг переполнения.
2*	Z	Флаг нулевого значения.
3*	N	Флаг отрицательного значения.
4*	UF	Флаг потери значимости разрядов числа с плавающей запятой.
5*	LV	Фиксируемый флаг переполнения.
6*	LUF	Фиксируемый флаг потери значимости разрядов числа с плавающей запятой.
7	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: - целые положительные результаты с переполнением устанавливаются в значение наибольшего положительного 32-разрядного числа в дополнительном коде (7FFF_FFFFh); - целые отрицательные результаты устанавливаются в наименьшее отрицательное 32-разрядное число в дополнительном коде (8000_0000h).
8	RM	Флаг режима повторения. Если RM=1, то PC модифицируется или при повторении блока команд, или в режиме одиночного повторения.
9	Резервный	Считывается как 0.
10	CF	«Замораживание» кэш. Если CF=1, то кэш заморожен. Если кэш разрешён (CE=1), то выбор из кэш разрешён, 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.
11	CE	Разрешение кэш. CE=1 разрешает кэш, позволяя использовать кэш в соответствии с алгоритмом кэш LRU (наименее недавно использованный). CE=0 запрещает кэш, кэш не изменяется. Не происходит выборка из кэш. Эта функция используется для отладки системы. При сбросе в этот разряд заносится 0. Очистка кэш (CC=1) разрешается, если CE=0.
12	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.
13	GIE	Глобальное разрешение прерываний. Если GIE=1, то ЦПУ отвечает на разрешённое прерывание. Если GIE=0, то ЦПУ не отвечает на разрешённое прерывание.
14-15	Резервный	Считывается как 0.
16-31	Резервный	Значение не определено.
* Семь флагов условий (разряды ST6-ST0) описываются более подробно в подразделе А.10.2 «Коды условий и флаги».		

А.3.1.8 Регистр разрешения прерываний ЦПУ/ПДП (IE)

Регистр разрешения прерываний ЦПУ/ПДП (IE) – это 32-разрядный регистр. Разряды разрешения прерываний ЦПУ: 10-0. Разряды разрешения прерываний ПДП: 26-16. Единица в разряде регистра разрешения прерываний ЦПУ/ПДП разрешает, а ноль – запрещает соответствующее прерывание. При сбросе в этот регистр записываются нули. В таблице А.3.3 определены разряды регистра, имена разрядов регистра и функции разрядов.

Таблица А.3.3 – Описание разрядов регистра IE

Разряд	Наименование	Функция
0	EINT0	Разрешение внешнего прерывания 0 (ЦПУ)
1	EINT1	Разрешение внешнего прерывания 1 (ЦПУ)
2	EINT2	Разрешение внешнего прерывания 2 (ЦПУ)
3	EINT3	Разрешение внешнего прерывания 3 (ЦПУ)
4	EXINT0	Разрешение прерывания передачи последовательного порта 0 (ЦПУ)
5	ERINT0	Разрешение прерывания приёма последовательного порта 0 (ЦПУ)
6	EXINT1	Разрешение прерывания передачи последовательного порта 1 (ЦПУ)
7	ERINT1	Разрешение прерывания приёма последовательного порта 1 (ЦПУ)
8	ETINT0	Разрешение прерывания таймера 0 (ЦПУ)
9	ETINT1	Разрешение прерывания таймера 1 (ЦПУ)
10	EDINT	Разрешение прерывания контроллера DMA (ЦПУ)
11-15	резерв	Значения не определены
16	EINT0	Разрешение внешнего прерывания 0 (ПДП)
17	EINT1	Разрешение внешнего прерывания 1 (ПДП)
18	EINT2	Разрешение внешнего прерывания 2 (ПДП)
19	EINT3	Разрешение внешнего прерывания 3 (ПДП)
20	EXINT0	Разрешение прерывания передачи последовательного порта 0 (ПДП)
21	ERINT0	Разрешение прерывания приёма последовательного порта 0 (ПДП)
22	EXINT1	Разрешение прерывания передачи последовательного порта 1 (ПДП)
23	ERINT1	Разрешение прерывания при приёме последовательным портом 1 (ПДП)
24	ETINT0	Разрешение прерывания таймера 0 (ПДП)
25	ETINT1	Разрешение прерывания таймера 1 (ПДП)
26	EDINT	Разрешение прерывания контроллера ПДП (ПДП)
27-31	резерв	Неопределённые значения

А.3.1.9 Регистр флага прерываний ЦПУ (IF)

Единица в разряде регистра флага прерываний ЦПУ указывает на то, что соответствующее прерывание установлено. Единица в разряде регистра может быть уставлена программно, вызывая прерывание. 0 показывает, что соответствующее прерывание не установлено. Если 0 записан в разряд регистра флага прерываний, то соответствующее прерывание очищено. При сбросе в этот регистр записывается 0. В таблице А.3.4 приведены список разрядов регистра флага прерываний ЦПУ, их имена и функции.

Таблица А.3.4 – Описание разрядов регистра IF

Разряд	Имя	Функция
0	INT0	Флаг 0 внешнего прерывания
1	INT1	Флаг 1 внешнего прерывания
2	INT2	Флаг 2 внешнего прерывания
3	INT3	Флаг 3 внешнего прерывания
4	XINT0	Флаг прерывания при передаче последовательного порта 0
5	RINT0	Флаг прерывания при приёме последовательного порта 0
6	XINT1	Флаг прерывания при передаче последовательного порта 1
7	RINT1	Флаг прерывания при приёме последовательного порта 1
8	TINT0	Флаг прерывания таймера 0
9	TINT1	Флаг прерывания таймера 1
10	DINT	Флаг прерывания канала ПДП
11-31	Резерв	Значения не определены

A.3.1.10 Регистр флага ввода-вывода (IOF)

Регистр флага ввода-вывода (IOF) управляет функцией специальных внешних выводов XF0 и XF1. Эти выводы могут быть определены в качестве входа или выхода. По ним также могут быть считаны или переданы данные. При сбросе в регистр IOF заносится 0. Список разрядов, их имена и функции регистра приведены в таблице A.3.5.

Таблица A.3.5 – Описание разрядов регистра IOF

Разряд	Наименование	Функция
0	Зарезервированы	Читается как 0.
1	I/OXF0	Если I/OXF0=0, XF0 конфигурирован как контакт входа общего назначения. Если I/OXF0=1, XF0 конфигурирован как контакт выхода общего назначения.
2	OUTXF0	Вывод данных на XF0.
3	INXF0	Ввод данных на XF0. Запись не влияет.
4	Зарезервирован	Считывается как 0.
5	I/OXF1	Если I/OXF1=0, XF1 конфигурирован как контакт входа общего назначения. Если I/OXF1=1, XF1 конфигурирован как контакт выхода общего назначения.
6	OUTXF1	Вывод данных на XF1.
7	INXF1	Ввод данных на XF1. Запись не влияет.
8-31	Зарезервированы	Считываются как 0.

A.3.1.11 Счётчик повторений (RC) и регистры повторений блока (RS, RE)

Счётчик повторений (RC) – это 32-разрядный регистр, используемый для точного определения количества повторений блока кода при выполнении.

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

32-разрядный регистр повторений и адреса (RE) содержит адрес конца повторяющегося блока памяти программы при работе в режиме повторений.

A.3.1.12 Счётчик команд (PC)

Счётчик команд (PC) – это 32-разрядный регистр, содержащий адрес следующей выполняемой команды. Счётчик команд не является частью регистрового файла, он является регистром, который может изменяться командами в процессе выполнения программы.

A.3.1.13 Резервные разряды и совместимость

Чтобы добиться совместимости с будущими схемами 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.

А.3.2 Память

Общее адресное пространство памяти ПЦОС составляет 16М 32-разрядных слов, содержит область памяти программ данных и область ввода-вывода, что позволяет таблицам, коэффициентам, программным кодам или данным храниться либо в ОЗУ, либо в ПЗУ. В этом случае использование памяти может быть максимизировано и область памяти распределяется, как требуется.

ОЗУ имеет блоки 0 и 1, каждый из которых размером 1К×32 бит. Блок ПЗУ – 4К×32 бит. Каждый из блоков ОЗУ или ПЗУ может обеспечивать два обращения ЦПУ за один цикл. Отдельные шины команд и данных и ПДП обеспечивают параллельную выборку программ, чтение/запись данных и операции ПДП. Это подробно описано в разделе А.9 «Работа конвейера».

А.3.2.1 Карта памяти ПЦОС

Карта памяти зависит от того, в каком режиме работает процессор: в режиме микропроцессора (MCBL/MP# = 0) или в режиме микрокомпьютера (MCBL/MP# = 1). Карта памяти для этих режимов представлены в таблицах 4.1, 4.2 КФДЛ.431299.034ТО. Область от 80_0000h до 80_1FFFh картирована для расширенной шины. Если эта область доступна, то активен MSTRB#. Область адресов от 80_2000h до 80_3FFFh зарезервирована. Адреса от 80_4000h до 80_5FFFh картированы для расширенной шины. Если эта область доступна, то активен IOSTRB#. Адреса от 806000h до 80_7FFFh зарезервированы. Все картированные в памяти периферийные регистры размещены от 80_8000h до 80_97FFh. В обоих режимах блок 0 ОЗУ размещён в адресах от 80900h до 809FFh и блок 1 ОЗУ расположен в адресах от 80_9C00h до 80_9FFFh. Для внешнего порта памяти допустимо размещение от 80_A000h до FF_FFFFh (STRB# активен).

Резервные пространства области памяти ПЦОС и резервные адреса периферийных шин не должны читаться и записываться пользователем. В этом случае может произойти останов работы ПЦОС и потребуется перезапуск системы.

А.3.2.2 Карта векторов сброса, прерывания, системного прерывания

Адреса для векторов возврата, прерывания и системного прерывания распределены от 0h до 3Fh. Векторы, хранящиеся в этих областях, являются начальными адресами программ сброса, прерывания и системного прерывания. Например, при сбросе содержимое области памяти 0h (вектор прерывания) загружается в PC, и выполнение начинается с этого адреса.

Системные прерывания 28-31 зарезервированы и не могут быть использованы!

А.3.2.3 Карта периферийной шины

Распределённые в памяти периферийные регистры расположены начиная с адреса 80_8000h. Каждое периферийное устройство занимает область в 16 слов карты памяти. Области от 80_8010h до 80_801Fh и от 80_8070h до 80_97FFh зарезервированы.

A.3.3 Кэш команд

64 × 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 внешней памяти и позволяет хранить программу во внешней памяти. Внешние шины в этом случае свободны от выборки программы, что может быть использовано ПДП и другими элементами системы.

Кэш может работать в полностью автоматическом режиме без вмешательства пользователя. Используется алгоритм изменения кэш LRU (Least Recently Used) – откачка наименее часто используемых фрагментов программы, см. A.3.3.2.

A.3.3.1 Архитектура кэш

Кэш команд содержит 64 × 32-разрядных слов ОЗУ и разделён на два сегмента по 32 слова каждый. 19-разрядный регистр сегмента начального адреса (SSA) соединён с каждым сегментом. Каждому слову в кэш соответствует одноразрядный P (Present) флаг.

Когда ЦПУ запрашивает командное слово из внешней памяти, то проверяется, не содержится ли слово в кэш команд. 19 старших разрядов адреса команды используются для выбора сегмента, 5 младших разрядов определяют адрес слова команды внутри выбранного сегмента. 19 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. 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.

Стек LRU определяет, который из двух сегментов квалифицируется как наиболее давно использованный после каждого доступа к кэш; таким образом, стек содержит 0, 1 или 1, 0. Каждый раз при доступе к сегменту, номер сегмента удаляется из стека LRU и проталкивается в вершину стека LRU. Таким образом, число в вершине стека представляет собой номер последнего использованного сегмента, а число на дне стека – номер наиболее давно используемого 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, то для неё выбирается наиболее давно используемый сегмент. 32 флага P для заменяемого сегмента устанавливаются в 0, и в сегментный регистр SSA записываются 19 старших разрядов адреса команды.

А.3.3.2 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):

- первый случай – 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 проталкивается на вершину стека LRU, т. о. перемещая номер другого сегмента на дно стека;

- второй случай – 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:

1) командное слово считывается из памяти и копируется в кэш;

2) номер сегмента, внутри которого содержится слово, удаляется из стека LRU и проталкивается в вершину стека LRU, т. о. перемещая номер другого сегмента на дно стека;

3) устанавливается соответствующий флаг P;

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

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

2) регистр SSA для выбранного сегмента загружается 19 старшими разрядами адреса запрошенного командного слова;

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

4) номер сегмента, содержащего слово, удаляется из стека LRU и проталкивается в вершину стека LRU, т. о. 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 с помощью выравнивания кода программы по границе адреса в 32 слова. Это может быть сделано, если использовать директиву ALIGN при написании программ на языке ассемблера.

А.3.3.3 Управляющие разряды кэш

Три управляющих разряда кэш размещены в регистре состояния ЦПУ: разряд очистки кэш (CC), разряд разрешения кэш (CE) и разряд «замораживания» кэш (CF).

Разряд очистки кэш (CC). При записи 1 в разряд очистки CC аннулируются все вводы в кэш. Все флаги P в кэш очищаются. Разряд CC всегда очищается после очистки кэш. Поэтому он всегда читается как 0. При сбросе кэш очищается и в этот разряд заносится 0.

Разряд разрешения кэш (CE). При записи 1 в этот разряд кэш разблокируется. Когда кэш разблокирован, то он используется в соответствии с вышеописанным алгоритмом кэш. При записи 0 в разряд разрешения кэш – кэш блокируется, дополнения или модификация не могут быть выполнены. Не выполняются дополнения регистра SSA, не изменяются флаги P (пока CC не равно 1) и не изменяется стек LRU. При записи 1 в CC, когда кэш блокирован, кэш будет очищен и, таким образом, очищены флаги P. Когда кэш блокирован, выборка из кэш производиться не будет. При сбросе в этот разряд заносится 0.

Разряд «замораживания» кэш (CF). Когда CF=1 – кэш заморожен. Если к тому же кэш разблокирован, то выбор из кэш разрешён, но состояние кэш не изменяется. Регистр SSA не обновляется, P флаги не изменяются (пока CC=0) и стек LRU не изменится. Эта функция может быть использована для хранения часто используемых кодов резидентно в кэш. Если кэш заморожен, то при занесении 1 в CC происходит очистка кэш и, следовательно, очистка флагов P. В результате сброса в этот разряд заносится 0.

Таблица А.3.6 – Результат установки различных комбинаций разрядов CE и CF

CE	CF	Результат
0	0	кэш не разрешён
0	1	кэш не разрешён
1	0	кэш разрешён и не заморожен
1	1	кэш разрешён и заморожен

А.4 Форматы данных и операции с плавающей запятой

Организация данных в архитектуре ПЦОС поддерживает три основных типа данных: целые, целые без знака и числа в формате с плавающей запятой.

Термины целые и целые без знака будут считаться эквивалентными. ПЦОС поддерживает короткие и с одинарной точностью форматы для целых со знаком и без знака. Он также поддерживает 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 на ПЦОС позволяет производить операции с плавающей запятой со скоростью целочисленных, в то же время, предотвращая проблемы с переполнением, выравниванием операндов и другие общие проблемы целочисленных операций.

В данном разделе подробно обсуждаются форматы данных и операции с плавающей запятой, поддерживаемые ПЦОС, рассматриваются следующие основные темы:

- целочисленные форматы (подраздел А.4.1);
- целочисленные форматы целых без знака (подраздел А.4.2);
- форматы значений с плавающей запятой (подраздел А.4.3);
- умножение значений с плавающей запятой (подраздел А.4.4);
- сложение и вычитание значений с плавающей запятой (подраздел А.4.5);
- нормализация (подраздел А.4.6);
- округление (подраздел А.4.7);
- преобразование значений с плавающей запятой в целые (подраздел А.4.8);
- преобразование целых значений в значения с плавающей запятой (подраздел А.4.9).

А.4.1 Форматы целых

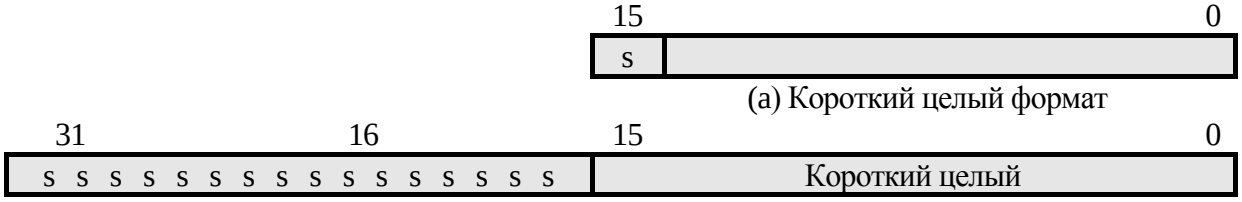
ПЦОС поддерживает два целочисленных формата: 16-разрядный короткий целый формат и 32-разрядный целый формат с одинарной точностью. Когда регистры повышенной точности применяются как целочисленные операнды, используются только разряды 31-0; разряды 39-32 остаются без изменения и не используются.

А.4.1.1 Короткий целочисленный формат

Короткий целый формат – это 16-разрядный в форме дополнения целый формат, используемый для непосредственных целых операндов. Для тех команд, которые содержат целые операнды, происходит расширение этого формата за счёт знака до 32 разрядов (смотри рисунок А.4.1). Диапазон целого числа Ni, представленный в коротком целом формате, равен:

$$-2^{15} \le Ni \le 2^{15} - 1 \qquad \text{(А.4.1)}$$

На рисунке А.4.1 – *s* – знаковый разряд.



(a) Короткий целый формат

(b) Расширение знака короткого целого формата

Рисунок А.4.1 – Короткий целый формат и расширение знака короткого целого

А.4.1.2 Целый формат с одинарной точностью

В целом формате с одинарной точностью целые числа представлены в форме дополнения. Диапазон целого числа *Np*, представленного в целом формате с одинарной точностью:

$$-2^{31} \le Np \le 2^{31} - 1 \qquad \text{(А.4.2)}$$

На рисунке А.4.2 приведён целый формат с одинарной точностью.

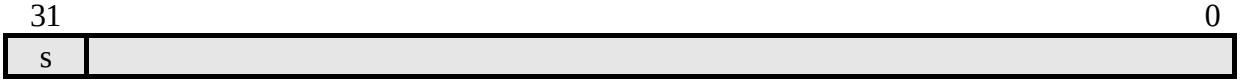


Рисунок А.4.2 – Целый формат с одинарной точностью

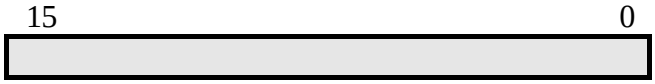
А.4.2 Форматы целых без знака

ПЦОС поддерживает два формата целых без знака: 16-разрядный короткий формат и 32-разрядный формат с одинарной точностью. В регистрах повышенной точности целые операнды без знака используют только разряды 31-0; разряды 39-32 остаются без изменений.

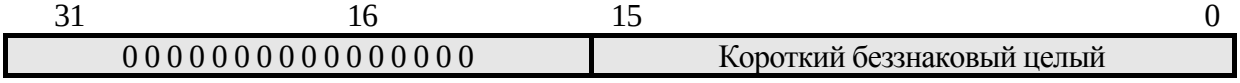
А.4.2.1 Короткий целочисленный формат без знака

На рисунке А.4.3 приведён 16-разрядный короткий формат без знака, используемый для непосредственных целых операндов без знака. В тех командах, которые содержат целые без знака операнды, этот формат заполнен нулями до 32 разряда. Диапазон целого числа *Ni*:

$$0 \le Ni \le 2^{16} \qquad \text{(А.4.3)}$$



(a) Короткий целый формат без знака



(b) Заполнение нулями короткого целого формата без знака

Рисунок А.4.3 – Короткий целочисленный формат без знака (a) и с заполнением нулями (b)

А.4.2.2 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 А.4.4. Диапазон целого числа *Np*:

$$0 \le Np \le 2^{32} \qquad \text{(А.4.4)}$$

Рисунок А.4.4 – Целочисленный формат с одинарной точностью без знака

А.4.3 Форматы числа с плавающей запятой

Все форматы чисел с плавающей запятой на ПЦОС состоят из трёх полей: поля экспоненты (e), одного поля знакового разряда (s) и поля дробной части (f). Они представлены на рисунке А.4.5.

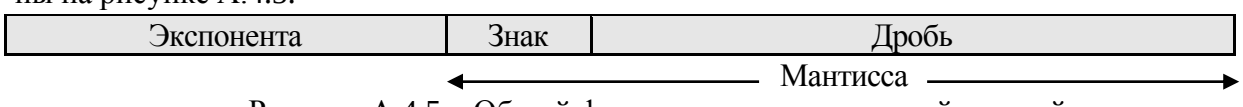


Рисунок А.4.5 – Общий формат числа с плавающей запятой

Поле экспоненты – это число в форме дополнения до двух. Поле знака и поле дробной части могут рассматриваться как единая часть и определяются как поле мантиссы (man). Дробная часть в форме дополнения до двух объединяется со знаковым разрядом, который включает в себя старший значащий разряд для создания мантиссы. Мантисса используется для представления нормализованного числа в дополнительном коде. В нормализованном представлении включается старший незнаковый разряд, обеспечивая, таким образом, дополнительный разряд точности.

Основная формула (А.4.5) для вычисления значения числа с плавающей точкой дана ниже.

Значение числа с плавающей запятой:

$$x = s\bar{s}.f_2 \times 2^e \ , \qquad \text{(А.4.5)}$$

где s – это значение одного бита, $\bar{s}$ – это инверсия значения одного бита, f – двоичное значение дробного поля и e – это десятичный эквивалент поля экспоненты.

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_2 – двоичное представление дробного поля;

- если s = 1, мантисса становиться равной $10.f_2$,

где f_2 – двоичное представление дробного поля.

Например, если f_2=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 переместится на одиннадцать разрядов вправо, полученное число 0100000000001$_2$, которое эквивалентно десятичному – 2049.

ПЦОС поддерживает три формата с плавающей запятой. Первый – короткий формат с плавающей запятой для непосредственных операндов с плавающей запятой, содержащих 4-разрядную экспоненту, 1 знаковый разряд и 11 разрядов для дробной части. Второй – это формат с одинарной точностью, содержащий 8-разрядную экспоненту, 1 знаковый разряд и 23-разряда для дробной части. Третий формат – это формат повышенной точности, содержащий 8-разрядную экспоненту, 1 знаковый разряд и 31 разряд для дробной части.

А.4.3.1 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.

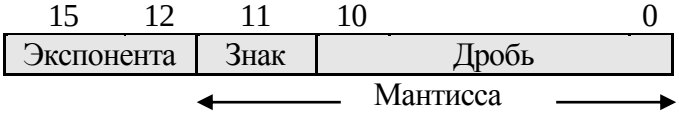


Рисунок А.4.6 – Короткий формат числа с плавающей запятой

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

$e = -8$

$s = 0$

$f = 0$

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:

$$x = 01.f_2 \times 2^e, \text{ если } s = 0 \quad \text{(A.4.6)}$$

$$x = 10.f_2 \times 2^e, \text{ если } s = 1 \quad \text{(A.4.7)}$$

$$x = 0, \text{ если } e = -8,\ s = 0,\ f = 0 \quad \text{(A.4.8)}$$

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: $x = -2 \times 2^7 = -2.5600 \times 10^2$.

А.4.3.2 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.

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

$$x = 01.f \times 2^e, \text{ если } s = 0 \quad \text{(A.4.9)}$$

$$x = 10.f \times 2^e, \text{ если } s = 1 \quad \text{(A.4.10)}$$

$$x = 0, \text{ если } e = -128 \quad \text{(A.4.11)}$$

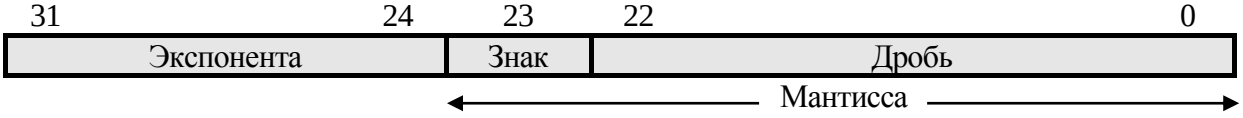


Рисунок А.4.7 – Формат числа с плавающей запятой с одинарной точностью

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

$e = -128$

$s = 0$

$f = 0$

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

- наибольшее положительное: $x = (2 - 2^{-23}) \times 2^{127} = 3.4028234 \times 10^{38}$;
- 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: $x = -2 \times 2^{127} = -3.4028236 \times 10^{38}$.

А.4.3.3 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.

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:

$$x = 01.f \times 2^e, \text{ если } s = 0 \quad \text{(А.4.12)}$$

$$x = 10.f \times 2^e, \text{ если } s = 1 \quad \text{(А.4.13)}$$

$$x = 0, \text{ если } e = -128,\ s = 0,\ f = 0 \quad \text{(А.4.14)}$$

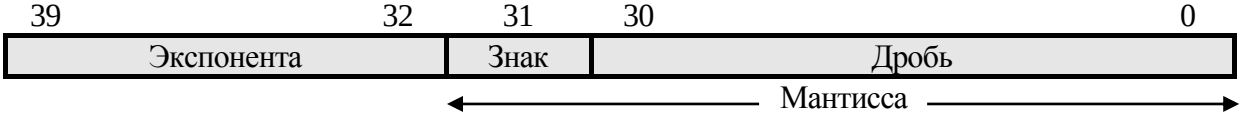


Рисунок А.4.8 – Формат числа с плавающей запятой с повышенной точностью

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

$e = -128$

$s = 0$

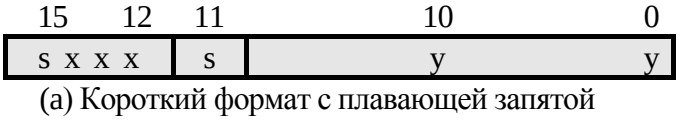
$f = 0$

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

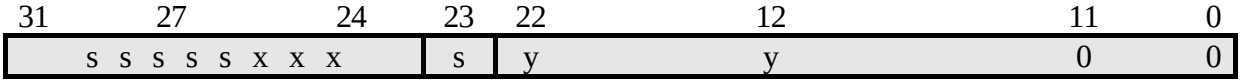
- наибольшее положительное: $x = (2 - 2^{-31}) \times 2^{127} = 3.4028236683 \times 10^{38}$;
- наименьшее положительное: $x = 1 \times 2^{-127} = 5.8774717541 \times 10^{-39}$;
- наибольшее отрицательное: $x = (-1 - 2^{-31}) \times 2^{-127} = -5.8774717569 \times 10^{-39}$;
- наименьшее отрицательное: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

А.4.3.4 Преобразования 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 А.4.9 – А.4.12 (*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.



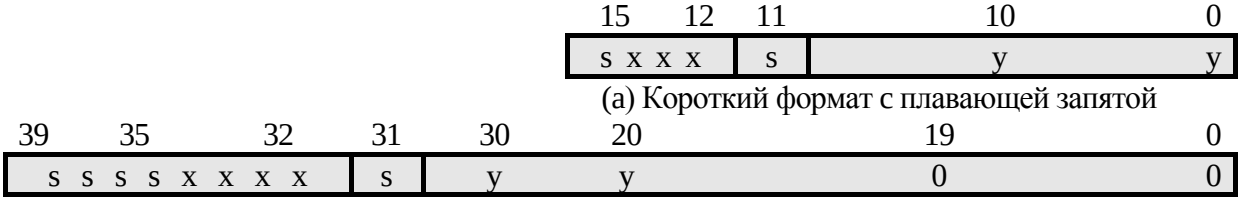
(a) Короткий формат с плавающей запятой



(b) Формат с плавающей запятой одинарной точности

Рисунок А.4.9 – Преобразование короткого формата с плавающей запятой в формат с плавающей запятой одинарной точности

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

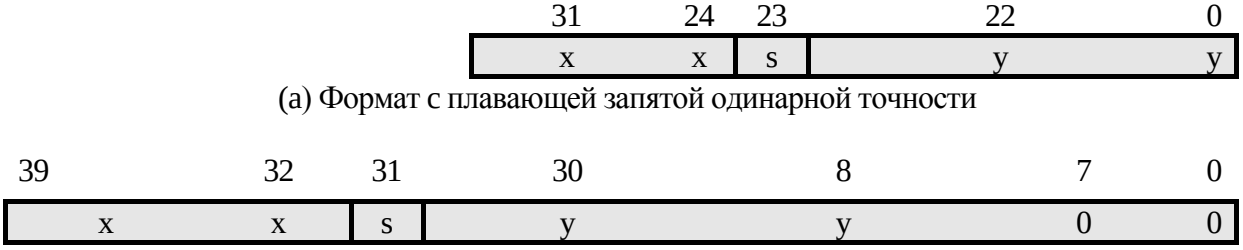


(a) Короткий формат с плавающей запятой

(b) Формат повышенной точности с плавающей запятой

Рисунок А.4.10 – Преобразование короткого формата с плавающей запятой в формат с плавающей запятой повышенной точности

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

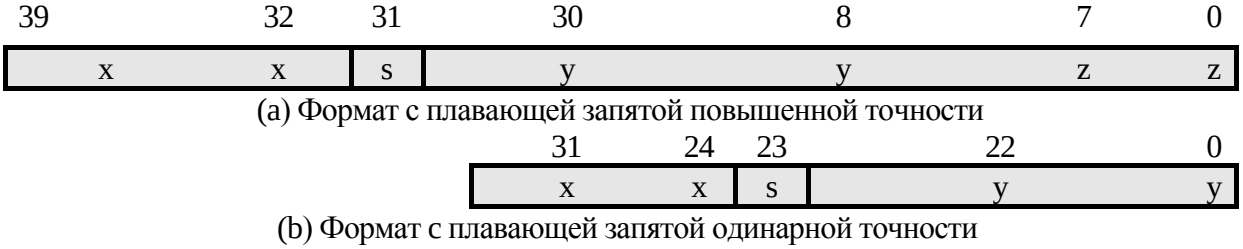


(a) Формат с плавающей запятой одинарной точности

(b) Формат повышенной точности с плавающей запятой

Рисунок А.4.11 – Преобразование формата с плавающей запятой одинарной точности в формат с плавающей запятой повышенной точности

Если выполняется преобразование из формата с одинарной точностью в формат с повышенной точностью, то крайние справа 8 разрядов дробного поля заполняются нулями.



(a) Формат с плавающей запятой повышенной точности

(b) Формат с плавающей запятой одинарной точности

Рисунок А.4.12 – Преобразование формата с плавающей запятой повышенной точности в формат с плавающей запятой одинарной точности

Если выполняется преобразование формата с повышенной точностью в формат с одинарной точностью, то крайние справа 8 разрядов дробного поля отбрасываются.

А.4.4 Умножение значений с плавающей запятой

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

$$a = a(man) \times 2^{a(exp)}, \qquad (A.4.15)$$

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

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

$$c = a \times b = a(man) \times b(man) \times 2^{(a(exp) + b(exp))} \qquad (A.4.16)$$

$$c(man) = a(man) \times b(man); \quad c(exp) = a(exp) + b(exp) \qquad (A.4.17)$$

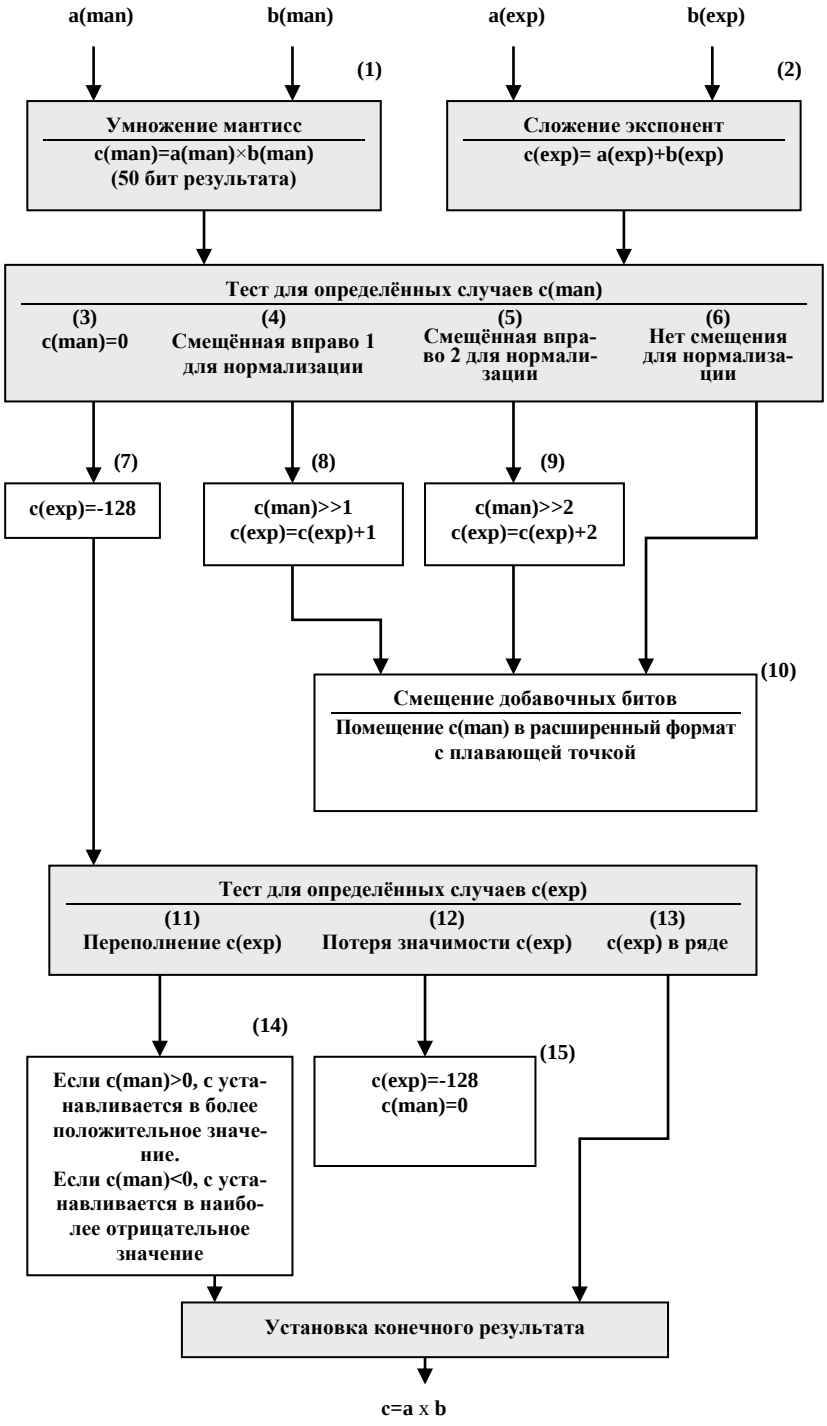


Рисунок А.4.13 – Блок-схема для умножения с плавающей точкой

При выполнении умножения с плавающей запятой, предполагается, что исходные операнды всегда будут в формате с плавающей запятой с одинарной точностью. Если исходные операнды в коротком формате, они расширяются до формата с плавающей запятой с одинарной точностью. Если исходные операнды в формате с плавающей запятой с повышенной точностью, они сокращаются до формата с одинарной точностью. Эти преобразования производятся автоматически аппаратно, без дополнительных затрат. Все результаты умножений с плавающей запятой предполагают формат с повышенной точностью. Эти умножения осуществляются в одном цикле.

На 1 этапе 24-разрядные мантиссы операндов источника умножаются, и на выходе получается 50-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) равна нулю, то на этапе 7 устанавливается c(exp) = -128, и обеспечивается, 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) переполнена в положительном направлении (этап 11), то c(exp) присваивается наибольшее положительное значение в формате с повышенной точностью. Если c(exp) переполнена 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 с присваивается значение ноль (этап 15), т. е. c(man) = 0 и c(exp) = -128.

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

Пример А.4.1 – Умножение с плавающей запятой (обе мантиссы равны -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 представлены в двоичном виде в соответствии с нормализованным форматом с плавающей запятой с одинарной точностью. Тогда:

$$10.00000000000000000000000 \times 2^{a(exp)}$$
$$\times\ 10.00000000000000000000000 \times 2^{b(exp)}$$
$$= 0100.00000000000000000000000000000000000000000000000 \times 2^{(a(exp)+b(exp))}$$

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

$$10.00000000000000000000000 \times 2^{a(exp)}$$
$$\times\ 10.00000000000000000000000 \times 2^{b(exp)}$$
$$= 01.00000000000000000000000000000000000000000000000 \times 2^{(a(exp)+b(exp)+2)}$$

При умножении с плавающей запятой экспонента результата может переполняться.

Это может произойти, когда экспонента изменяется при выполнении нормализации.

Пример А.4.2 – Умножение с плавающей запятой (обе мантиссы равны 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:

$$01.10000000000000000000000 \times 2^{a(exp)}$$
$$\times 01.10000000000000000000000 \times 2^{b(exp)}$$
$$= 0010.01000000000000000000000000000000000000000000000000 \times 2^{(a(exp)+b(exp))}$$

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

$$01.10000000000000000000000 \times 2^{a(exp)}$$
$$\times 01.10000000000000000000000 \times 2^{b(exp)}$$
$$= 01.00100000000000000000000000000000000000000000000000 \times 2^{(a(exp)+b(exp)+1)}$$

Пример А.4.3 – 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:

$$01.00000000000000000000000 \times 2^{a(exp)}$$
$$\times 01.00000000000000000000000 \times 2^{b(exp)}$$
$$= 0001.00000000000000000000000000000000000000000000000000 \times 2^{(a(exp)+b(exp))}$$

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 А.4.4 – Умножение с плавающей запятой между положительным и отрицательным числами

Дано:

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

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

Тогда:

$$01.00000000000000000000000 \times 2^{a(exp)}$$
$$\times 10.00000000000000000000000 \times 2^{b(exp)}$$
$$= 1110.00000000000000000000000000000000000000000000000000 \times 2^{(a(exp)+b(exp))}$$

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

Пример А.4.5 – Умножение с плавающей запятой на ноль

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

А.4.5 Сложение и вычитание с плавающей запятой

Блок-схема для сложения с плавающей точкой представлена на рисунке А.4.14. При сложении и вычитании с плавающей запятой два числа с плавающей запятой a и b должны быть определены как:

$$a = a(man) \times 2^{a(exp)} \quad (А.4.18)$$

$$b = b(man) \times 2^{b(exp)} \quad (А.4.19)$$

Сумма или разность a и b должна быть определена как:

$$c = a \pm b = \quad (А.4.20)$$

$$= (a(man) \pm (b(man) \times 2^{-(a(exp) - b(exp))})) \times 2^{a(exp)}, \quad \text{если } a(exp) \geq b(exp) \quad (А.4.21)$$

$$= ((a(man) \times 2^{-(b(exp) - a(exp))}) \pm b(man)) \times 2^{b(exp)}, \quad \text{если } a(exp) < b(exp) \quad (А.4.22)$$

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) ± 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 выравнивания мантисс, 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 наибольшее 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(exp) присваивается наименьшее отрицательное значение с повышенной точностью.

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

Пример А.4.6 – Сложение с плавающей запятой

Если будут складываться два нормализованных числа, дано:

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

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

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:

$01.10000000000000000000000000000000 \times 2^{0}$

$+ 00.10000000000000000000000000000000 \times 2^{0}$

$= 010.00000000000000000000000000000000 \times 2^{0}$

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

$01.10000000000000000000000000000000 \times 2^{0}$

$+ 00.10000000000000000000000000000000 \times 2^{0}$

$= 01.00000000000000000000000000000000 \times 2^{1}$

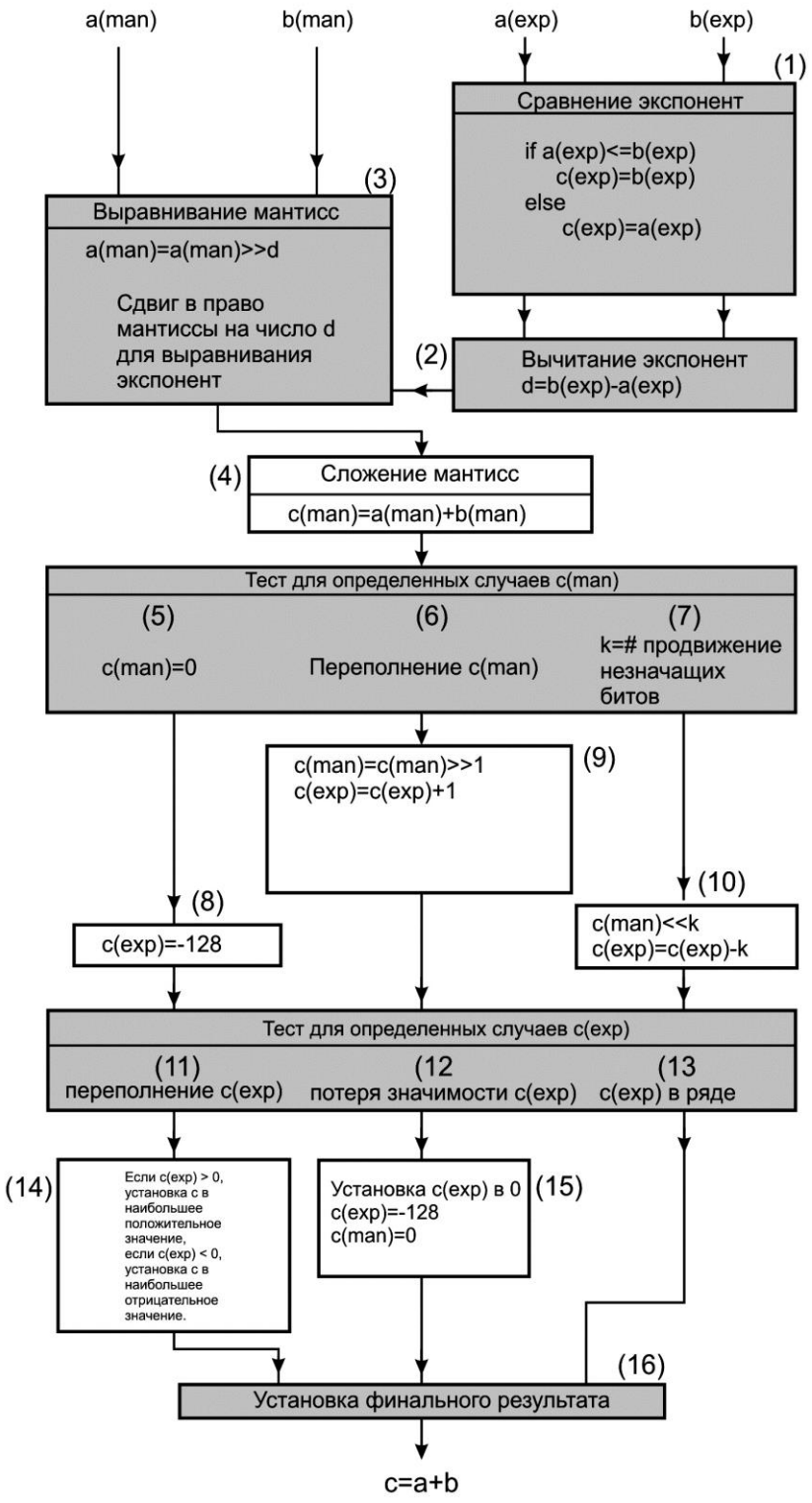


Рисунок А.4.14 – Блок-схема для сложения с плавающей точкой

Пример А.4.7 – Вычитание с плавающей запятой

Дано:

$a = 01.00000000000000000000000000000001 \times 2^0$

$b = 01.00000000000000000000000000000000 \times 2^0$

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

$01.00000000000000000000000000000001 \times 2^0$

$- 01.00000000000000000000000000000000 \times 2^0$

$= 00.00000000000000000000000000000001 \times 2^0$

Результат должен быть нормализован. В этом случае требуется сдвиг влево на 31.

Соответственно изменяется экспонента результата. В результате получится:

$01.00000000000000000000000000000001 \times 2^0$

$- 01.00000000000000000000000000000000 \times 2^0$

$= 01.00000000000000000000000000000000 \times 2^{-31}$

Пример А.4.8 – Сложение с плавающей запятой со сдвигом на 32 разряда

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

$a = 01.11111111111111111111111111111111 \times 2^{127}$

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

Должна быть выполнена операция a + b.

$01.11111111111111111111111111111111 \times 2^{127}$

$+ 10.00000000000000000000000000000000 \times 2^{127}$

$= 11.11111111111111111111111111111111 \times 2^{127}$

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

$01.11111111111111111111111111111111 \times 2^{127}$

$+ 10.00000000000000000000000000000000 \times 2^{127}$

$= 10.00000000000000000000000000000000 \times 2^{95}$

Пример А.4.9 – Сложение/вычитание с плавающей запятой и ноль

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

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

$0 \pm 0 = 0$

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

А.4.6 Использование команды NORM для нормализации

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

- 1 этап – размещает старший незнаковый разряд числа с плавающей запятой;
- 2 этап – сдвигает влево число для нормализации;
- 3 этап – устанавливает экспоненту.

Данное значение а с плавающей запятой с повышенной точностью не нормализовано, нормализация показана на рисунке А.4.15.

Пример А.4.10 – Команда NORM. Форматы данных и операции с плавающей запятой

Предположим, что регистр повышенной точности содержит значения:

man = 00000000000000000001000000000001, exp = 0

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

man = 0.00000000000000000001000000000001, exp = 0

Знак этого числа расширен 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

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

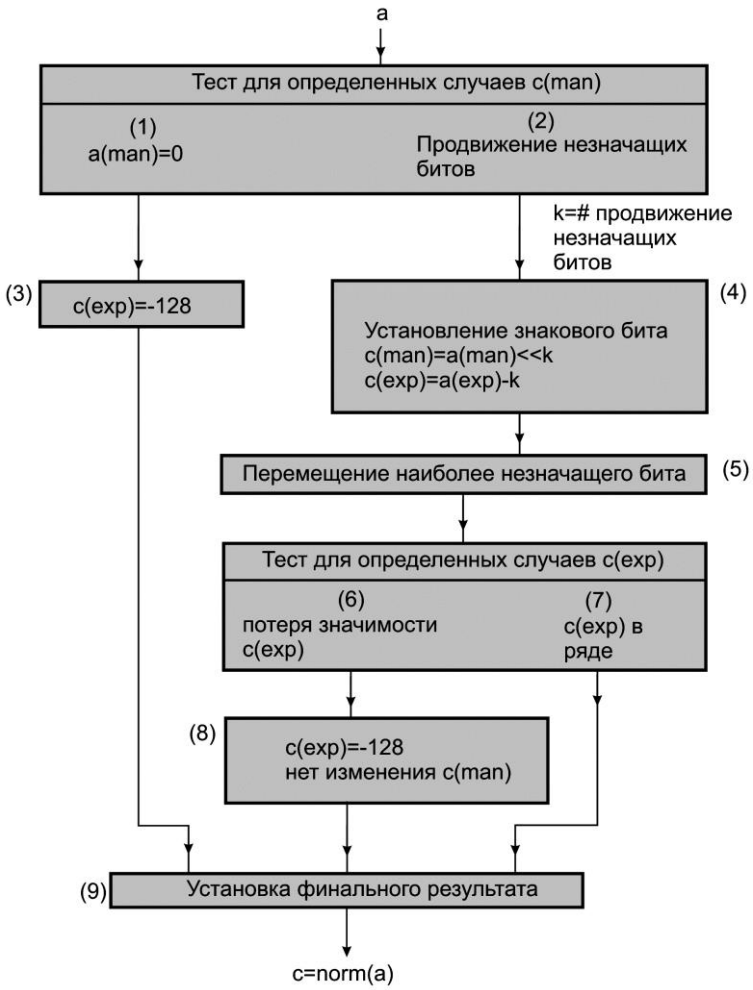


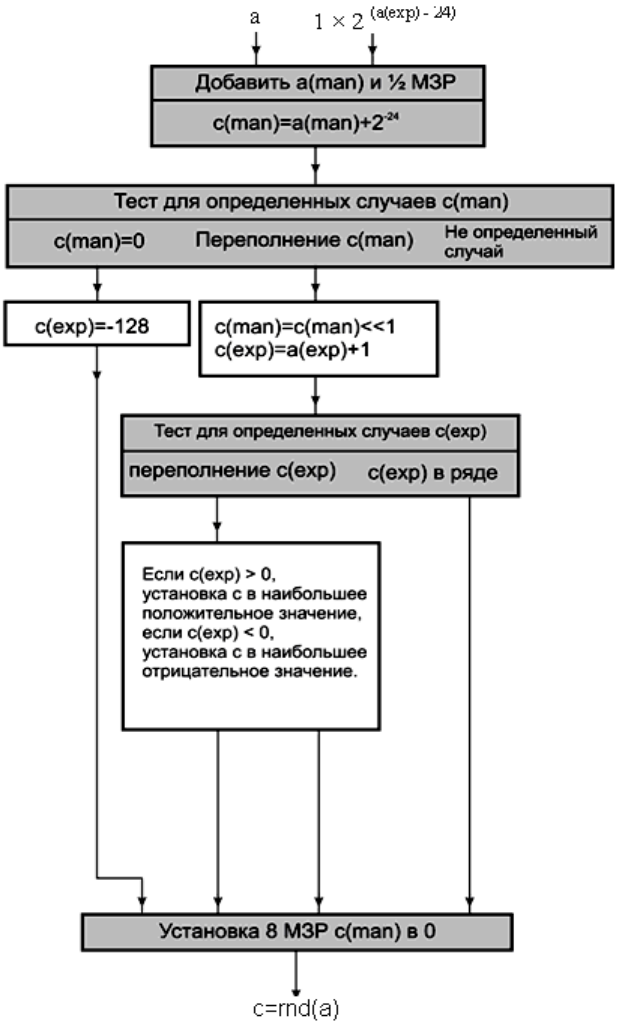
Рисунок А.4.15 – Блок-схема выполнения команды NORM

А.4.7 Округление: команда RND

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

$$c = a(man) \times 2^{a(exp)} + (1 \times 2^{(a(exp)-24)}) \qquad (A.4.23)$$

Затем выполняется преобразование из формата с плавающей запятой с повышенной точностью в формат с плавающей запятой с одинарной точностью.



МЗР – младший значащий разряд

Рисунок А.4.16 – Блок-схема для округления числа с плавающей запятой с помощью команды RND

А.4.8 Преобразование значения с плавающей запятой в целое

Блок-схема для преобразования значения с плавающей запятой в целое показана на рисунке А.4.17.

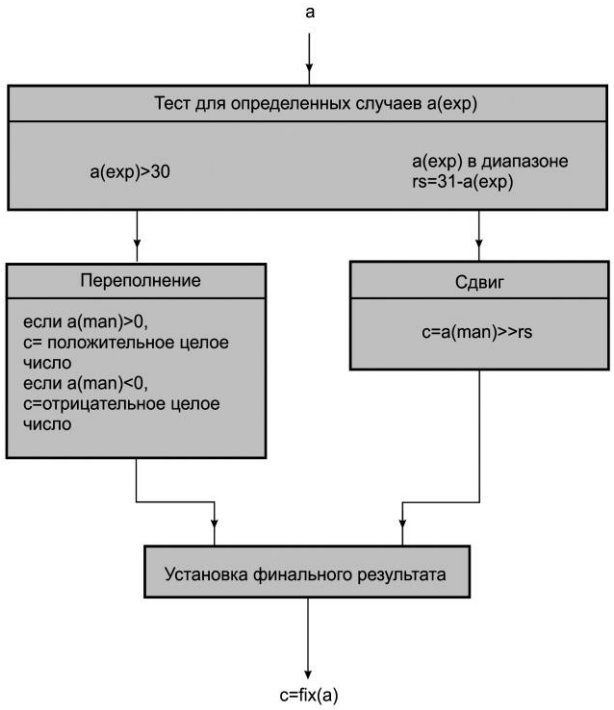


Рисунок А.4.17 – Блок-схема преобразования значения с плавающей запятой в целое с помощью команды FIX

Преобразование значения с плавающей запятой в целое, используя команду 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 *a* в целое будет обозначаться *fix(a).* Преобразование не приведёт к переполнению, если преобразуемое число *a* находится в диапазоне:

$$-2^{31} \le a \le 2^{31} - 1 \qquad \text{(А.4.24)}$$

Сначала необходимо определить, что:

$$a(\text{exp}) \le 30 \qquad \text{(А.4.25)}$$

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:

$$\text{rs} = 31 - \text{a(exp)} \qquad \text{(А.4.26)}$$

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

если $0 \le a < 1$, тогда fix(a) = 0;

если $-1 \le a < 0$, тогда fix(a) = -1.

A.4.9 Преобразование целого в значение с плавающей запятой командой FLOAT

Преобразование целого значения в значение с плавающей запятой, используя команду FLOAT, позволяет осуществить преобразование целого с одинарной точностью в число в формате с плавающей запятой повышенной точности.

Блок-схема этого преобразования дана на рисунке А.4.18.

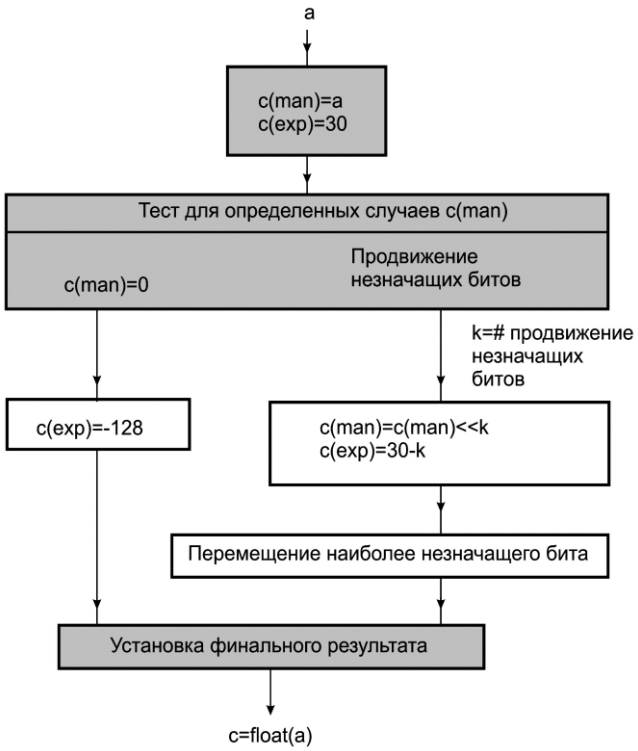


Рисунок А.4.18 – Блок-схема преобразования целого числа в число с плавающей запятой с помощью команды FLOAT

A.5 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 рассматриваются операции, кодирование и применение режимов адресации, описано управление системными стеком, очередями и исключениями из очереди в памяти.

В данном разделе рассмотрены следующие основные темы:

- типы адресации (подраздел А.5.1):
  - регистровый;
  - прямой;
  - косвенный;
  - короткий непосредственный;
  - длинный непосредственный;
  - относительный (относительно PC);
- группы режимов адресации (подраздел А.5.2):
  - основные режимы адресации;

- трёхоперандные режимы адресации;
- параллельные режимы адресации;
- режим длинной непосредственной адресации;
- режимы адресации условных переходов;
- циклическая адресация (подраздел А.5.3);
- битреверсная адресация (подраздел А.5.4);
- управление стеком системы (подраздел А.5.5).

А.5.1 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 командах и отсутствуют в других.

По этой причине типы адресации используются в пяти различных группах режимов адресации:

1) основные режимы адресации (G):
   - регистровый;
   - прямой;
   - косвенный;
   - короткий непосредственный;
2) трёхоперандные режимы адресации (T):
   - регистровый;
   - косвенный;
3) режимы параллельной адресации (P):
   - регистровый;
   - косвенный;
4) режим длинной непосредственной адресации:
   - длинной непосредственной;
5) режимы адресации условных переходов (B):
   - регистровый;
   - относительный.

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

А.5.1.1 Регистровая адресация

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

Состав регистрового файла ЦПУ приведён в таблице А.5.1.

Таблица А.5.1 – Состав регистрового файла ЦПУ

Название регистра	Машинный адрес (hex)*	Функциональное назначение
R0	00h	Регистр повышенной точности 0
R1	01h	Регистр повышенной точности 1
R2	02h	Регистр повышенной точности 2
R3	03h	Регистр повышенной точности 3
R4	04h	Регистр повышенной точности 4
R5	05h	Регистр повышенной точности 5
R6	06h	Регистр повышенной точности 6
R7	07h	Регистр повышенной точности 7
AR0	08h	Вспомогательный регистр 0
AR1	09h	Вспомогательный регистр 1
AR2	0Ah	Вспомогательный регистр 2
AR3	0Bh	Вспомогательный регистр 3
AR4	0Ch	Вспомогательный регистр 4
AR5	0Dh	Вспомогательный регистр 5
AR6	0Eh	Вспомогательный регистр 6
AR7	0Fh	Вспомогательный регистр 7
DP	10h	Указатель страницы данных
IR0	11h	Индексный регистр 0
IR1	12h	Индексный регистр 1
BK	13h	Регистр размера блока
SP	14h	Указатель системного стека
ST	15h	Регистр состояния
IE	16h	Регистр разрешения прерываний ЦПУ/ПДП
IF	17h	Регистр флагов прерываний ЦПУ
IOF	18h	Флаги ввода-вывода
RS	19h	Регистр начального адреса повторений
RE	1Ah	Регистр конечного адреса повторений
RC	1Bh	Счётчик повторений

* Машинный адрес – получаемое при ассемблировании шестнадцатеричное значение, кодирующее идентификатор регистра в соответствующем поле объектного кода команд работы с регистрами (не является физическим адресом памяти ЦПУ).

A.5.1.2 Прямая адресация

В прямой адресации адрес данных формируется путём объединения восьми младших разрядов указателя страницы данных (DP) с 16 младшими разрядами командного слова (expr). В результате программист имеет возможность обращаться к большому адресному пространству (256 страниц по 64K слов на каждой) и нет необходимости изменять указатель страницы. На рисунке А.5.1 приведено формирование адреса данных.

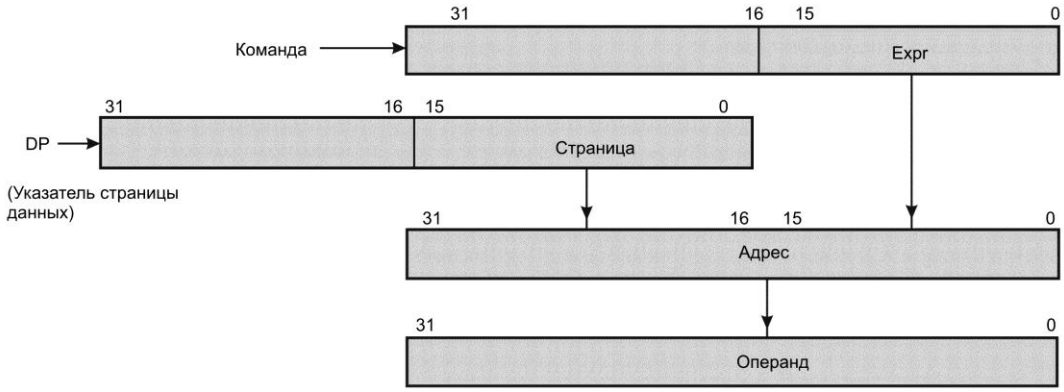


Рисунок А.5.1 – Прямая адресация

Синтаксис и операции при прямой адресации приведены ниже.
Синтаксис: @expr
Операция: адрес = DP объединён с expr

В примере А.5.1 приводится пример команды с данными до и после выполнения команды.

Пример А.5.1 – Прямая адресация

ADDI @0BCDEh, R7

Перед выполнением команды:	После выполнения команды:
DP = 8Ah	DP = 8Ah
R7 = 0h	R7 = 12345678h
Данные в 8ABCDEh = 12345678h	Данные в 8ABCDEh = 12345678h

A.5.1.3 Косвенная адресация

К косвенной адресации прибегают для определения адреса операнда в памяти, используя вспомогательный регистр, а также дополнительное смещение и индексные регистры. В косвенной адресации используются только 24 младших разряда вспомогательных и индексных регистров. Арифметические операции над содержимым регистров (24 младшими разрядами без знака) выполняются арифметическими устройствами вспомогательных регистров (ARAUn). Старшие восемь разрядов не изменяются.

Гибкость косвенной адресации обеспечивается одновременной с операциями ЦПУ модификацией вспомогательных регистров посредством ARAUn. Косвенная адресация определяется в командном слове пятиразрядным полем mod. Смещение – это либо восьмиразрядное целое без знака, содержащееся в командном слове или неявное смещение на один. Два индексных регистра IR0 и IR1 также могут быть использованы в косвенной адресации. В некоторых случаях дополнительно можно использовать циклическую или битреверсную адресацию. Механизм формирования адресов в циклической адресации рассмотрен в подразделе А.5.3, битреверсной – в подразделе А.5.4.

В таблице А.5.2 представлены разные виды косвенной адресации с соответствующим каждому виду полем модификации (mod), синтаксисом ассемблера, операцией и функцией.

Далее приведены 18 примеров, показывающих операцию для каждого вида косвенной адресации.

Пример А.5.2 – Косвенная адресация через вспомогательный регистр

Адрес операнда, который будет выбран, содержится во вспомогательном регистре (ARn).

Операция: адрес операнда = ARn

Синтаксис: * ARn

Поле модификации: 11000

Рисунок А.5.2 – Косвенная адресация через вспомогательный регистр

Пример А.5.3 – Косвенная адресация с предварительным добавлением смещения

Выбираемый адрес операнда – это сумма содержимого вспомогательного регистра (ARn) и смещения (disp). Смещение – это либо восьмиразрядное целое без знака, содержащееся в командном слове, либо неявное значение единицы.

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

Синтаксис ассемблера: * + ARn (disp)

Поле модификации: 00000

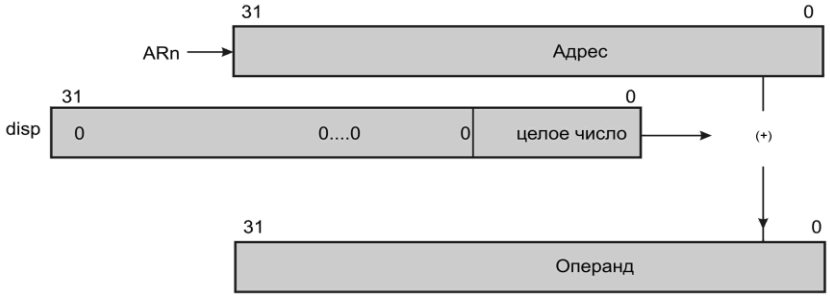


Рисунок А.5.3 – Косвенная адресация с предварительным добавлением смещения

Пример А.5.4 – Косвенная адресация с предварительным вычитанием смещения

Выбираемый адрес операнда соответствует содержимому вспомогательного регистра минус смещение (disp). Смещение – это или восьмиразрядное целое, содержащееся в командном слове, или неявное значение единицы.

Операция: адрес операнда = ARn – disp

Синтаксис ассемблера: * - ARn (disp)

Поле модификации: 00001

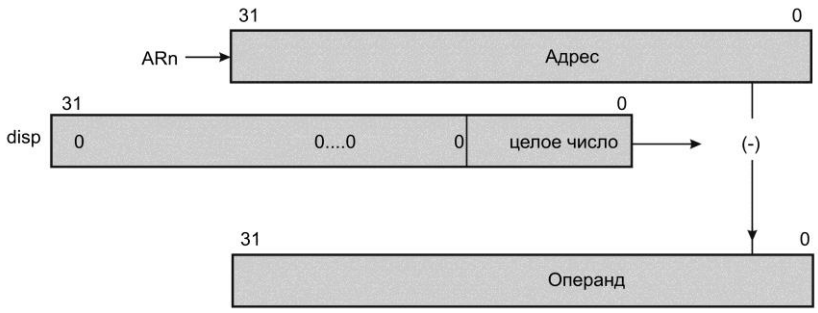


Рисунок А.5.4 – Косвенная адресация с предварительным вычитанием смещения

Пример А.5.5 – 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 инструкционном слове, или подразумевается значение 1. После этого данные выбираются, вспомогательный регистр обновляется со сгенерированным адресом.

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

ARn = ARn + disp

Синтаксис ассемблера: *++ARn (disp)

Модификационное поле: 00010

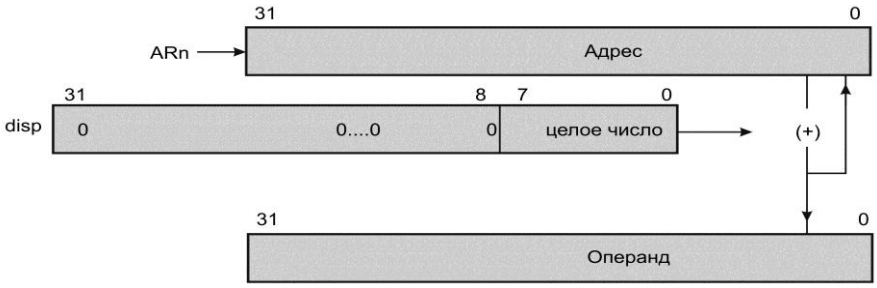


Рисунок А.5.5 – Косвенная адресация с предварительным добавлением смещения и модификацией

Пример А.5.6 – Косвенная адресация с предварительным вычитанием смещения и модификацией

Выбираемый адрес операнда – это разность между содержимым вспомогательного регистра (ARn) и смещения (disp). Смещение – это либо восьмиразрядное целое, содержащееся в командном слове, либо неявное значение единицы. После выборки данных вспомогательный регистр загружается сгенерированным адресом.

Операция: адрес операнда = ARn – disp,

ARn = ARn + disp

Синтаксис ассемблера: * – – ARn (disp)

Поле модификации: 00011

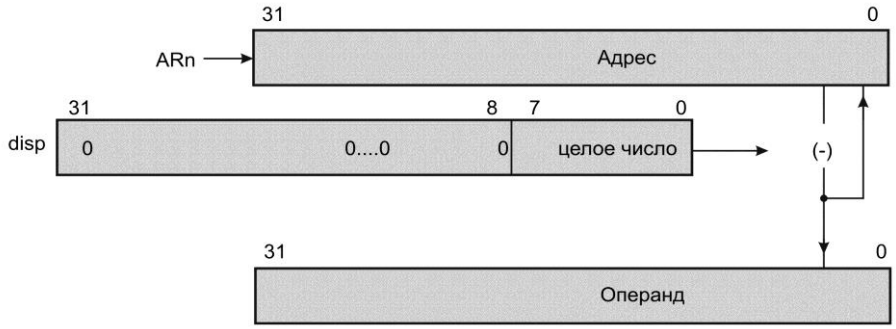


Рисунок А.5.6 – Косвенная адресация с предварительным вычитанием смещения и модификацией

Пример А.5.7 – Косвенная адресация с последующим добавлением смещения и модификацией

Выбранный адрес операнда соответствует содержимому вспомогательного регистра (ARn). После выбора операнда смещение (disp) добавляется к содержимому вспомогательного регистра. Смещение – это либо восьмиразрядное целое, содержащееся в командном слове, либо неявное значение единицы.

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

Синтаксис ассемблера: *ARn ++ (disp)

Поле модификации: 00100

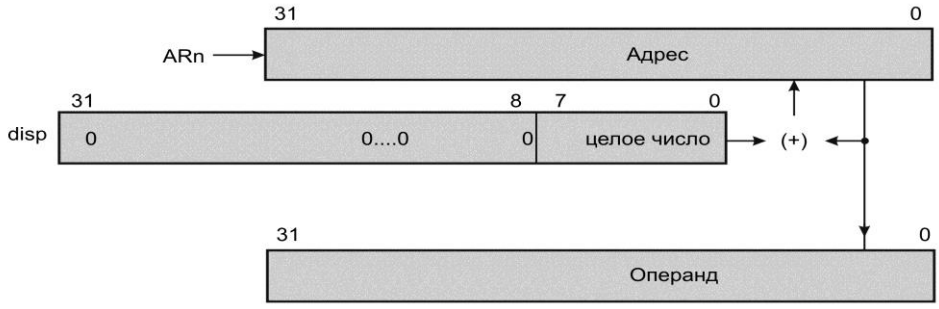


Рисунок А.5.7 – Косвенная адресация с последующим добавлением смещения и модификацией

Пример А.5.8 – Косвенная адресация с последующим вычитанием смещения и модификацией

Выбираемый адрес операнда соответствует содержимому вспомогательного регистра (ARn). После выборки операнда смещение (disp) – вычитается из содержимого вспомогательного регистра. Смещение – это либо восьмиразрядное целое, содержащееся в командном слове, либо неявное значение единицы.

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

Синтаксис ассемблера: * ARn – – (disp)

Поле модификации: 00101

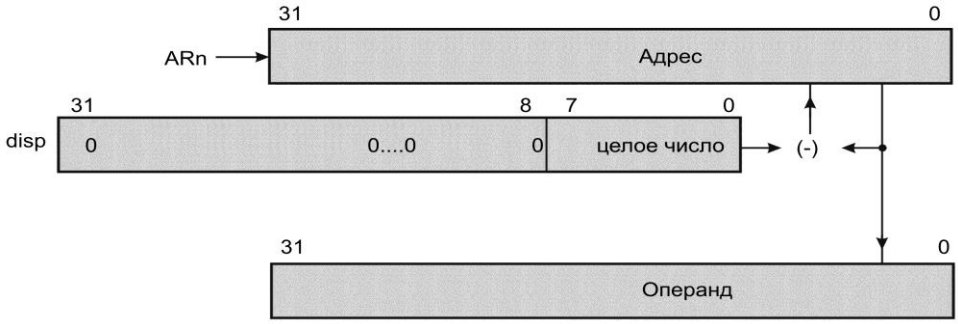


Рисунок А.5.8 – Косвенная адресация с последующим вычитанием смещения и модификацией

Пример А.5.9 – Косвенная адресация с последующим сложением смещения и циклической модификацией

Выбираемый операнд соответствует содержимому вспомогательного регистра (ARn). После выбора операнда смещение добавляется к содержимому вспомогательного регистра, использующего циклическую адресацию. Результат используется для изменения вспомогательного регистра. Смещение – это либо восьмиразрядное целое, содержащееся в командном слове, либо неявное значение единицы.

Операция: адрес операнда = ARn,

ARn = circ(ARn + disp)

Синтаксис ассемблера: * ARn ++ (disp) %

Поле модификации: 00110

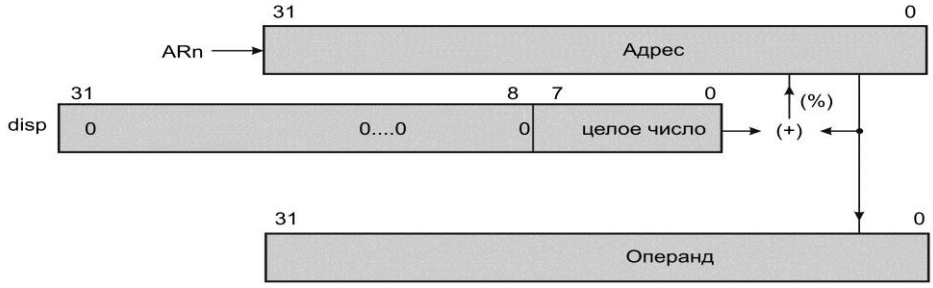


Рисунок А.5.9 – Косвенная адресация с последующим сложением смещения и циклической модификацией

Пример А.5.10 – Косвенная адресация с последующим вычитанием смещения и циклической модификацией

Выбираемый адрес операнда соответствует содержимому вспомогательного регистра (ARn). После выбора операнда смещение (disp) вычитается из содержимого вспомогательного регистра, используя циклическую адресацию. Результат используется для изменения вспомогательного регистра. Смещение – это либо восьмиразрядное целое, содержащееся в командном слове, либо неявное значение единицы.

Операция: адрес операнда = ARn,

ARn = circ(ARn - disp)

Синтаксис ассемблера: * ARn – – (disp) %

Поле модификации: 00111

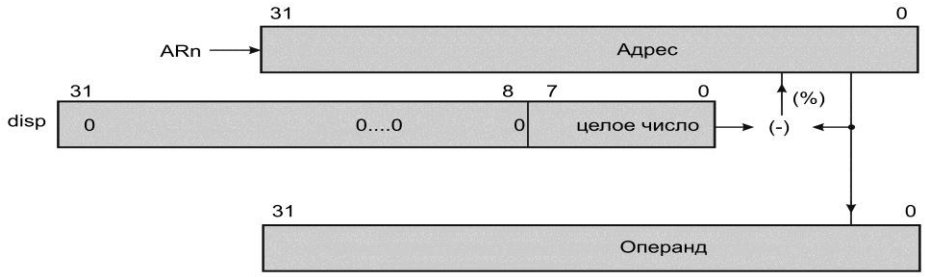


Рисунок А.5.10 – Косвенная адресация с последующим вычитанием смещения и циклической модификацией

Пример А.5.11 – Косвенная адресация с предварительным добавлением индекса

Выбираемый адрес операнда – это сумма содержимого вспомогательного регистра (ARn) и индексного регистра (IR0 или IR1).

Операция: адрес операнда = ARn + IRm

Синтаксис ассемблера: * + ARn (IRm)

Поле модификации: 01000, если m=0,
10000, если m=1.

Рисунок А.5.11 – Косвенная адресация с предварительным добавлением индекса

Пример А.5.12 – Косвенная адресация с предварительным вычитанием индекса

Выбираемый адрес операнда – это разность между содержимым вспомогательного регистра (ARn) и индексного регистра (IR0 или IR1).

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

Синтаксис ассемблера: * - ARn (IRm)

Поле модификации: 01001, если m=0,
10001, если m=1.

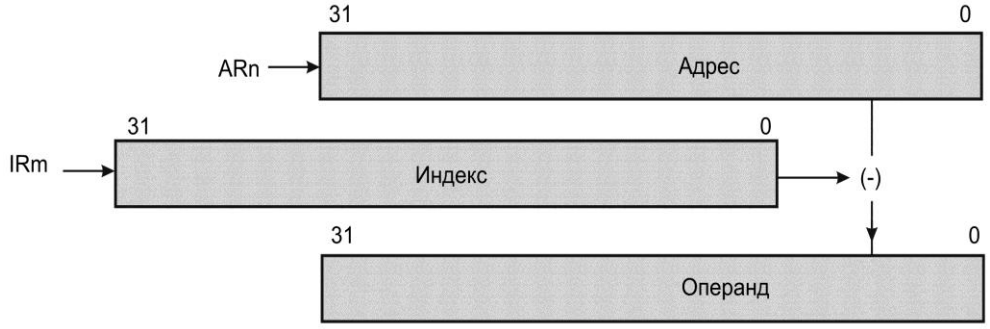


Рисунок А.5.12 – Косвенная адресация с предварительным вычитанием индекса

Пример А.5.13 – Косвенная адресация с предварительным сложением индекса и модификацией

Выбираемый адрес операнда – это сумма содержимого вспомогательного регистра (ARn) и индексного регистра (IR0 или IR1). После выборки данных содержимое вспомогательного регистра заменяется сгенерированным адресом.

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

Синтаксис ассемблера**:** * ++ ARn (IRm)

Поле модификации: 01010, если m=0,
10010, если m=1.

Рисунок А.5.13 – Косвенная адресация с предварительным сложением индекса и модификацией

Пример А.5.14 – Косвенная адресация с предварительным вычитанием индекса и модификацией

Выбираемый адрес операнда – это разность между содержимым вспомогательного регистра (ARn) и индексного регистра (IRm). Результирующий адрес становится новым содержимым вспомогательного регистра.

Операция**:** адрес операнда = ARn - IRm,
ARn = ARn - IRm

Синтаксис ассемблера: * – – ARn (IRm)

Поле модификации: 01011, если m=0,
10011, если m=1.

Рисунок А.5.14 – Косвенная адресация с предварительным вычитанием индекса и модификацией

Пример А.5.15 – Косвенная адресация с последующим прибавлением индекса и модификацией

Выбираемый адрес операнда соответствует содержимому вспомогательного регистра (ARn). После выборки операнда содержимое индексного регистра (IR0 или IR1) прибавляется к содержимому вспомогательного регистра.

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

Синтаксис ассемблера**:** * ARn ++ (IRm)

Поле модификации: 01100, если m=0,
10100, если m=1.

Рисунок А.5.15 – Косвенная адресация с последующим прибавлением индекса и модификацией

Пример А.5.16 – Косвенная адресация с последующим вычитанием индекса и модификацией

Выбираемый адрес операнда соответствует содержимому вспомогательного регистра (ARn). После выборки операнда содержимое индексного регистра (IR0 или IR1) вычитается из вспомогательного регистра.

Операция: адрес операнда = ARn,
ARn = ARn - IRm

Синтаксис ассемблера: * ARn – – (IRm)

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

Рисунок А.5.16 – Косвенная адресация с последующим вычитанием индекса и модификацией

Пример А.5.17 – Косвенная адресация с последующим сложением с индексом и циклической 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 круговую адресацию и перезаписывается содержимым вспомогательного регистра.

Операция**:** адрес операнда = ARn,

ARn = circ(ARn + IRm)

Синтаксис ассемблера: *ARn ++ (IRm) %

Поле модификации: 01110, если m=0,

10110, если m=1.

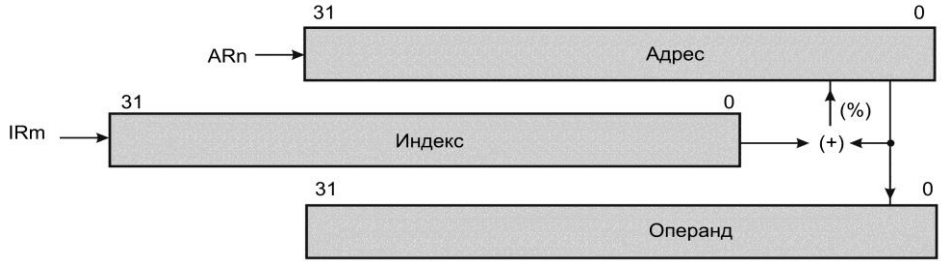


Рисунок А.5.17 – Косвенная адресация с последующим сложением с индексом и циклической модификацией

Пример А.5.18 – Косвенная адресация с последующим вычитанием индекса и циклической модификацией

Выбираемый адрес операнда соответствует содержимому вспомогательного регистра (ARn). После выборки операнда содержимое индексного регистра (IR0 или IR1) вычитается из содержимого вспомогательного регистра. Это значение вычисляется, используя циклическую адресацию, и замещает содержимое вспомогательного регистра.

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

Синтаксис ассемблера: * ARn – – (IRm)%

Поле модификации: 01111 , если m=0,

10111 , если m=1.

Рисунок А.5.18 – Косвенная адресация с последующим вычитанием индекса и циклической модификацией

Пример А.5.19 – Косвенная адресация с последующим прибавлением индекса и битреверсной модификацией

Выбираемый адрес операнда соответствует содержимому вспомогательного регистра (ARn). После выборки операнда содержимое индексного регистра (IR0) складывается с содержимым вспомогательного регистра. Сложение выполняется с обратным распространением переноса, который может быть использован для выдачи битреверсного адреса (B). Это значение замещает содержимое вспомогательного регистра.

Операция: адрес операнда = ARn,
ARn = B(ARn + IR0)

Синтаксис ассемблера: * ARn ++ (IR0)B

Поле модификации: 11001

Рисунок А.5.19 – Косвенная адресация с последующим прибавлением индекса и битреверсной модификацией

А.5.1.4 Короткая непосредственная адресация

В короткой непосредственной адресации операнд – это 16-разрядная непосредственная величина, содержащаяся в 16 младших значащих разрядах командного слова (expr). В зависимости от типа данных, требуемых для команды, короткий непосредственный операнд может быть целым в дополнительном коде, целым без знака или числом с плавающей запятой.

Синтаксис для этого режима приведён ниже.

Синтаксис: expr

В примере А.5.20 приведён пример команды с данными до и после команды.

Пример А.5.20 – Короткая непосредственная адресация

SUBI 1, R0

До команды:	После команды:
R0 = 0h	R0 = 0FFFFFFFFh

А.5.1.5 Длинная непосредственная адресация

В длинной непосредственной адресации операнд – это 24-разрядная непосредственная величина, содержащаяся в 24 младших разрядах командного слова (expr). Синтаксис для этого режима приведён ниже.

Синтаксис: expr

В примере А.5.21 дан пример команды с данными до и после выполнения команды.

Пример А.5.21 – Длинная непосредственная адресация

BR 8000h

До команды:	После команды:
PC = 0h	PC = 8000h

A.5.1.6 Относительная адресация (относительно PC)

Относительная адресация используется для ветвления. Ассемблер использует src (метку или адрес), определённый пользователем и генерирует смещение.

Если переход стандартный, то смещение равно: метка − (PC + 1).

Если переход задержанный, то смещение равно: метка − (PC + 3). Смещение хранится как 16-разрядное целое со знаком в младших разрядах командного слова. Смещение добавляется к значению PC, если условие выполнено.

Синтаксис: expr

В примере A.5.22 представлена команда с данными до и после выполнения команды.

Пример A.5.22 – Адресация относительно PC

BU NEWPC; PC=1001h, NEWPC=1005h, смещение=3

До команды:	После команды:
PC=1001h	PC=1005h

A.5.2 Группы режимов адресации

Шесть типов адресации, приведённые в разделе A.5.1, используются для формирования следующих пяти групп режимов адресации:

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

A.5.2.1 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 dst – операнд назначения, src – операнд источника, «операция» – операция, которая будет выполнена, используя основные режимы адресации для определения указанных операндов. Разряды 31-29 – нули, определяющие команды основного режима адресации. Разряды 22 и 21 определяют поле основного режима адресации (G), которое определяет назначение разрядов с 15 по 0 для адресации операнда src.

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

0 0 - регистровая (все регистры ЦПУ, если не определены иначе);

0 1 - прямая;

1 0 - косвенная;

1 1 - непосредственная.

Если поля src и dst соответствуют спецификациям регистра, то значения в этих полях соответствуют адресам регистра ЦПУ как определено таблицей A.5.2. Для режимов основной адресации допустимы следующие значения ARn: (ARn, $0 \le n \le 7$).

31 29	28 23	22	21	20 16	15 14 13 12 11 10 9 8 7 6 5	4 0
0 0 0	операция	0	0	dst	0 0 0 0 0 0 0 0 0 0 0	src
0 0 0	операция	0	1	dst	прямая	
0 0 0	операция	1	0	dst	modn ARn	disp
0 0 0	операция	1	1	dst	непосредственная	
		G	G	Назначение	Исходные операнды	

Рисунок A.5.20 – Основные режимы адресации

Таблица А.5.2 – Кодирование для основных режимов адресации

Поле модификации	Синтаксис	Операция	Описание
Косвенная адресация с заменой			
00000	*+ ARn(disp)	addr = ARn + disp	Сложение с предварительным смещением
00001	*- ARn(disp)	addr = ARn - disp	Вычитание с предварительным смещением
00010	* ++ ARn(disp)	addr = ARn + disp ARn = ARn + disp	Сложение и модификация с предварительным смещением
00011	* – – ARn(disp)	addr = ARn - disp ARn = ARn - disp	Вычитание и модификация с предварительным смещением
00100	* ARn ++(disp)	addr = ARn ARn = ARn + disp	Сложение и модификация с последующим смещением
00101	* ARn – – (disp)	addr = ARn ARn = ARn - disp	Вычитание и модификация с последующим смещением
00110	* ARn ++ disp)%	addr = ARn ARn= circ(ARn+disp)	Сложение и циклическая модификация с последующим смещением
00111	* ARn – – (disp)%	addr = ARn ARn =circ(ARn-disp)	Вычитание и циклическая модификация с последующим смещением
Косвенная адресация с индексным регистром (IR0)			
01000	* + ARn(IR0)	addr = ARn + IR0	Сложение с предварительным индексированием (IR0)
01001	* - ARn(IR0)	addr = ARn + IR0	Вычитание с предварительным индексированием (IR0)
01010	* ++ ARn(IR0)	addr = ARn + IR0 ARn =ARn + IR0	Сложение и модификация с предварительным индексированием (IR0)
01011	*– – ARn(IR0)	addr = Arn - IR0 Arn = Arn - IR0	Вычитание и модификация с предварительным индексированием (IR0)
01100	* ARn ++ (IR0)	addr = ARn ARn = ARn + IR0	Сложение и модификация с последующим индексированием (IR0)
01101	* ARn – – (IR0)	addr = ARn ARn = ARn - IR0	Вычитание и модификация с последующим индексированием (IR0)
01110	* ARn ++ (IR0)%	addr = ARn ARn = irc(ARn+IR0)	Сложение и циклическая модификация с последующим индексированием
01111	* ARn – – (IR0)%	addr = ARn ARn = circ(ARn-IR0)	Вычитание и циклическая модификация с последующим индексированием
Косвенная адресация с индексным регистром (IR1)			
10000	* + ARn(IR1)	addr = ARn + IR1	Сложение с предварительным индексированием (IR1)
10001	* - ARn(IR1)	addr = ARn - IR1	Вычитание с предварительным индексированием (IR1)
10010	* ++ ARn(IR1)	addr = ARn + IR1 ARn = ARn + IR1	Вычитание и модификация с предварительным индексированием (IR1)
10011	* – – ARn(IR1)	addr = ARn - IR1 ARn = ARn - IR1	Вычитание и модификация с предварительным индексированием (IR1)
10100	* ARn ++ (IR1)	addr = ARn ARn = ARn + IR1	Сложение и модификация с последующим индексированием (IR1)
10101	* ARn – – (IR1)	addr = ARn ARn = ARn - IR1	Вычитание и модификация с последующим индексированием (IR1)
10110	* ARn ++ (IR1)%	addr = ARn ARn = irc(ARn+IR1)	Сложение и циклическая модификация с последующим индексированием (IR1)
10111	* ARn – – (IR1)%	addr = ARn ARn = circ(ARn-IR1)	Вычитание и циклическая модификация с последующим индексированием (IR1)
Косвенная адресация (специальные случаи)			
11000	* ARn	addr = ARn	Косвенная
11001	* ARn ++ (IR0)B	addr = ARn ARn = B(ARn + IR0)	Сложение и битреверсная модификация с последующим индексированием (IR0)

A.5.2.2 Трёхоперандные режимы адресации

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

SRC1 «операция» SRC2 → dst,

где dst – операнд назначения, SRC1 и SRC2 – операнды источника, «операция» определяет выполняемую операцию.

Примечание – Цифра «3» может быть опущена в трёхоперандных командах.

В разрядах 31-29 установлено значение 001, обозначающее команды трёхоперандного режима адресации. Разряды 22 и 21 определяют поле (T) трёхоперандного режима адресации, которое определяет, как интерпретируются разряды 15-0 для адресации операндов SRC. Разряды 15-8 используются для определения адреса SRC1 и разряды 7-0 для определения адреса SRC2.

Варианты для разрядов 22 и 21 следующие:

T (22 21)	SRC1	SRC2
0 0	Регистровый	Регистровый
0 1	Косвенный	Косвенный
1 0	Регистровый	Регистровый
1 1	Косвенный	Косвенный

Если поля SRC1 и SRC2 используют одинаковые вспомогательные регистры, то оба адреса сгенерированы правильно. Однако, только значение, созданное полем SRC1, сохраняется в определённом вспомогательном регистре. Ассемблер выдаёт предупреждение, если это условие определено пользователем.

Допустимы следующие значения ARn и ARm:

ARn, $0 \le n \le 7$

ARm, $0 \le m \le 7$

Сокращения «modm» или «modn» обозначают поле модификации, соответствующее ARn или ARm. В таблице А.5.2 приведена полная информация.

В косвенной адресации трёхоперандного режима адресации, 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.

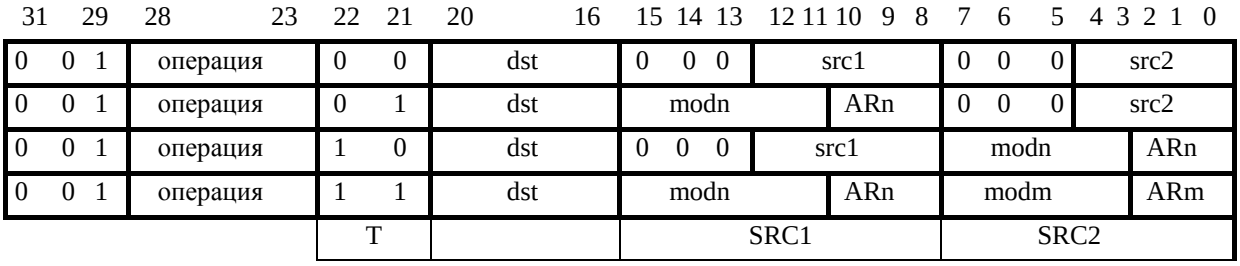


Рисунок А.5.21 – Кодирование для трёхоперандных режимов адресации

A.5.2.3 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. Операнды источника, обозначенные src1 и src2, используют регистры повышенной точности. Выполняемые параллельные операции обозначены как «операция».

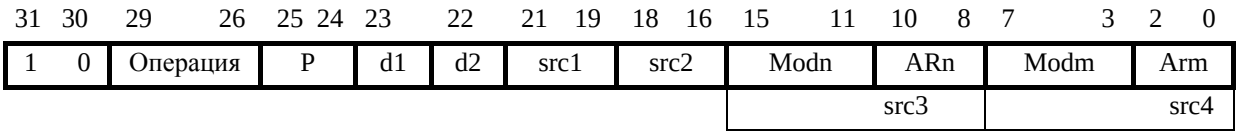


Рисунок А.5.22 – Кодирование для параллельных режимов адресации

Поле параллельного режима адресации (P) определено по использованию операндов, т. е. являются они операндами источника или назначения. Связь между полем P и операндами подробно представлена в описании отдельных параллельных команд (см. раздел А.10 «Команды языка ассемблер»). Однако операнды всегда кодируются одинаково. В разрядах 31 и 30 установлено значение 10, обозначающее команды параллельного режима адресации. Разряды 21-0 определяют поле параллельного режима адресации (P), которое определяет как разряды 21-0 интерпретируются 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, $0 \leq$ src1 ≤ 7 (регистры повышенной точности R0-R7);

src2, $0 \leq$ src2 ≤ 7 (регистры повышенной точности R0-R7);

d1, если 0, dst1 это R0; если 1, dst1 это R1;

d2, если 0, dst2 это R2; если 1, dst2 это R3;

P, $0 \leq P \leq 3$;

src3 – косвенная (disp = 0, 1, IR0, IR1);

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

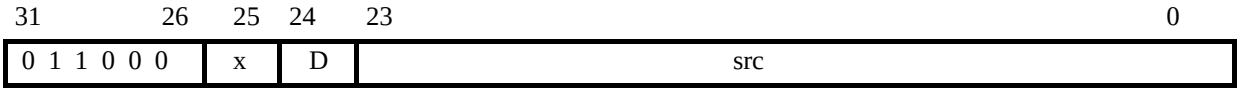
Как в трёхоперандном режиме адресации, косвенная адресация в параллельном режиме адресации допустима для смещения 1 или 0 и при использовании индексных регистров (IR0 и IR1). Замена 1 неявная и не записывается явно в командном слове.

Если поля src1 и src2 используют одинаковые вспомогательные регистры, то оба адреса сгенерированы правильно, но только значение, созданное полем src3, хранится в определённом вспомогательном регистре. Ассемблер выдаёт предупреждение, если это условие определено пользователем.

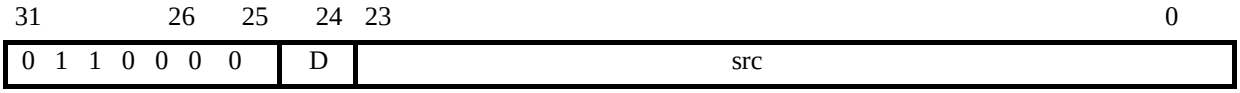
А.5.2.4 Режим длинной непосредственной адресации

Режим длинной непосредственной адресации, см. рисунок А.5.23, используется для кодирования инструкций управления программой (BR, BRd и CALL), для которых полезно иметь 24-разрядный адрес, содержащийся в командном слове. Безусловные переходы BR (стандартный) и BRD (задержанный) используют режим длинной непосредственной адресации. В разрядах 31-26 установлено значение 011000, 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 – это 24-разрядный src.



или для BR(D):



или для CALL:

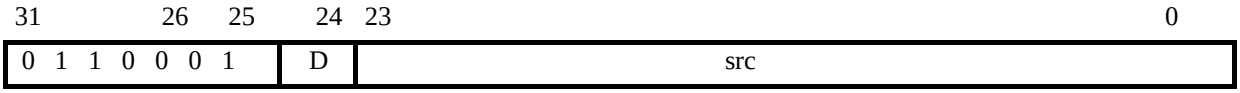


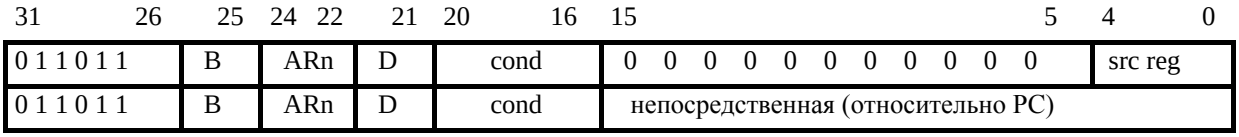
Рисунок А.5.23 – Кодирование для режима длинной непосредственной адресации

А.5.2.5 Режимы адресации условных переходов

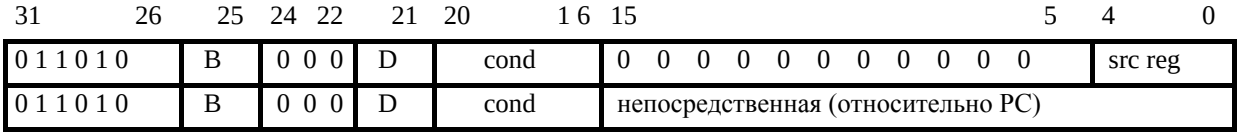
Команды, использующие режимы адресации условных переходов (Bcond, BcondD, CALLcond, DBcond и DBcondD), могут выполнять различные условные операции. В разрядах 31-27 установлено значение 01101, обозначающее команды режима адресации условных переходов; разряд 26 установлен в 0 или 1, первый выбирает DBcond, другой – Bcond. Выбор 25 разряда определяет режим адресации условных переходов (B).

На рисунке А.5.24 показано кодирование для режима адресации условных переходов.

DBcond(D):



Bcond(D):



CALLcond(D):

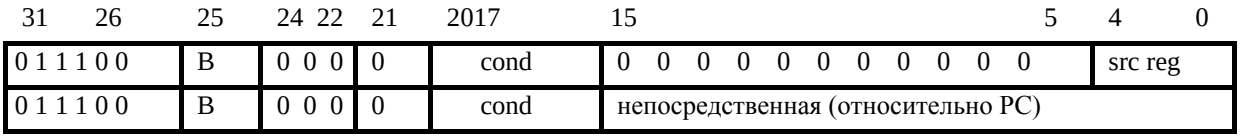


Рисунок А.5.24 – Кодирование для режимов адресации условных переходов

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 условия (cond) определяет условие, которое проверяется для определения того, какое действие необходимо выполнить, т. е. должно ли происходить ветвление.

A.5.3 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, которое содержит 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. В этом подразделе описан циклический режим 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, 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.

Циклический буфер размером R должен начинаться на границе K разряда (т. е. K младших значащих разрядов 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 (31-word) циклический буфер должен начинаться с адреса, чьи 5 младших значащих разрядов = 0 (т. е. XXXXXXXXXXXXXXXXXXXXXXXXX00000) и значение 31 должно быть загружено в регистр BK.

В циклической адресации «index» относится к N младшим значащим разрядам выбранного вспомогательного регистра, и «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 $0 \leq \text{index} + \text{step} < \text{BK}$:

index = index + step.

Если index + step ≥ BK:

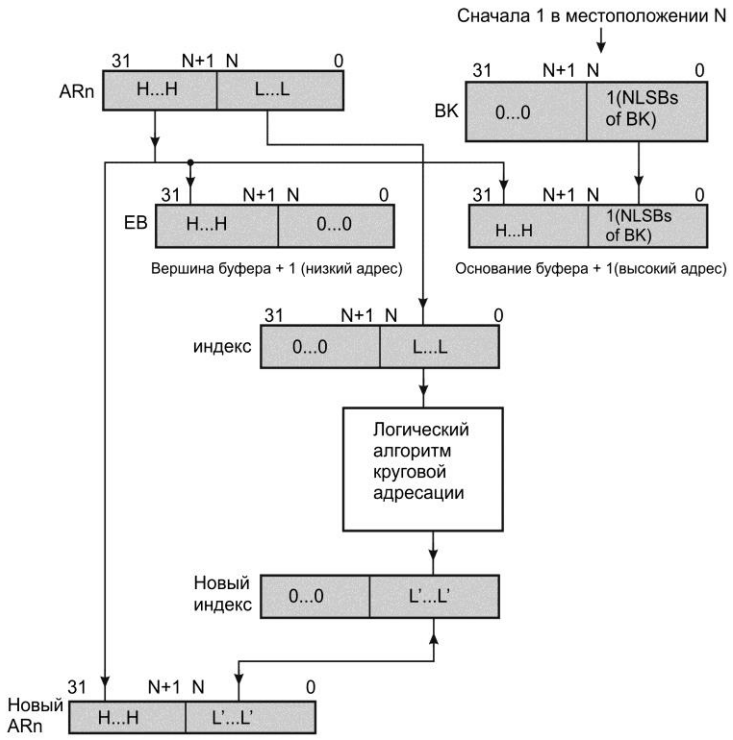
index = index + step - BK.

Если index + step < 0:

index = index + step + BK.

Циклическая адресация особенно полезна для реализации КИХ-фильтров (фильтров с конечной импульсной характеристикой).

Блок-схема циклической адресации представлена на рисунке А.5.25.



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 А.5.25 – Блок-схема циклической адресации

На рисунке А.5.26 показана реализация циклического буфера. Это иллюстрирует взаимосвязь сгенерированных величин и элементов в циклическом буфере.

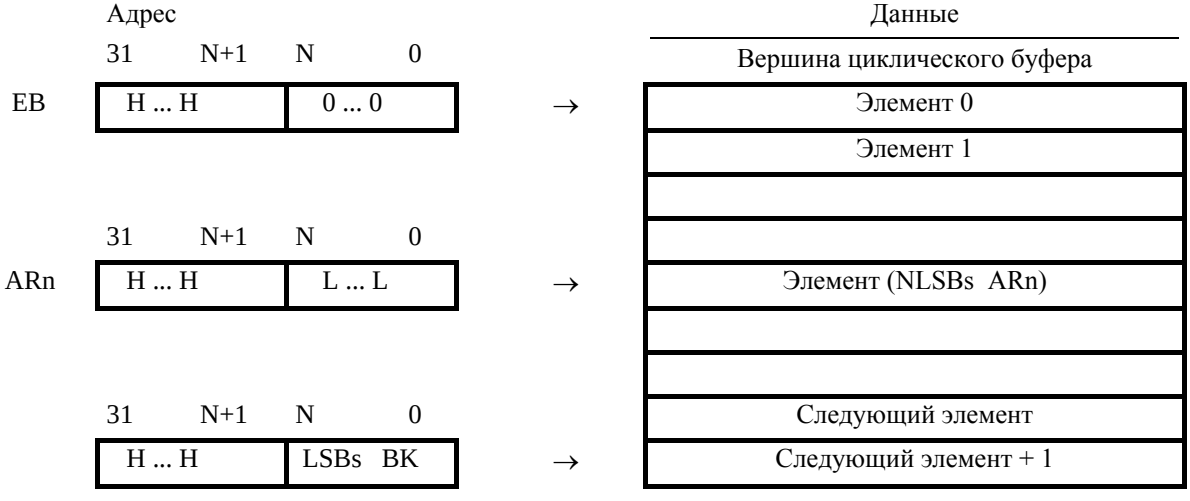


Рисунок А.5.26 – Реализация циклического буфера

На рисунке А.5.27 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 то, 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 А.5.27 – Пример циклической адресации

А.5.4 Битреверсная адресация

Битреверсная адресация в ПЦОС улучшает характеристики скорости выполнения и использования памяти программ для алгоритмов БПФ, которые используют различные основания. Базовый адрес битреверсной 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; 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 битреверсную адресацию, генерируется адрес в битреверсной форме.

Рисунок А.5.28 показывает последовательность модификаций AR2 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 А.5.28 – Пример битреверсной адресации

Чтобы проиллюстрировать этот вид адресации, были взяты восьмиразрядные вспомогательные регистры. Например, AR2 содержит значение 0110 0000 (96). Это базовый адрес данных в памяти. Регистр IR0 содержит значение 0000 1000 (8).

В таблице А.5.3 приведены отношения шагов индекса и четырёх младших разрядов AR2.

Как видно, можно найти четыре младших значащих разряда реверсированием набора разрядов шагов.

Таблица А.5.3 – Шаг индекса и битреверсная адресация

Шаг	Набор разрядов	Битреверсный набор	Битреверсный 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

А.5.5 Управление системным стеком и стеком пользователя

ПЦОС обеспечивает специальный указатель системного стека (SP) для построения стеков в памяти. Вспомогательный регистр также может быть использован для построения множества общих линейных списков. Далее обсуждается реализация следующих способов линейных списков:

- стек – линейный список, для которого все вставки и удаления выполняются в одном конце списка;
- очередь – линейный список, для которого все вставки выполняются в одном конце списка и все удаления выполняются в другом конце списка;
- удаление – линейный список очереди с двумя концами, для которого вставки и удаления могут выполняться в любом из концов списка;
- указатель системного стека (SP) – это 32-разрядный регистр, который содержит адрес вершины системного стека. Системный стек заполняется от младших адресов памяти к старшим. SP всегда указывает на следующий элемент, подталкиваемый в стек.

При проталкивании данных в стек выполняется прединкремент, а при выталкивании – постдекремент указателя системного стека.

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

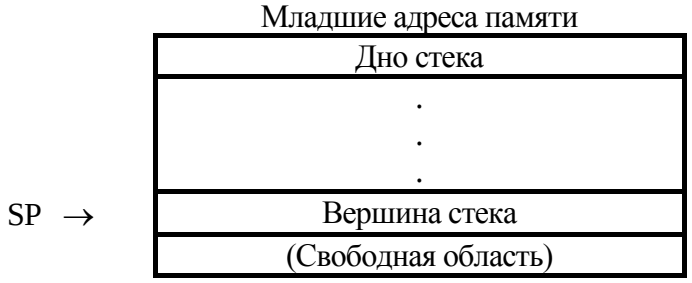


Рисунок А.5.29 – Конфигурация системного стека

А.5.5.1 Стеки

Стеки могут быть построены от младших адресов памяти к старшим или от старших к младшим. Они реализуются с использованием режимов прединкремента/декремента и постинкремента/декремента изменения вспомогательных регистров (AR). Изменение стека от старших адресов памяти к младшим может быть выполнено по двум вариантам:

- 1 вариант: записывает данные в память, используя ***– –**ARn для проталкивания данных в стек, и считывает из памяти, используя ***ARn ++** для выталкивания данных из стека;

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

Рисунок А.5.30 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 А.5.30 – Реализация стеков от старших адресов памяти к младшим

Изменение стека от младших адресов памяти к старшим также может быть реализовано по двум вариантам:

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

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

На рисунке А.5.31 приведены эти два варианта.

В варианте 3 AR всегда указывает на вершину стека, а в варианте 4 AR всегда указывает на следующую свободную область в стеке.

Рисунок А.5.31 – Реализация стеков от младших адресов памяти к старшим

А.5.5.2 Очереди

Очередь подобна FIFO и реализуется на основе манипуляций со вспомогательными регистрами. Для очередей используются два вспомогательных регистра, один – для адресации начала очереди, с которого данные выталкиваются из стека (или удаляются из очереди), другой – для адресации конца очереди, где данные вталкиваются в стек. При корректном управлении вспомогательными регистрами, очередь может быть циклической (в циклической очереди указатель её конца разрешается использовать и для указания на начало очереди).

A.6 Управление выполнением программы

ПЦОС обеспечивает полное множество гибких и мощных конструкций, которые обеспечивают управление программным и аппаратным обеспечением выполнения программы. Программное управление обеспечивает повторения, ветвления, вызовы, системные прерывания и возвраты. Аппаратное управление обеспечивает выполнение операции, сброс и прерывания. Т. к. программирование включает множество различных конструкций, можно выбрать одну, наиболее подходящую для конкретного случая.

Несколько команд блокировки обеспечивают гибкую поддержку мультипроцессорного режима работы через использование внешних сигналов, являющихся мощным средством синхронизации. Они также гарантируют целостность связи и результата в высокоскоростных операциях.

ПЦОС поддерживает сигнал немаскируемого внешнего сброса и некоторое количество внутренних и внешних прерываний. Эти функции могут быть запрограммированы для специального применения.

В данном разделе рассматриваются следующие основные темы:

- режимы повторений (подраздел A.6.1);
- инициализация (операции) (A.6.1.1 – A.6.1.3);
- задержанные переходы (подраздел A.6.2);
- вызовы, системные прерывания и возвраты (подраздел A.6.3);
- операции блокировки (подраздел A.6.4);
- операция сброса (подраздел A.6.5);
- прерывания (подраздел A.6.6).

A.6.1 Режимы повторений

Режимы повторений ПЦОС могут применяться для реализации циклов без дополнительных потерь. Для многих алгоритмов наибольшее время тратится на реализацию внутреннего ядра программы. Использование режимов повторений позволяет максимально сократить время выполнения критичных по времени секций программы.

ПЦОС обеспечивает две команды для реализации циклов без дополнительных потерь:

- режим RPTB – повторение блока кода;
- режим RPTS – повторение одной команды.

RPTB позволяет выполнять блок кода определённое количество раз.

RPTS обеспечивает повторение одной команды определённое количество раз и уменьшает нагрузку шины посредством выбора команды только один раз.

Три регистра (RS, RE и RC) связаны с установкой счётчика программ, если установка происходит в режиме повторения. В таблице A.6.1 описаны эти регистры.

Таблица A.6.1 – Регистры режима повторения

Регистр	Функция
Регистр начального адреса повторений RS	Содержит адрес первой команды повторяемого блока команд.
Регистр конечного адреса повторений RE	Содержит адрес последней команды повторяемого блока кода.
Регистр счёта повторений RC	Содержит значение, на 1 меньшее числа повторений блока кода. Например, для выполнения блока N раз, необходимо загрузить N-1 в RC.

A.6.1.1 Инициализация режима повторений

Существует два разряда, которые очень важны для операций RPTB 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 и RPTS. Если S=0, то ЦПУ не выполняет выборки в режиме одного повторения. Если S=1 и RM=1, то ЦПУ выполняет выборки в режиме одного повторения.

Для корректной работы режимов повторений требуется, чтобы все вышеописанные регистры и поля регистра состояния были правильно инициализированы.

Команды RPTB и RPTS выполняют эту инициализацию с некоторыми отличиями, (см. A.6.1.2 и A.6.1.3).

A.6.1.2 Инициализация RPTB

Когда выполняется RPTB src, выполняются следующие операции

Операция 1: PC + 1 → RS

Операция 2: src → RE

Операция 3: 1 → Разряд RM регистра состояния

Операция 4: 0 → S разряд.

Операция 1 загружает начальный адрес блока в RS. Операция 2 загружает src в RE (конечный адрес блока). Операнд src – это 24-разрядная величина, содержащаяся в командном слове. Операция 3 устанавливает регистр состояния для указания режима повторения операции. Операция 4 показывает, что это режим повторения блока.

Следующая требующаяся информация – это количество повторений блока.

Значение определяется инициализацией регистра RC (счётчик повторений). Так как выполнение RPTB не загружает RC, этот регистр должен загружаться пользователем явно. Типичная установка операции повторения блока показана ниже:

```
LDI    15, RC;  15 → RC
RPTB   LOOP;  LOOP → RE, PC + 1 → RS, 1 → RM, 0 → S
```

Режимы повторений повторяют блок кодов команд, по крайней мере, один раз в обычной операции. Счётчик операций будет загружаться числом, на единицу меньшим, чем число повторений блока, так что значение 0 в RC повторяет блок кода один раз. Все повторения блоков, начатые RPTB, могут быть прерваны.

A.6.1.3 Инициализация RPTS

Во время выполнения RPTS src производится следующая последовательность операций.

Операция 1: PC + 1 → RS

Операция 2: PC + 1 → RE

Операция 3: 1 → Разряд RM регистра состояния

Операция 4: 1 → Бит S

Операция 5: src → RC

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.

Операнд src загружается в RC (операция 5), и следующая команда выполняется src+1 раз.

Повторения одной команды, указанной RPTS, не являются прерываемыми, так как RPTS выбирает командное слово только один раз и хранит его в регистре команд для многократного использования. Прерывание 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 иметь одну команду, которую можно и повторять, и прерывать, то можно использовать команду RPTB.

A.6.1.4 Работа режима повторения

Информация в регистрах режима повторения и соответствующих разрядах управления используется для мониторинга изменений программного счётчика PC, когда производится выборка в режиме повторения. В режимах повторения PC компарируется с содержимым регистра RE. Если они соответствуют друг другу и счётчик повторений неотрицательный, то счётчик повторений уменьшается, PC загружается с начального адреса повторений и обработка продолжается. Выборка и соответствующие разряды состояния при необходимости изменяются.

Примечание – Счётчик повторений (RC) никогда не изменяется, если RM равно нулю. Число повторений максимально, если RC=8000_0000h, что соответствует числу повторений 8000_0001h.

Подробно алгоритм изменения PC показан в примере A.6.1.

Пример A.6.1 – 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
выбор команды из памяти    ; Выборка команды из памяти
else                       ; Если не первая выборка
выбор 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
выбор 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
```

Пример А.6.2 показывает типичную установку блока повторений.

Пример А.6.2 – Выполнение RPTB

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

Использование этого режима изменения PC позволяет проанализировать, что произойдёт в случае ветвлений внутри блока. Предполагается, что следующее значение PC будет либо PC+1, либо содержимым регистра RS. Таким образом, этот метод повторения блока допустим для любого количества ветвлений внутри блока повторений. Выполнение может идти в любом месте внутри пользовательской программы через прерывания, вызовы подпрограмм и т. д. Для надлежащей модификации счётчика цикла должна быть выбрана последняя команда цикла.

Повторение цикла может быть остановлено до завершения посредством занесения 0 в счётчик повторения или занесением 0 в разряд RM регистра состояния.

Так как режим повторения блока изменяет счётчик программ, другие команды не могут изменять счётчик программ в то же самое время.

Необходимо придерживаться двух правил:

- следующая команда в блоке (или только одна команда в блоке единичного размера) не может быть: Bcond, BR, Dbcond, CALL, CALLcond, TRAPcond, RETIcond, RETScond, IDLE, RPTB или RPTS. Пример А.6.3 иллюстрирует некорректное размещение стандартного перехода;

- ни одна из следующих четырёх команд не может быть в конце блока (или только одна команда в блоке единичного размера): BcondD, BRD или DBcondD.

Пример А.6.3 показывает некорректное размещение задержанного ветвления.

Если какое-либо из этих правил нарушено, PC будет не определен!

Пример А.6.3 – Некорректное размещение стандартного перехода

```
LDI    15, RC                 ; Загрузка 15 в счётчик повторений
RPTB   ENDLOP                 ; Выполнение блока кода
STLOOP                        ; от STLOOP до ENDLOP 16 раз
```

```
.
.
.
ENDLOP   BR   OOPS      ; Этот переход нарушает правило 1
```

Пример А.6.4 – Некорректное размещение задержанного перехода

```
LDI    15, RC              ; Загрузка счётчика повторений с 15
RPTB      ENDLOP           ; Выполнение блока кода
STLOOP                     ; от STLOOP до ENDLOP 16 раз
BRD    OOPS                ; Этот переход нарушает правило 2
ADDF
MPYF
ENDLOP   SUBF
```

Повторения блоков (RPTB) могут быть вложенными. Так как все управление определено регистрами RS, RE, RC и ST, допустимо хранение и восстановление этих регистров для обеспечения их вложенности. Разряд RM в регистре состояния может быть использован для определения активности режима повторения блока. Например, если составляется программа обслуживания прерывания, требующая использование RPTB, возможно, что прерывание, связанное с программой, может происходить во время другого повторения блока. Программа обработки прерывания может проверять разряд RM для определения того, является ли активным режим повторения. Если этот разряд установлен, программа прерывания сохраняет RS, RE, RC и ST. Затем программа прерывания может выполнять повторение блока. Перед возвратом в прерванную программу, программа прерывания восстанавливает регистры RS, RE, RC и ST. Если разряд RM не установлен, то сохранять и восстанавливать эти регистры нет необходимости.

А.6.2 Задержанные переходы

В ПЦОС 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 на ПЦОС выполняется за четыре цикла. В этот класс включены вызовы, возвраты и системные 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 ПЦОС. Однако, в отличие от режимов повторения блоков, задержанные переходы могут быть использованы не только для организации цикла. Каждому задержанному переходу соответствует стандартный переход, применяемый, если задержанный переход не может быть использован. Задержанные (**D**elayed) переходы в ПЦОС реализуются инструкциями Bcond**D**, BR**D** и DBcond**D**.

Условные задержанные переходы используют условия, которые возникают в конце исполнения инструкции, непосредственно предшествующей задержанному переходу. Флаги условий не зависят от команд, следующих за задержанным переходом. Флаги условий устанавливаются предыдущей командой тогда, когда регистром назначения этой команды является один из регистров повышенной точности (R7-R0) или при выполнении команд сравнения (CMPF, SMPF3, CMPI, CMPI3, TSTB или TSTB3). Задержанные переходы гарантируют, что три следующие команды будут выполнены, независимо от других конфликтов конвейера.

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, BcondB, BR, BRD, DBcond, DBcondB, CALL, CALLcond, TRAPcond, RETIcond, RETScond, RPTB, RPTS или IDLE, см. пример А.6.5.

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

При некорректном использовании задержанных переходов PC будет не определен!

Пример А.6.5 – Некорректное размещение задержанного перехода

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

А.6.3 Вызовы, системные прерывания и возвраты

Вызовы и системные прерывания позволяют реализовывать выполнение подпрограмм или функций с последующим возвратом в основную программу.

Команды CALL, CALLcond и TRAPcond сохраняют значение PC в стеке перед изменением содержимого PC, позволяя в последующем обеспечить возврат к основному программному коду, используя команды RETScond или RETIcond.

Команда CALL сохраняет значение следующего PC в стеке и загружает в PC операнд src (24-разрядное непосредственное значение). Команда CALLcond подобна команде CALL, за исключением того, что:

- она выполняется тогда, когда определённое условие есть «истина» (20 условий – включая безусловные – приведены в подразделе А.10.2);
- src является либо относительным (относительно PC) смещением, либо определено регистровым режимом адресации.

Флаги условий устанавливаются предыдущей командой, только когда регистром назначения является один из регистров повышенной точности (R7-R0), либо когда выполняется одна из команд сравнения CMPF, CMPF3, CMPI, CMPI3, TSTB или TSTB3.

Команда TRAPcond также выполняется тогда, когда определённое условие есть «истина» (условия те же, что и для CALLcond). При выполнении команды прерывания запрещены записью 0 в разряд GIE ST, следующее значение PC сохраняется в стеке, и вектор выбирается одним из адресов 20h - 3Fh и загружается в PC.

Соответствующий адрес определяется номером системного прерывания в команде TRAP. Использование RETIcond для возврата опять разрешает прерывания.

RETScond возвращает выполнение от одной из вышеперечисленных команд, загружая вершину стека в PC. Для выполнения команды необходимо, чтобы определённое условие было «истина». Условия те же, что и для CALLcond.

RETIcond возвращает из системного прерывания или вызова аналогично команде RETScond и, кроме того, устанавливает бит GIE в регистре состояния, таким образом, разрешая прерывания, разряды, разрешения которых установлены в 1. Условия те же, что и для CALLcond.

На самом деле, вызовы и системные прерывания завершаются практически одинаково (т. е. подфункция вызывается, выполняется, затем управление возвращается к вызывающей функции). Системные прерывания имеют следующие преимущества:

- прерывания автоматически запрещаются при выполнении системного прерывания. Это позволяет выполнять критичную программу без риска прерываний, поэтому для разрешения прерываний используется команда RETIcond;

- можно использовать системные прерывания для косвенного вызова функций. Это практически незаменимо, когда ядро программы содержит основные подфункции для использования в приложениях. В этом случае функции в ядре могут быть изменены и переадресованы без перекомпиляции каждой прикладной программы.

A.6.4 Операции блокировки

Одна из наиболее распространённых операций в мультипроцессорных конфигурациях – разделение основной памяти многими процессорами. Для того чтобы разрешить нескольким процессорам доступ к основной памяти и совместное использование данных последовательным методом, необходимы некоторые виды арбитража или квитирования установления связи. Обеспечению требований по арбитражу служат операции блокировки ПЦОС.

ПЦОС обеспечивает гибкие средства поддержки мультипроцессорной конфигурации с помощью пяти команд, относящихся к операциям блокировки. Используя внешние сигналы, эти команды обеспечивают механизм общей синхронизации. Они также гарантируют целостность связей и результата при высокоскоростных операциях.

Операции блокировки используют два внешних вывода флага – XF0 и XF1.

XF0 должен быть определён как выход и XF1 – как вход. Когда флаги сконфинурированы подобным способом, XF0 сигнализирует запрос операции блокировки, а XF1 действует как сигнал подтверждения для запрошенной операции блокировки. В этом режиме XF1 и XF0 имеют активный низкий уровень.

Внешняя синхронизация для блокированных загрузок и сохранений в памяти такая же, как стандартная загрузка и запись в памяти. Блокированные загрузки и сохранения в памяти могут быть продлены подобно стандартным доступам с использованием соответствующего сигнала готовности (RDY# int или XRDY# int).

Команды LDFI и LDII выполняют следующие действия:

- одновременная установка XF0 в 0 и начало цикла чтения. Временная диаграмма XF0 подобна адресной шине во время выполнения цикла чтения;
- выполняется команда LDF или LDI и продляется цикл считывания до тех пор, пока XF1 не станет нулём и будет выдан сигнал готовности (RDY# int или XRDY# int);
- снимается установка XF0 в 0 и завершается цикл чтения.

Операция чтение/запись идентична любому другому циклу чтение/запись, за исключением особого использования XF0 и XF1. Операнд src для LDFI и LDII всегда прямой или косвенный адрес памяти. XF0 устанавливается в 0, если src расположен во внешней памяти (т. е. STRB#, MSTRB# или IOSTRB# активны) или src – одно из внутренних периферийных устройств. При обращении к памяти ядра XF0 не устанавливается, и выполнение операции аналогично выполнению LDF или LDI во внутренней памяти.

Команды STFI и STII выполняют следующие операции:

- одновременно устанавливают XF0 в 1 и запускают цикл записи. Временная диаграмма XF0 подобна диаграмме адресной шины во время выполнения цикла записи;
- выполняются команды STF и STI и цикл записи продляется до тех пор, пока не будет сигнализирована готовность (RDY# int или XRDY# int).

Как в случае для LDFI и LDII, размещение dst операций STFI и STII влияет на выработку XF0. Если dst во внешней памяти (STRB#, MSTRB# или IOSTRB# активны) или dst – это одно из внутриядерных периферийных устройств, XF0 устанавливается в 1. При обращении к внутренней памяти XF0 не устанавливается, и выполнение операции аналогично выполнению STF или STI во внутренней памяти.

Функции команды SIGI следующие:

- устанавливает XF0 в 0;
- удерживается до тех пор, пока XF1 не установится в 0;
- устанавливает XF0 в 1 и завершает операцию.

Несмотря на то, что команды LDFI, LDII и SIGI ожидают установки XF1, они могут быть прерваны. LDFI и LDII для прерывания требуют сигнала готовности (RDY# int и XRDY# int). Так как прерывание имеет место на границах цикла шины (см. подраздел A.6.6 «Прерывания»), прерывание может иметь место некоторое время спустя установки разрешения. Это позволяет применять механизм защиты от тупиковых условий, используя прерывание блокированной загрузки, которая продолжается слишком долго. На выходе из прерывания выполняется следующая команда. Команды STFI и STII не могут быть прерваны. Так как команды STFI и STII завершаются при сигнализации готовности, задержка до возникновения прерывания может быть такой же, как и для любых других команд.

Операции блокировки могут быть использованы для выполнения ждущего цикла (цикла ожидания сигнала занятости), для управления мультипроцессорным счётчиком, для реализации простого механизма семафора или для обеспечения синхронизации двух ПЦОС. Следующие примеры иллюстрируют применение команд операций блокировки.

Пример А.6.6 показывает реализацию ждущего цикла. Если размещение LOCK является блокировкой для критической секции программы и ненулевое состояние означает, что блокировка занята, то может быть использован алгоритм для ждущего цикла.

Пример А.6.6 – Ждущий цикл

```
        LDI   1,R0              ; Заносит 1 в R0
L1:     LDII  @LOCK, R1         ; Начало операции блокировки
                                ; Содержимое LOCK → R1
        STII  R0, @LOCK         ; Заносит R0 (=1) в LOCK, XF0=1
                                ; Конец операции блокировки
        BNZ   L1                ; Сохраняет пока LOCK=0
```

Пример А.6.7 показывает, как положение COUNT может содержать значение счётчика количество специальных операций, которые необходимо выполнить. Эти операции могут быть выполнены любым процессором в системе. Если значение счётчика равно нулю, процессор может ожидать до тех пор, пока значение счётчика не станет ненулевым до начала обработки. Алгоритм для изменения COUNT показан в примере А.6.7.

Пример А.6.7 – Мультипроцессорный счётчик операций

```
CT:  OR    4, IOF               ; XF0=1
                                ; Конец операции блокировки
     LDII  @COUNT, R1           ; Начало операции блокировки
                                ; Содержимое COUNT → R1
     BZ    CT                   ; Если COUNT=0, продолжает попытки
     SUBI  1, R1                ; Уменьшение R1 (=COUNT)
     STII  R1, @COUNT           ; Изменяет COUNT, XF0=1
                                ; Конец операции блокировки
```

Иногда нескольким процессорам необходимо иметь доступ к общим данным или другим общим ресурсам. Часть кода, которая должна обеспечивать доступ к общим данным, называется критической секцией.

Чтобы упростить программирование для критических секций, используют семафоры. Семафоры – это переменные, которые могут принимать только неотрицательные значения. Две простых неделимых операции определены для семафоров (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 эти процессы обращаются и модифицируют семафор S, они только выполняют доступ и модификацию S.

Для ввода критической секции операция P выполняется на общем семафоре, устанавливая S (S инициализируется в 1). Первый процессор, выполняющий P(S), будет иметь возможность вводить эти критические секции. Все другие процессоры блокированы, потому что S стало равным 0. После прохождения этой критической секции процессор выполняет V(S), таким образом, разрешая другому процессору успешно выполнить P(S).

Код ПЦОС для V(S) приведён в примере А.6.8, и код для P(S) приведён в примере А.6.9.

Пример А.6.8 – Реализация V(S)

```
V:    LDII   @S, R0     ; Начало блокированного чтения (XF0 = 0)
                        ; Содержимое S → R0
      ADDI   1, R0      ; Инкремент R0 (=S)
      STII   R0, @S     ; Изменение S, конец блокировки (XF0=0)
```

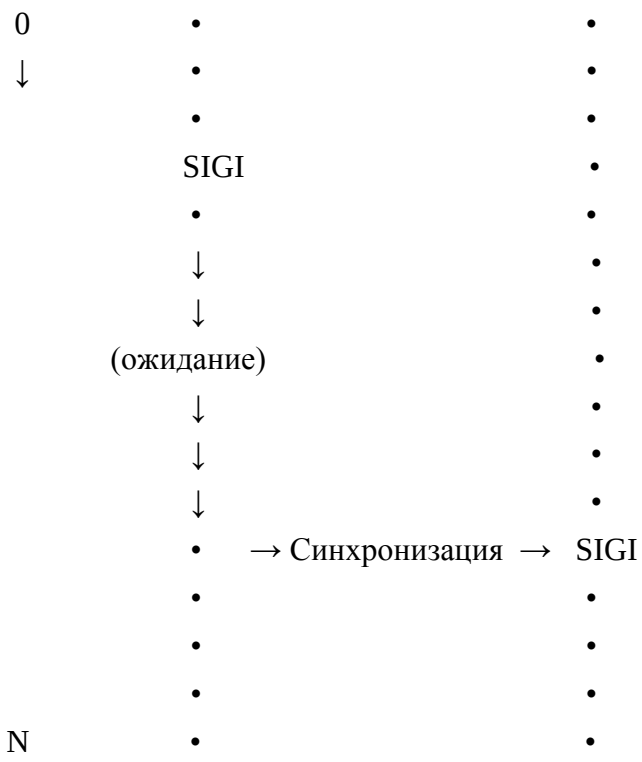
Пример А.6.9 – Реализация P(S)

```
P:    OR     4, IOF     ; Конец блокировки (XF0=1)
      LDII   @S, R0     ; Начало блокированного чтения S
                        ; Содержимое S → R0
      BZ     P          ; Если S=0, переход на P и проверка заново
      SUBI   1, R0      ; Декремент R0 (=S)
      STII   R0, @S     ; Изменение S, конец блокировки (XF0=1)
```

Операция SIGI может быть использована для синхронизации на командном уровне нескольких ПЦОС; программный код приведён в примере А.6.10.

Процессор 1 работает до тех пор, пока он выполняет SIGI. И далее он ожидает до тех пор, пока процессор 2 выполняет SIGI. В этой точке два процессора синхронизируются и продолжают выполнение.

Пример А.6.10 – Программа для синхронизации двух ПЦОС на программном уровне



А.6.5 Операция сброса

ПЦОС имеет внешний сигнал сброса RESET#, который используется для обеспечения сброса системы. Данный подраздел описывает операцию сброса.

При включении питания состояние процессора ПЦОС не определено. Для установки процессора в известное состояние используется сигнал RESET#. Этот сигнал должен быть установлен в низкий уровень в течение 10 и более циклов H1 для гарантированного сброса системы. H1 – выходной тактовый сигнал, генерируемый ПЦОС.

Сброс воздействует на другие выводы процессора синхронно или асинхронно. Синхронный сброс зависит от внутренней синхронизации ПЦОС. Асинхронный сброс прямо влияет на выводы и происходит быстрее синхронного.

При системном сбросе выполняются следующие дополнительные операции:

- сбрасывается периферия. Это синхронная операция. Сброс периферии описан в разделе А.8 «Периферийные устройства»;
- сбрасываются регистры управления внешней шиной. Значения регистров управления при сбросе описаны в разделе А.7 «Работа с внешней шиной»;
- в следующие регистры ЦПУ загружается 0:
    - ST (регистр состояния ЦПУ);
    - IE (флаги разрешения прерывания ЦПУ/ПДП);
    - IF (флаги прерывания ЦПУ);
    - IOF (флаги ввода-вывода);
- вектор сброса считывается по адресу памяти 0h и загружается в PC (счётчик команд). Этот вектор содержит стартовый адрес программы сброса;
- начинается выполнение программы инициализации;
- система из нескольких ПЦОС с общей системной синхронизацией может быть сброшена и засинхронизирована. При переходе сигнала RESET# из 1 в 0 процессор переводится в известное состояние внутренней фазировки, поэтому все ПЦОС переходят в одно состояние по внутренней синхронизации.

Кроме вышеописанных, состояние остальных регистров после сброса не определено.

А.6.6 Прерывания

ПЦОС поддерживают несколько внутренних и внешних прерываний, которые могут использоваться в различных приложениях. В текущем подразделе обсуждается действие данных прерываний.

Дополнительная информация, относящаяся к внутренним прерываниям, приведена в разделе А.8 «Периферийные устройства».

Внешние прерывания имеют внутреннюю синхронизацию, что иллюстрируется тремя триггерами, синхронизируемыми сигналами H1 и H3. Затем вход прерывания устанавливает соответствующий разряд регистра флага прерываний (IF), если прерывание активно.

Внешние прерывания записываются внутри по заднему фронту H1. Внешнее прерывание должно удерживаться в низком уровне, по меньшей мере, в течение одного цикла H1/H3 для определения его процессором.

Прерывания должны удерживаться в низком уровне в течение такого времени, чтобы задний фронт сигнала H1 попадал в данный промежуток от одного до двух раз. Если прерывание удерживается в низком уровне более чем при трёх задних фронтах сигнала H1, может быть определено более одного прерывания.

A.6.6.1 Разряды управления прерываниями

Когда соответствующее прерывание обработано ЦПУ или контроллером ПДП, соответствующий разряд флага прерываний очищается внутренним сигналом подтверждения прерывания. Необходимо отметить, что, если сигнал INTn все ещё в низком уровне после появления сигнала подтверждения прерывания, разряд флага прерывания будет очищен только на один цикл, после чего установлен снова, т. к. сигнал INTn все ещё в низком уровне. Соответственно, теоретически возможно, что в зависимости от того, когда считан регистр IF, этот разряд может быть нулём, даже если сигнал INTn в низком уровне. При сбросе ПЦОС в регистр флага прерываний записываются нули, очищая, таким образом, все необработанные (ожидающие обработки) прерывания.

Разряды регистра флагов прерываний могут быть считаны и записаны из программы. Запись 1 в соответствующий разряд IF устанавливает соответствующий флаг прерывания в 1. Соответственно, запись 0 сбрасывает соответствующий флаг прерывания в 0. Таким образом, все прерывания могут быть записаны и/или очищены программным путём. Т. к. флаги прерываний могут быть считаны, выводы прерываний могут быть включены как программные, если не требуется интерфейс, управляемый по прерываниям.

Внутренние прерывания работают подобным же образом. Разряды регистра IF, относящиеся к внутренним прерываниям, могут быть считаны и записаны из программы. Запись 1 в соответствующий разряд IF устанавливает соответствующий флаг прерывания в 1, запись 0 – очищает его. Все внутренние прерывания имеют длину в один цикл H1/H3.

Разряд разрешения глобального прерывания ЦПУ (GIE), расположенный в регистре состояния ЦПУ (ST), управляет всеми прерываниями ЦПУ. Все прерывания ПДП (DMA) управляются разрядом разрешения глобального прерывания ПДП, который не зависит от ST (GIE) и относится только к ПДП. Разряд разрешения глобального прерывания ПДП зависит, в частности, от состояния разрядов DMA SYNCH. Они впрямую не доступны из программы. Результат операции AND над содержимым разряда флага прерывания и разрешениями прерывания поступает в блок обработки внутренних прерываний.

Для обеспечения максимальной производительности в обработке прерываний, предусмотрена команда подтверждения прерывания (IACK). IACK управляет выводом IACK# и обеспечивает холостое (фиктивное) чтение. Чтение выполняется по адресу, определяемому операндом команды IACK. При использовании IACK он обычно установлен на начало программы обработки прерывания. Для соответствующих приложений он может быть установлен на конец программы обработки прерываний или вовсе не является необходимым.

А.6.6.2 Рассмотрение прерываний в ПЦОС

Следует внимательно рассмотреть вопросы, касающиеся прерываний ПЦОС, в особенности при изменении содержимого регистра состояния, когда устанавливается разряд глобального разрешения прерывания (GIE). Это может привести к неправильной установке и сбросу разряда GIE, как описано ниже.

Разряд GIE установлен в 0 при прерывании. Это может привести к сбою процессора, если в течение двух циклов по определению прерывания из какой-либо команды делается попытка чтения или изменения регистра состояния.

Например, если регистр состояния записывается в стек, он будет сохранен неправильно, если прерывание будет подтверждено двумя циклами ранее команды сохранения.

Когда сигнал прерывания определен, ПЦОС продолжает выполнение команд, находящихся в стадии чтения и декодирования на конвейере. Однако, поскольку прерывание подтверждено, разряд GIE сбрасывается в 0, и команда сохранения, находящаяся в конвейере, запишет неверное значение регистра состояния. Например, если программа выглядит следующим образом:

```
                        ...
                        NOP
Прерывание определено → LDI   @V_ADDR, AR1
                        MPYI  *AR1, R0
                        PUSH  ST
                         ...
                        POP   ST
                         ...
```

– то команда PUSH ST сохранит содержимое ST в памяти, при этом GIE=0.

Так как предполагается, что устройство имеет GIE=1, то команда POP ST возвратит неверное значение регистра состояния в ST.

Подобная же ситуация может возникнуть, если GIE =1 и выполняемая инструкция предназначена для изменения других разрядов состояния без установки разряда GIE.

В вышеприведённом примере подобная ошибка может возникнуть при определении прерывания за два цикла до команды POP ST. В этом случае прерывание очищает разряд GIE, но выполнение команды POP ST устанавливает разряд GIE. Т. к. прерывание было определено, то программа обработки прерываний будет выполнена с разрешёнными прерываниями, в то время как ожидался запрет.

В подобной ситуации одним из решений может быть использование TRAP.

Например, можно использовать TRAP0 для сброса GIE и TRAP1 для его установки. TRAP0 и TRAP1 могут быть выполнены при использовании команд RETS и RETI соответственно.

Другим выходом является включение следующего программного фрагмента, который защищает от изменения или сохранения статусный регистр путём запрещения прерываний через регистр разрешения прерывания:

```
PUSH    IE              ; Сохраняет регистр IE.
                        ; Команды включены для решения проблем с конвейером
LDI     0, IE           ; Очистка регистра IE
NOP                     ; Две NOP или другие команды
NOP                     ;
AND  0DFFFh, ST   ; Установить GIE=0.
                        ; Команда, которая считывает или записывает в регистр ST
POP  IE                 ; Команды включены для решения
                        ; проблем с конвейером
```

ПЦОС имеет два дополнительных исключения при обработке прерываний:

- первое исключение – разряд глобального разрешения прерываний регистра состояния (GIE) может быть ошибочно сброшен в 0 (запрещающая установка), если выполнены все следующие условия:

- выбрана команда условного системного прерывания (TRAPcond),

- условие TRAP не выполнено, и возник конфликт конвейера, что ведёт за собой задержку фазы декодирования или чтения команды.

В течение фазы декодирования условного программного прерывания, прерывания временно запрещены для гарантии того, что системное прерывание будет выполнено до последующего прерывания. Если возникает конфликт конвейера, вызывая задержку выполнения условного системного прерывания, состояние запрета прерывания может стать последним известным состоянием разряда GIE. В случае если условие системного прерывания не выполняется (false), то прерывания будут постоянно запрещены до тех пор, пока не будет непосредственно установлен разряд GIE. Условие не присутствует само по себе, когда условие системного прерывания выполняется (true), потому что при нормальном выполнении команды GIE сбрасывается, и стандартным подходом к программированию является установка GIE в единицу до окончания подпрограммы системного прерывания.

Вот пример нескольких последовательностей команд, которые могут привести к конфликту конвейера:

```
a)      LDI         mem, SP
        TRAPcond    n
b)      LDI         mem, SP
        NOP
        TRAPcond    n
c)      STI         SP, mem
        TRAPcond    n
d)      STI         Rx, *ARy
        LDI         *ARx, Ry
       || LDI       *ARz, Rw
        TRAPcond    n
```

Другие подобные условия также могут привести к задержке в выполнении.

Однако можно порекомендовать следующее решение для преодоления проблемы: ввести две команды NOP непосредственно перед командой TRAPcond. Одна команда NOP в некоторых случаях не помогает, как показано выше в примере (b).

Это предотвращает возможности возникновения многих конфликтов конвейера в непосредственно предшествующих командах и позволяет выполнять условный переход по системному программному прерыванию без задержек;

- второе исключение – асинхронные обращения к регистру флагов прерываний (IF) может привести к ошибке распознавания и обработки прерывания в ПЦОС. Это может произойти в случае, когда прерывания сгенерированы и записаны в регистр IF в том же цикле, когда ЦПУ изменяет содержимое IF.

Логика работы схемы прерываний такова, что ЦПУ имеет приоритет записи; таким образом, прерывание может быть потеряно. Это также верно в случае внутреннего прерывания, такого как прерывание ПДП. Эта ситуация может возникнуть как результат решения опросить определённые прерывания или желания очистить текущие (ожидающие обработки) прерывания, которые установлены из-за большой длины импульса на входе прерывания. Для случая большой длины импульса прерывание может быть выработано после того, как ЦПУ откликается на прерывание и пытается автоматически очистить его при обработке вектора прерывания.

В данном случае рекомендуется не использовать технику опроса прерывания, а строить внешние входы прерываний таким образом, чтобы ширина импульса на входе прерывания была между 1 – 2 длин командного цикла. Можно также при использовании техники опроса периодически запрещать и разрешать прерывания, которые будут опрашиваться, обеспечивая, таким образом, нормальную обработку прерываний; это автоматически очищает флаг прерываний без воздействия на другие прерывания. Если необходимо очистить прерывание, ожидающее обработку, рекомендуется использование ячейки памяти, служащей признаком того, что прерывание недействительно. Когда программа обработки прерывания сможет прочитать эту ячейку, необходимо очистить её (если ожидающее прерывание недействительно) и сделать возврат.

A.6.6.3 Приоритеты и управление

ЦПУ управляет расстановкой приоритетов прерываний (см. таблицу A.6.2, в которой даны адреса векторов сброса и прерываний и приоритеты). Если ПДП не использует прерывания для синхронизации пересылок, на него будет влиять обработка прерываний ЦПУ. Если ЦПУ вовлечён в конфликт конвейера (переход, регистр или память), реакции на прерывание не будет до тех пор, пока не разрешится конфликт.

Возможно, однако, прервать ЦПУ и ПДП одновременно с теми или другими прерываниями и, таким образом, синхронизировать их работу. Например, может быть, необходимо вызвать высокоприоритетную передачу ПДП, что предотвращает конфликты шин с ЦПУ, т. е. даёт ПДП более высокий приоритет, чем ЦПУ.

Это можно сделать при использовании прерывания, которое указывает ЦПУ на необходимость перехода к программе прерывания, которая содержит команду IDLE. Тогда, если то же самое прерывание используется для синхронизации пересылок ПДП, счётчик пересылок ПДП может быть использован для генерации прерывания и, таким образом, возвращать управление ЦПУ вслед за пересылкой ПДП.

Так как ЦПУ и ПДП используют один и тот же набор флагов прерываний, ПДП может очистить флаг прерывания до того, как ЦПУ сможет отозваться на него.

Например, если прерывания ЦПУ запрещены, ПДП может отработать прерывания и таким образом очистить соответствующие флаги прерываний.

Если в конвейере возникает задержанный переход, выполнение прерываний задерживается до окончания выполнения перехода. Если прерывание возникает на первом цикле выборки команды, выбранная команда не выполняется, и адрес данной команды записывается в вершину системного стека. Если прерывание возникает после первого цикла выборки, в случае многоцикловой команды, вызванной состояниями ожидания, эта команда выполняется и адрес следующей команды, которая должна будет выбрана, записывается в вершину системного стека. Если программная выборка отсутствует, новая выборка не происходит. После записи в стек адреса соответствующей команды, вектор прерывания выбирается и загружается в счётчик команд (PC), и выполнение продолжается.

Таблица А.6.2 – Адреса векторов сброса и прерываний

Сброс или прерывание вектора	Адрес	Приоритет	Функция
RESET#	0h	0	Вход внешнего сигнала сброса на вывод RESET#
INT0#	1h	1	Вход внешнего прерывания на вывод INT0#
INT1#	2h	2	Вход внешнего прерывания на вывод INT1#
INT2#	3h	3	Вход внешнего прерывания на вывод INT2#
INT3#	4h	4	Вход внешнего прерывания на вывод INT3#
XINT0	5h	5	Внутреннее прерывание; вырабатывается, когда буфер передачи последовательного порта 0 пуст
RINT0	6h	6	Внутреннее прерывание; вырабатывается, когда буфер приёма последовательного порта 0 заполнен
XINT1	7h	7	Внутреннее прерывание; вырабатывается, когда буфер передачи последовательного порта 1 пуст
RINT1	8h	8	Внутреннее прерывание; вырабатывается, когда буфер приёма последовательного порта 1 заполнен
TINT0	9h	9	Внутреннее прерывание; вырабатывается таймером 0
TINT1	0Ah	10	Внутреннее прерывание; вырабатывается таймером 1
DINT	0Bh	11	Внутреннее прерывание; вырабатывается контроллером ПДП

В ПЦОС ЦПУ и ПДП могут получать и обрабатывать прерывания параллельно. Прерывания опрошены, и ЦПУ и ПДП начинают их обработку. В течение прерываний, относящихся к ЦПУ, флаг прерывания, соответствующий разрешённому прерыванию с самым высоким приоритетом, очищается, и GIE устанавливается в ноль. ЦПУ завершает все выбранные команды. Вектор прерывания выбирается и загружается в PC, и ЦПУ продолжает выполнение. Цикл ПДП такой же, как для ЦПУ. После очистки соответствующего флага прерывания ПДП работает в соответствии с состоянием разряда SYNCH в регистре глобального управления ПДП.

A.7 Работа с внешней шиной

Память и внешние периферийные устройства могут быть доступны через два внешних интерфейса на ПЦОС: основной и расширенный. Генерация состояния ожидания, обеспечивающая доступ к медленной памяти и периферии, может управляться путём манипуляции с картированными в памяти регистрами управления, связанными с интерфейсами и внешними входными сигналами.

Основные положения, связанные с аппаратным интерфейсом и обсуждаемые в данном разделе, приведены ниже:

- регистры управления внешним интерфейсом (подраздел A.7.1):
    - основная шина (A.7.1.1);
    - расширенная шина (A.7.1.2);
- синхронизация внешнего интерфейса (подраздел A.7.2);
- программируемые состояния ожидания (подраздел A.7.3);
- программируемые групповые переключения (подраздел A.7.4).

A.7.1 Регистры управления внешним интерфейсом

ПЦОС поддерживает два внешних интерфейса: основной и расширенный. Основная шина состоит из 32-разрядной шины данных, 24-разрядной шины адреса и набора сигналов управления. Расширенная шина состоит из 32-разрядной шины данных, 13-разрядной шины адреса и набора сигналов управления. Обе шины поддерживают программно-управляемые состояния ожидания и внешний входной сигнал готовности и обе шины могут быть использованы для доступа данных, программ и ввода-вывода.

Доступ определяется по активному сигналу строба (STRB#, MSTRB#, IOSTRB#).

Когда обеспечивается доступ основной шины, сигнал STRB# находится в низком уровне. Расширенная шина в ПЦОС поддерживает два типа доступа:

- доступ к памяти по низкому уровню сигнала MSTRB#. Синхронизация доступа для MSTRB# та же, что и для доступа STRB# на основной шине;
- доступ к внешнему периферийному устройству по низкому уровню сигнала IOSTRB#.

Каждая из шин (основная и дополнительная) имеет связанный с ней регистр управления.

A.7.1.1 Регистр управления основной шиной

Регистр управления основной шиной PBCR (Primary-Bus Control Register; адрес – 80_8060h) – 32-разрядный регистр, который состоит из разрядов управления основной шины. Таблица A.7.1 содержит список разрядов регистра с их именами и функциями.

Таблица А.7.1 – Разряды регистра управления основной шиной PBCR

Разряд	Имя	Значение при сбросе	Функция
0	HOLDST	x (0 или 1)	Разряд состояния удержания. Этот сигнал указывает, будет ли порт удержан (HOLDST=1) или нет (HOLDST=0). Этот разряд состояния действителен при удержании порта аппаратно или программно.
1	NOHOLD	0	Сигнал удержания порта. NOHOLD разрешает или запрещает удержание порта внешним сигналом HOLD#. Когда NOHOLD=1, ПЦОС захватывает внешнюю шину и управляет ею, вне зависимости от запросов на продолжение или обслуживание от внешних устройств. Сигнал HOLDA# вырабатывается при получении сигнала HOLD#. Однако он выставляется при выработке внутреннего удержания (HIZ=1).
2	HIZ	0	Внутреннее удержание. Когда установлен (HIZ=1), порт устанавливается в состояние удержания. Это эквивалентно внешнему сигналу HOLD#. Устанавливая высокоимпедансное состояние, ПЦОС может «уступить» порт внешней памяти программным путём. HOLDA# устанавливается в низкий уровень при установке порта в высокоимпедансное состояние.
3-4	SWW	1 1	Программный режим ожидания. Совместно с WTCNT это двухразрядное поле определяет режим генерации состояния ожидания.
5-7	WTCNT	1 1 1	Программный режим ожидания. Это трёхразрядное поле определяет число циклов для дальнейшего использования в программировании режима ожидания для генерации внутренних состояний ожидания. Диапазон от нуля (WTCNT=0 0 0) до семи (WTCNT=1 1 1) циклов H1/H3.
8-12	BNKCMP	1 0 0 0 0	Сравнение банка. Это 5-разрядное поле используется для определения числа старших разрядов адреса (MSB), которые будут использованы для определения размера банка. Устанавливается в 1 0 0 0 0 при сбросе.
13-31	Резервные	0	Считываются как 0.

А.7.1.2 Регистр управления расширенной шиной

Регистр управления расширенной шиной EBCR (Expansion-Bus Control Register; адрес – 80_8060h) – 32-разрядный регистр, который состоит из разрядов управления для расширенной шины. Таблица А.7.2 содержит список разрядов регистра с их именами и функциями.

Таблица А.7.2 – Разряды регистра управления расширенной шиной EBCR

Разряд	Имя	Значение при сбросе	Функция
2-0	Зарезервированы	0 0 0	Считываются как 0.
4-3	SWW	1 1	Программный режим ожидания. Совместно с WTCNT это двухразрядное поле определяет режим генерации состояния ожидания. Устанавливается в 1 1 при сбросе.
7-5	WTCNT	1 1 1	Программный режим ожидания. Это трёхразрядное поле определяет число циклов для дальнейшего использования в программировании режима ожидания для генерации внутренних состояний ожидания. Диапазон от нуля (WTCNT=0 0 0) до семи (WTCNT=1 1 1) циклов H1/H3. Устанавливается в 1 1 1 при сбросе.
31-8	Зарезервированы	0-0	Считываются как 0.

А.7.2 Внешний интерфейс

Параллельные шины обеспечивают три взаимоисключающих адресных пространства через использование трёх отдельных сигналов управления: STRB#, MSTRB# и IOSTRB#. Сигнал STRB# управляет запросами на основной шине, а сигналы MSTRB# и IOSTRB# управляют запросами на расширенной шине, т. к. две шины независимы, два запроса могут идти параллельно.

За исключением переключения банка и внешней функции HOLD, временные диаграммы циклов основной шины и циклов расширенной шины MSTRB# идентичны и обсуждаются вместе. В случае идентичности диаграммы используется обозначение (M)STRB#. Аналогично обозначения (X)R/W#, (X)A, (X)D и (X)RDY# используются для обозначения эквивалентности сигналов основной и расширенной шины. Циклы IOSTRB# расширенной шины имеют другую диаграмму и обсуждаются отдельно.

А.7.2.1 Циклы ввода-вывода основной шины

Все циклы шины включают в себя целое число тактовых циклов H1. Один цикл H1 определён от одного падающего фронта H1 до следующего падающего фронта. Для полноскоростных запросов (без состояния ожидания) запись занимает два цикла H1 и чтение занимает один цикл; однако, если чтение следует за записью, чтение занимает два цикла. Это применимо как для основной шины, так и для доступа по MSTRB# к расширенной шине с другой стороны (с точки зрения ЦПУ и ПДП), запись требует только одного цикла при отсутствии текущих обращений по этому интерфейсу. Нижеследующие рассуждения относятся к обращениям с нулевым состоянием ожидания, кроме специально оговорённых случаев.

Сигнал (M)STRB# находится в низком уровне, как при чтении, так и при записи, которые продолжаются в течение одного цикла H1. Кроме того, до и после активной части – (M)STRB# низкий – только для записи присутствует переходный цикл H1.

В течение переходного цикла происходит следующее:

- (M)STRB# – высокий;
- если требуется, изменяется состояние (X)R/W# по возрастающему фронту H1;
- если требуется, по возрастающему фронту H1 изменяется адрес, если предыдущий цикл H1 был активной частью записи. Если предыдущий цикл H1 был чтением, адрес изменяется по падающему фронту H1.

Данные в цикле считываются как можно позднее для обеспечения максимального времени доступа по отношению к установке адреса. Необходимо обратить внимание, что хотя внешняя запись требует двух циклов внутри (с точки зрения ЦПУ и ПДП), они требуют только одного цикла при отсутствии текущих обращений по данному интерфейсу.

На типовых временных диаграммах для всех внешних интерфейсов строб (X)R/W# не изменяется, пока активны (M)STRB# или IOSTRB#.

Адрес и записываемые данные удерживаются приблизительно в течение половины цикла после изменения (M)STRB#.

Длительность (M)STRB#, (X)R/W# и (X)A также увеличивается на один цикл.

В последующем при выборе (X)RDY# он равен 0.

Пока изначально (X)RDY#=1, цикл записи увеличивается. (M)STRB#, (X)R/W# и (X)A увеличивается на один цикл. В последующем при выборе (X)RDY# он равен 0.

A.7.2.2 Циклы ввода-вывода расширенной шины

В противоположность основной шине и циклам (M)STRB#, IOSTRB# осуществляет чтение и запись в течение двух циклов (без состояния ожидания) при схожей временной диаграмме. В течение этих циклов адрес всегда изменяется по заднему фронту H1, и IOSTRB# находится в низком уровне от переднего фронта первого цикла H1 до переднего фронта второго цикла H1. Сигнал IOSTRB# всегда неактивен (в высоком уровне) между циклами, и XR/W# в высоком уровне – для чтения и в низком – для записи.

Для обращений IOSTRB# циклы чтения и записи требуют, как минимум два цикла. Некоторые внешние периферийные устройства могут менять свои разряды состояния при чтении или записи. Важно, однако, чтобы адрес удерживался при связи с данной периферией. Для чтения и записи при активном IOSTRB#, он должен полностью охватываться адресом.

Для каждого дополнительного состояния ожидания IOSTRB#, XR/W# и XA увеличиваются на один цикл. Выбор XRDY# повторяется каждый цикл.

Стробы (STRB#, MSTRB#, IOSTRB#) переходят в 1. Адреса не определены и сигналы готовности (XRDY# или RDY#) игнорируются.

HOLD# – внешний асинхронный вход. Задержка по времени между установкой HOLD#=0 и HOLDA#=0 минимум 1 цикл. Если HOLDA#=0, то адрес, шина данных и связанные с ними стробы устанавливаются в высокоимпедансное состояние. Все обращения через интерфейс завершаются до подтверждения удержания.

А.7.3 Программируемые состояния ожидания

Управление генерацией состояния ожидания через регистры управления, картированные в памяти, связано как с основным, так и с расширенным интерфейсом.

Используется поле WTCNT для загрузки внутреннего таймера и поле SWW для выбора одного из четырёх следующих режимов генерации состояния ожидания:

- внешний RDY#;
- полученный из WTCNT RDY# wtcnt;
- логическое AND (И) над RDY# и RDY# wtcnt;
- логическое OR (ИЛИ) над RDY# и RDY# wtcnt.

Четыре поля используются для выработки внутреннего сигнала готовности, RDY# int, который управляет доступом (обращениями). Пока RDY# int=1, текущий внешний запрос задерживается. Когда RDY# int=0, выполняется текущий запрос.

Т. к. использование программируемых состояний ожидания для обоих внешних интерфейсов идентично, в следующих разделах обсуждается только интерфейс основной шины.

RDY# wtcnt – 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.

Пока счётчик не 0, RDY# wtcnt=1. Пока счётчик равен 0, RDY# wtcnt=0.

Когда SWW=0 0, RDY# int зависит только от RDY#. RDY# wtcnt игнорируется.

Таблица истинности для данного режима приведена в таблице А.7.3.

Таблица А.7.3 – Генерация состояния ожидания при SWW=0 0

RDY#	RDY# wtcnt	RDY# int
0	0	0
0	1	0
1	0	1
1	1	1

Когда SWW=0 1, RDY# int только от RDY# wtcnt, RDY# игнорируется.

Таблица истинности для данного режима приведена в таблице А.7.4.

Таблица А.7.4 – Генерация состояния ожидания при SWW=0 1

RDY#	RDY# wtcnt	RDY# int
0	0	0
0	1	1
1	0	0
1	1	1

Когда SWW=1 0, RDY# int определяется как логическое ИЛИ (электрически И, т. к. эти сигналы имеют активный низкий уровень) от RDY# и RDY# wtcnt (см. таблицу А.7.5).

Когда SWW=1 1, RDY# int определяется как логическое И (электрически ИЛИ, т. к. эти сигналы имеют активный низкий уровень) от RDY# и RDY# wtcnt (см. таблицу А.7.6).

Таблица А.7.5 – Генерация состояния ожидания при SWW=1 0

RDY#	RDY# wtcnt	RDY# int
0	0	0
0	1	0
1	0	0
1	1	1

Таблица А.7.6 – Генерация состояния ожидания при SWW=1 1

RDY#	RDY# wtcnt	RDY# int
0	0	0
0	1	1
1	0	1
1	1	1

А.7.4 Программируемое переключение банков

Программируемое переключение банков позволяет производить переключение между банками внешней памяти без вводимых извне состояний ожидания благодаря памяти, которая требует нескольких циклов для выключения. Переключение банков поддерживается только для основной шины.

Размер банка определяется числом разрядов, которые будут опрошены. Например, если BNKCMP=16, то для определения банка используется 16 старших разрядов адреса. Т. к. адреса 24-разрядные, размер банка определяется 8 младшими разрядами, определяя размер банка в 256 слов. Если BNKCMP≥16, компарируются только 16 старших разрядов. Поддерживаются размеры банков от 2^8=256 до 2^{24}=16M. В таблице А.7.7 приведено соответствие между BNKCMP, разрядами адреса, использованными для определения банка и результирующим размером банка.

Таблица А.7.7 – Размер банка в зависимости от значения поля BNKCMP

BNKCMP	Разряды	Размер (32-битное слово)
00000	нет	2^{24}=16M
00001	23	2^{23}=8M
00010	23-22	2^{22}=4M
00011	23-21	2^{21}=2M
00100	23-20	2^{20}=1M
00101	23-19	2^{19}=512K
00110	23-18	2^{18}=256K
00111	23-17	2^{17}=128K
01000	23-16	2^{16}=64K
01001	23-15	2^{15}=32K
01010	23-14	2^{14}=16K
01011	23-13	2^{13}=8K
01100	23-12	2^{12}=4K
01101	23-11	2^{11}=2K
01110	23-10	2^{10}=1K
01111	23-9	2^9=512
10000	23-8	2^8=128
10000-11111	Зарезервированы	Не определены

ПЦОС имеет внутренний регистр, который содержит старшие разряды (как определено в поле BNKCMP) последнего адреса, используемого для чтения или записи через основной интерфейс. При сбросе регистр устанавливается в ноль.

Если старшие разряды адреса, который будет использован для текущего чтения через основной интерфейс, не совпадают с содержащимися во внутреннем регистре, чтение не выполняется в один цикл H1/H3. В течение дополнительного цикла синхронизации шина адреса переключается на новый адрес, но строб STRB# не активен (высокий). Содержимое внутреннего регистра заменяется значениями старших разрядов адреса, по данному адресу будет выполняться текущее чтение. Если старшие разряды адреса, который использован для текущего чтения, совпадают с содержащимися во внутреннем регистре, имеет нормальный цикл чтения.

Если осуществляются повторные чтения из одного банка памяти, дополнительные циклы чтения не требуются. Если осуществляются чтения из разных банков памяти, конфликт памяти разрешается введением дополнительных циклов.

Эта возможность может быть запрещена при установке BNKCMP в 0. Введение дополнительных циклов осуществляется только при чтении. Изменения старших разрядов во внутреннем регистре происходят для всех циклов чтения и записи через основной интерфейс.

А.8 Периферийные устройства

Процессорное ядро содержит два таймера, два последовательных порта и встроенный контроллер прямого доступа к памяти (ПДП). Эти периферийные модули управляются через регистры, картированные в памяти, указывающие на соответствующую периферийную шину. Контроллер ПДП используется для обеспечения операций ввода-вывода без обращения к ЦПУ. Таким образом, это даёт возможность обращения к медленной внешней памяти и периферии (АЦП, последовательным портам и т. д.) без соответствующего обращения к ЦПУ.

Основные характеристики периферийных устройств, обсуждаемые в данном разделе, приведены ниже:

- модули таймера (подраздел А.8.1);
- регистры;
- генерация импульсов;
- режимы работы;
- последовательные порты (подраздел А.8.2);
- регистры;
- рабочие конфигурации;
- временные диаграммы;
- примеры;
- контроллер ПДП (подраздел А.8.3);
- регистры;
- операция пересылки памяти при ПДП;
- синхронизация каналов ПДП.

А.8.1 Модули таймера

Модули таймера ПЦОС – 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 сигналов ПЦОС (или внешним устройствам) через определённые интервалы или считывания внешних событий. С внутренним тактированием таймер может быть использован в качестве указателя внешнему ЦАП начать преобразование или прервать контроллер ПДП ПЦОС для начала пересылки данных. Прерывание таймера является одним из внутренних прерываний. С внешней синхронизацией таймер может считать внешние события и прерывать ЦПУ после определённого числа событий. Доступный для каждого таймера вывод входа/выхода может быть использован как тактовый вход таймера, как выходной сигнал синхронизации или вывод входа/выхода общего назначения.

Каждый таймер имеет три регистра, картированных в памяти:

- регистр глобального управления (Global-control register);
- регистр периода (Period register);
- регистр счётчика (Counter register).

Регистр глобального управления определяет режим работы таймера, отслеживает состояние таймера и управляет выводом входа/выхода таймера.

Регистр периода определяет частоту выдачи сигналов таймером.

Регистр счётчика содержит текущее значение инкрементируемого счётчика. Таймер может инкрементироваться по переднему или заднему фронту входного сигнала синхронизации. Счётчик устанавливается в 0 и может вызвать внутреннее прерывание, когда его значение равно значению, содержащемуся в регистре периода. Генератор импульсов генерирует два типа сигналов: импульс или такт.

A.8.1.1 Регистр глобального управления таймером

Регистр глобального управления таймером – 32-разрядный регистр, содержащий разряды глобального управления и управления портом для модуля таймера. Таблица A.8.1 определяет 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).

Таблица A.8.1 – Регистр глобального управления таймером

Разряды	Имя	Значение при сбросе	Функция
1	2	3	4
0	FUNC	0	FUNC управляет работой TCLK: если FUNC=0, TCLK конфигурируется как цифровой порт ввода-вывода общего назначения; если FUNC=1, TCLK конфигурируется как вывод таймера
1	I/O	0	Если FUNC=0 и CLKSRC=0, TCLK конфигурируется как вывод входа/выхода общего назначения. В этом случае, если I/O=0, TCLK конфигурируется как входной вывод общего назначения. Если I/O=1, TCLK конфигурируется как выходной вывод общего назначения.
2	DATOUT	0	Управляет TCLK, когда ПЦОС находится в режиме порта ввода-вывода. Может быть также использован как ввод в таймер.
3	DATIN	Уровень на TCLK	Вход данных на TCLK или DATOUT. При записи состояние не изменяется.
4-5	Зарезер-вированы	0-0	Считываются как 0.
6	GO	0	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.

Окончание таблицы А.8.1

1	2	3	4
7	HLD	0	Сигнал удержания счётчика. Когда этот разряд ноль, 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 счётчик может продолжиться с состояния останова при установке HLD в 1. Регистры таймера могут считываться и модифицироваться, пока счётчик удерживается. RESET# имеет более высокий приоритет, чем HLD. В таблице А.8.2 приводится результат записи в GO и HLD.
8	C/P	0	Управление режимом такт/импульс. Когда C/P=1, выбирается режим такт, что указывает на то, 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.
9	CLKSRC	0	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. Это будет меньше, чем f(H1)/2.
10	INV	0	Разряд управления инверсией. Если использован внешний источник тактирования и INV=1, то внешний такт инвертируется при входе в счётчик. Если выход генератора импульсов подключён 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.
11	TSTAT	0	Указывает на состояние таймера. Он отслеживает выход неинвертированного вывода TCLK. Этот флаг выставляет прерывание ЦПУ при переходе из 0 в 1. Запись не влияет.
12-31	Зарезер-вированы	0-0	Считываются как 0.

В таблице А.8.2 приводится результат записи при использовании определённых значений разрядов GO и HLD в регистр глобального управления.

Таблица А.8.2 – Результат записи определённых значений 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.

А.8.1.2 Регистры периода таймера и счётчика

32-разрядный регистр периода таймера используется для определения частоты выдачи сигнала таймера. Регистр счётчика таймера – 32-разрядный регистр, который сбрасывается в ноль каждый раз по достижении значения, находящегося в регистре периода. Оба регистра устанавливаются в ноль при сбросе.

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=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.

Записи с периферийной шины «передавливают» изменения регистра счётчика, и новое состояние изменяет значение регистра управления.

А.8.1.3 Генерация импульсов таймера

Генератор импульсов таймера может генерировать несколько различных внешних сигналов. Эти сигналы могут быть инвертированы разрядом INV. 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).

Период выдачи сигнала таймера определяется частотой входа такта таймера и регистром периода. Следующие выражения определены либо для внутреннего, либо для внешнего тактирования таймера:

f(импульсный режим) = f(такта таймера) / период регистра

f(тактовый режим) = f(такта таймера) / (2 × период регистра)

Если регистр периода равен 0, см. А.8.1.2.

А.8.1.4 Режимы работы таймера

Таймер может передавать с выхода и принимать на вход в нескольких различных режимах, зависящих от установок CLKSRC, FUNC и I/O.

Режимы работы таймера описаны ниже.

Если CLKSRC=1 и FUNC=0, вход таймера поступает с внутреннего такта. Внутреннее тактирование не зависит от разряда INV. В этом режиме TCLK подключён к управлению ввода-вывода порта и может быть использован как вывод ввода-вывода общего назначения. Если I/O=0, TCLK конфигурируется как вход общего назначения, состояние которого может быть считано в DATIN. При этом DATOUT не влияет ни на TCLK, ни на DATIN. Если I/O=1, TCLK конфигурируется как выход общего назначения. DATOUT помещается на TCLK и может быть считано в DATIN.

Если CLKSRC=1 и FUNC=1, вход таймера поступает с внутреннего такта, выход таймера при этом идёт на TCLK. Это значение может быть проинвертировано, используя INV, и значение выхода на TCLK может быть считано в DATIN.

Если CLKSRC=0 и FUNC=0, таймер управляется в зависимости от состояния разряда I/O. Если I/O =0, вход таймера идёт от TCLK. Это значение может быть проинвертировано, используя INV, и значение выхода на TCLK может быть считано в DATIN. Если I/O =1, то TCLK – выходной вывод. При этом и TCLK и таймер управляются DATOUT. Все переходы из 0 в 1 на DATOUT инкрементируют счётчик. Значение DATOUT может быть считано в DATIN.

Если CLKSRC=0 и FUNC=1, таймер управляется TCLK. Если INV=0, счётчик инкрементируется при всех переходах из 0 в 1 на TCLK. Если INV=1, счётчик инкрементируется при всех переходах из 1 в 0 на TCLK. Значение TCLK может быть считано в DATIN.

А.8.1.5 Прерывания таймера

Прерывание таймера вырабатывается при автоматическом сбросе регистра счётчика таймера в 0. Регистр счётчика таймера автоматически сбрасывается в 0 каждый раз, когда его значение больше или равно значению в регистре периода таймера. Прерывание таймера может быть использовано для прерывания либо ЦПУ, либо ПДП.

Управление разрешением прерывания существует для каждого таймера либо для ЦПУ, либо для ПДП, как указано в регистре разрешения прерываний ЦПУ/ПДП (см. А 3.1.8). При выработке прерывания таймера проверяется состояние соответствующего вывода TCLK (при FUNC=1 и CLKSRC=1) в регистре глобального управления таймера. Точное изменение состояния зависит от состояния разряда C/P.

А.8.1.6 Инициализация/реконфигурация таймера

Таймеры управляются через картированные в памяти регистры, адресуемые по соответствующей периферийной шине. Общая процедура для инициализации и/или реконфигурации таймеров:

- 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#;
- 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 регистре глобального управления таймера.

А.8.2 Последовательные порты

ПЦОС имеет два полностью независимых двунаправленных последовательных порта. Оба порта одинаковы и имеют дополнительно набор регистров управления.

Каждый последовательный порт может быть настроен на передачу 8-, 16-, 24- или 32-разрядных слов данных одновременно в обоих направлениях. Тактирование для каждого последовательного порта может быть как внутренним, посредством таймера регистра последовательного порта и регистра периода, так и внешним, посредством подключения внешнего источника синхронизации. Внутренняя синхронизация представляет собой разделённую частоту выходной синхронизации, f(H1). Поддерживается также непрерывный режим пересылки, при котором последовательный порт может передавать и принимать любое количество слов без нового импульса синхронизации.

Каждый последовательный порт имеет восемь регистров, картированных в памяти:

- регистр глобального управления;
- два регистра управления для 6 последовательных выводов входа/выхода;
- три регистра таймера приёма/передачи;
- регистр передачи данных;
- регистр приёма данных.

Регистр глобального управления контролирует общее функционирование последовательного порта и определяет режим работы последовательного порта. Два регистра управления порта контролируют функционирование 6 выводов последовательного порта. Буфер передачи содержит следующее полное передаваемое слово. Буфер приёма содержит последнее полное принятое слово. Три дополнительных регистра связаны с секцией приёма/передачи таймера последовательного порта.

Регистр глобального управления последовательного порта – 32-разрядный регистр, содержащий разряды общего управления последовательного порта.

В таблице А.8.3 определены разряды регистра, названия разрядов и их функции.

Таблица А.8.3 – Описание разрядов регистра глобального управления последовательного порта

Разряд	Имя	Значение при сбросе	Функция
1	2	3	4
0	RRDY	0	Если RRDY=1, в буфере приёма имеются новые данные, готовые к считыванию. От считывания DRR до RRDY=1 проходит 3 цикла H1/H3. Возрастающий фронт этого сигнала выставляет RINT. Если RRDY=0 при сбросе, то приёмный буфер не имеет новых данных с последнего чтения. RRDY=0 при сбросе и после считывания буфера приёма
1	XRDY#	0	Если XRDY#=1, последние данные записаны из буфера передачи в сдвигатель, и буфер готов к чтению нового слова. От загрузки сдвигового регистра передачи до установки XRDY#=1 проходит 3 цикла H1/H3
2	FSXOUT	0	Устанавливает вывод FSX как вход (FSXOUT=0) или как выход (FSXOUT=1)

Продолжение таблицы А.8.3

1	2	3	4
3	XSREMPTY	0	Если XSREMPTY=0, то передающий сдвиговый регистр пуст. Если XSREMPTY=1, то передающий сдвиговый регистр не пуст. Сброс или XRESET устанавливает данный разряд в 0
4	RSRFULL	0	Если RSRFULL=1, наблюдается переполнение приёмника. В непрерывном режиме RSRFULL устанавливается в 1, когда и RSR и DRR полны. В пакетном (непрерывном) режиме RSRFULL устанавливается в 1, когда и RSR и DRR полны, и приходит новый FSR. При чтении данный разряд устанавливается в 0. Этот разряд может быть установлен в 0 только при системном сбросе, сбросе приёма последовательного порта (RRESET=1) или при чтении. Когда приёмник пытается установить RSRFULL в 1 в то же самое время, когда происходит чтение глобального регистра, приёмник обладает большим приоритетом, и RSRFULL устанавливается в 1. Если RSRFULL=0, переполнения приёмника не происходит
5	HS	0	Если HS=1, то режим квитирования передачи разрешён. Если HS=0, то режим квитирования передачи запрещён
6	XCLKSRCE	0	Если XCLKSRCE=1, используется внутренняя синхронизация передачи. Если XCLKSRCE=0, используется внешняя синхронизация передачи
7	RCLKSRCE	0	Если RCLKSRCE=1, используется внутренняя синхронизация приёма. Если RCLKSRCE=0, используется внешняя синхронизация приёма
8	XVAREN	0	Этот разряд определяет фиксированную (XVAREN=0) или переменную (XVAREN=1) выдачу указателя периода данных при передаче. При фиксированном периоде данных FSX активен для последнего цикла XCLK и становится неактивным до начала передачи. При переменном периоде данных FSX активен в течение передачи всех разрядов. При использовании внешнего FSX и переменного указателя периода данных вывод DX управляется передатчиком, когда FSX удерживается активным, или пока слово не будет сдвинуто наружу
9	RVAREN	0	Этот разряд определяет фиксированную (RVAREN=0) или переменную (RVAREN=1) выдачу указателя периода данных при приёме. При фиксированном периоде данных FSR активен для последнего цикла RCLK и становится неактивным до начала приёма. При переменном периоде данных FSR активен в течение всего приёма всех разрядов
10	XFSM	0	Режим кадровой синхронизации передачи. Устанавливает порт в режим непрерывной работы (XFSM=1) или стандартный режим (XFSM=0). В непрерывном режиме только первое слово блока вырабатывает импульс синхронизации, а остаток просто непрерывно передаётся до окончания блока. В стандартном режиме с каждым словом связан импульс синхронизации

Продолжение таблицы А.8.3

1	2	3	4
11	RFSM	0	Режим кадровой синхронизации приёма. Устанавливает порт в режим непрерывной работы (RFSM=1) или стандартный режим (RFSM=0). В непрерывном режиме только первое слово блока вырабатывает импульс синхронизации, а остаток просто непрерывно принимается без ожидания другого синхроимпульса. В стандартном режиме с каждым словом связан импульс синхронизации
12	CLKXP	0	Полярность CLKX. Если CLKXP=0, то CLKX активен высоким уровнем. Если CLKXP=1, то CLKX активен низким уровнем
13	CLKRP	0	Полярность CLKR. Если CLKRP=0, то CLKR активен высоким уровнем. Если CLKRP=1, то CLKR активен низким уровнем
14	DXP	0	Полярность DX. Если DXP=0, то DX активен высоким уровнем. Если DXP=1, то DX активен низким уровнем
15	DRP	0	Полярность DR. Если DRP=0, то DR активен высоким уровнем. Если DRP=1, то DR активен низким уровнем
16	FSXP	0	Полярность FSX. Если FSXP=0, то FSX активен высоким уровнем. Если FSXP=1, то FSX активен низким уровнем
17	FSRP	0	Полярность FSR. Если FSRP=0, то FSR активен высоким уровнем. Если FSRP=1, то FSR активен низким уровнем
18-19	XLEN	0-0	Эти два разряда определяют длину слова передачи последовательного порта. Предполагается, что все данные (right-justified) установлены по правой границе буфера передачи, когда определены менее 32 разрядов
20-21	RLEN	0-0	Эти два разряда определяют длину слова приёма последовательного порта. Все данные (right-justified) выровнены по правой границе буфера приёма: 0-0 … 8 разрядов; 1-0 … 24 разряда; 0-1 …16 разрядов; 1-1 … 32 разряда
22	XTINT	0	Разрешение прерывания таймера передачи: если XTINT=0, прерывание таймера передачи запрещено; если XTINT=1, прерывание таймера передачи разрешено
23	XINT	0	Разрешение прерывания передачи: если XINT=0, прерывание передачи запрещено; если XINT=1, прерывание передачи разрешено
24	RTINT	0	Разрешение прерывания таймера приёма: если RTINT=0, прерывание таймера приёма запрещено; если RTINT=1, прерывание таймера приёма разрешено.
25	RINT	0	Разрешение прерывания приёма: если RINT=0, прерывание приёма запрещено; если RINT=1, прерывание приёма разрешено
26	XRESET	0	Сброс передатчика. Если XRESET=0, передающая часть последовательного порта сбрасывается. Для выхода из сброса передающей части последовательного порта необходимо установить XRESET в 1. Однако нельзя устанавливать XRESET в 1 до истечения минимум 3 циклов после установки XRESET в неактивное состояние. Это применимо только к системному сбросу. Установка XRESET в 0 не изменяет состояния регистров управления порта. Он устанавливает передатчик в состояние, соответствующее началу кадра данных. Сброс передатчика приводит к установке прерывания передачи. XFSM может быть записан без сброса регистра глобального управления

Окончание таблицы А.8.3

1	2	3	4
27	RRESET	0	Сброс приёмника. Если RRESET=0, приёмная часть последовательного порта сбрасывается. Для выхода из сброса приёмной части последовательного порта необходимо установить RRESET в 1. Установка RRESET в 0 не изменяет состояния регистров управления порта. Он устанавливает приёмник в состояние, соответствующее началу кадра данных. RFSM может быть записан без сброса регистра глобального управления
28-31	Зарезервированы	0-0	Считываются как 0

А.8.2.1 Регистр управления выводами FSX, DX, CLKX последовательного порта

Этот 32-разрядный регистр управляет работой выводов FSX, DX, CLKX последовательного порта. При сбросе все разряды данного регистра устанавливаются в 0.

В таблице А.8.4 определены разряды регистра, названия разрядов и их функции.

Таблица А.8.4 – Разряды регистра управления выводами FSX, DX, CLKX последовательного порта

Разряд	Наименование	Значение при сбросе	Функция
0	CLKXFUNC	0	CLKXFUNC управляет работой CLKX. Если CLKXFUNC=0, CLKX установлен как цифровой порт ввода-вывода общего назначения. Если CLKXFUNC=1, то CLKX – вывод последовательного порта
1	CLKXI/O	0	Если CLKXI/O=0, CLKX установлен как вход общего назначения. Если CLKXI/O=1, то CLKX – выход общего назначения
2	CLKXDATOUT	0	Выход данных для CLKX.
3	CLKXDATIN	x *	Вход данных для CLKX. Запись не влияет.
4	DXFUNC	0	DXFUNC управляет работой DX. Если DXFUNC=0, DX установлен как цифровой порт ввода-вывода общего назначения. Если DXFUNC=1, то DX – вывод последовательного порта
5	DXI/O	0	Если DXI/O=0, DX установлен как вход общего назначения. Если DXI/O=1, то DX – выход общего назначения.
6	DXDATOUT	0	Выход данных для DX
7	DXDATIN	x *	Вход данных для DX. Запись не влияет
8	FSXFUNC	0	FSXFUNC управляет работой FSX. Если FSXFUNC=0, FSX установлен как цифровой порт ввода-вывода общего назначения. Если FSXFUNC=1, то FSX – вывод последовательного порта
9	FSXI/O	0	Если FSXI/O=0, FSX установлен как вход общего назначения. Если FSXI/O=1, то FSX – выход общего назначения
10	FSXDATOUT	0	Выход данных для FSX
11	FSXDATIN	x *	Вход данных на FSX. Запись не влияет
12-31	Зарезервированы	0-0	Считываются как 0
* x = 0 или 1.			

A.8.2.2 Регистр управления выводами FSR, DR, CLKR последовательного порта

Этот 32-разрядный регистр управляет работой выводов FSR, DR, CLKR последовательного порта. При сбросе все разряды данного регистра устанавливаются в 0.

В таблице А.8.5 определены разряды регистра, названия разрядов и их функции.

Таблица А.8.5 – Разряды регистра управления выводами FSR, DR, CLKR последовательного порта

Разряд	Наименование	Значение при сбросе	Функция
0	CLKRFUNC	0	CLKRFUNC управляет работой CLKR. Если CLKRFUNC=0, CLKR установлен как цифровой порт ввода-вывода общего назначения. Если CLKRFUNC=1, то CLKR – вывод последовательного порта
1	CLKRI/O	0	Если CLKRI/O=0, CLKR установлен как вход общего назначения. Если CLKRI/O=1, то CLKR – выход общего назначения
2	CLKRDATOUT	0	Выход данных для CLKR
3	CLKRDATIN	x *	Вход данных для CLKR. Запись не влияет
4	DRFUNC	0	DRFUNC управляет работой DR. Если DRFUNC=0, DR установлен как цифровой порт ввода-вывода общего назначения. Если DRFUNC=1, то DR – вывод последовательного порта
5	DRI/O	0	Если DRI/O=0, DR установлен как вход общего назначения. Если DRI/O=1, то DR – выход общего назначения
6	DRDATOUT	0	Выход данных для DR
7	DRDATIN	x*	Вход данных для DR. Запись не влияет
8	FSRFUNC	0	FSRFUNC управляет работой FSR. Если FSRFUNC=0, FSR установлен как цифровой порт ввода-вывода общего назначения. Если FSRFUNC=1, то FSR – вывод последовательного порта
9	FSRI/O	0	Если FSRI/O=0, FSR установлен как вход общего назначения. Если FSRI/O=1, то FSR – выход общего назначения
10	FSRDATOUT	0	Выход данных для FSR
11	FSRDATIN	x *	Вход данных на FSR. Запись не влияет
12-31	Зарезервированы	00	Считываются как 0
* x = 0 или 1.			

A.8.2.3 Регистр управления таймера приёма/передачи

32-разрядный регистр управления таймером приёма/передачи содержит разряды управления для модуля таймера. При сбросе все разряды устанавливаются в 0. В таблице А.8.6 перечислены разряды регистра, их наименования и функции. Разряды 5-0 управляют таймером передатчика. Разряды 11-6 управляют таймером приёмника. См. подраздел А.8.1 «Модули таймера» для более полной информации о таймерах.

Таблица А.8.6 – Регистр управления таймером приёма/передачи

Разряд	Имя	Значение при сбросе	Функция
1	2	3	4
0	XGO	0	Разряд XGO сбрасывает и запускает счётчик таймера. Когда XGO=1 и таймер не удерживается, счётчик обнуляется и начинает инкрементироваться по следующему возрастающему фронту тактового входа таймера. XGO очищается по тому же возрастающему фронту. XGO=0 не влияет на таймер.
1	XHLD	0	Сигнал удержания счётчика передачи. Когда этот разряд ноль, счётчик запрещён и удерживается в его текущем состоянии. Внутренний счётчик с делением на два тоже удерживается таким образом, что счётчик может продолжиться с состояния останова при установке XHLD в 1. Регистры таймера могут считываться и модифицироваться, пока счётчик удерживается. RESET# имеет более высокий приоритет, чем XHLD.
2	XC/P	0	Управление режимом такт/импульс. Когда XC/P=1, выбирается режим такт, что указывает на то, что флаг состояния и внешний выход будут иметь цикл со скважностью 50 %. Когда XC/P=0, флаг состояния и внешний выход будут активными для одного цикла CLKOUT в течение каждого периода таймера.
3	XCLKSRC	0	Определяет источник синхронизации таймера передачи. Когда XCLKSRC=1, для инкрементирования счётчика будет использоваться внутренний такт с частотой в половину частоты CLKOUT. Когда XCLKSRC=0, для инкрементирования счётчика может использоваться внешний сигнал с вывода CLKX. Внешний такт синхронизирован изнутри, обеспечивая, таким образом, возможность работы с внешними асинхронными источниками тактирования, которые не превышают максимально поддерживаемой частоты внешнего такта. Это будет меньше, чем f(H1)/2,6.
4	Зарезервирован	0	Считывается как 0.
5	XTSTAT	0	Указывает состояние таймера передачи. Он следит за тем, что будет на выходе неинвертированного вывода CLKX. Данный флаг выставляет прерывание ЦПУ при переходе из 0 в 1. Запись не влияет.
6	RGO	0	Разряд RGO сбрасывает и запускает счётчик таймера. Когда RGO=1 и таймер не удерживается, счётчик обнуляется и начинает инкрементироваться по следующему возрастающему фронту тактового входа таймера. RGO очищается по тому же возрастающему фронту. RGO=0 не влияет на таймер.
7	RHLD	0	Сигнал удержания счётчика приёма. Когда этот разряд ноль, счётчик запрещён и удерживается в его текущем состоянии. Внутренний счётчик с делением на два тоже удерживается таким образом, что счётчик может продолжиться с состояния останова при установке RHLD в 1. Регистры таймера могут считываться и модифицироваться, пока счётчик удерживается. RESET# имеет более высокий приоритет, чем RHLD.

Окончание таблицы А.8.6

1	2	3	4
8	RC/P	0	Управление режимом R такт/импульс. Когда RC/P=1, выбирается режим такт, что указывает на то, что флаг состояния и внешний выход будут иметь цикл со скважностью 50 %. Когда RC/P=0, флаг состояния и внешний выход будут активными для одного цикла CLKOUT в течение каждого периода таймера.
9	RCLKSRC	0	Определяет источник синхронизации таймера приёма. Когда RCLKSRC=1, для инкрементирования счётчика будет использоваться внутренний такт с частотой в половину частоты CLKOUT. Когда RCLKSRC=0, для инкрементирования счётчика может использоваться внешний сигнал с вывода CLKR. Внешний такт синхронизирован изнутри, обеспечивая, таким образом, возможность работы с внешними асинхронными источниками тактирования, которые не превышают максимально поддерживаемой частоты внешнего такта. Это будет меньше, чем f(H1)/2,6.
10	Зарезервирован	0	Считывается как 0.
11	RTSTAT	0	Указывает состояние таймера приёма. Он следит за тем, что будет на выходе неинвертированного вывода CLKR. Данный флаг выставляет прерывание ЦПУ при переходе из 0 в 1. Запись не влияет.
12-31	Зарезервированы	0-0	Считываются как 0.

А.8.2.4 Регистр счётчика таймера приёма/передачи

Регистр счётчика таймера приёма/передачи представляет собой 32-разрядный регистр. Разряды 15-0 относятся к счётчику таймера передачи, разряды 31-16 относятся к счётчику таймера приёма. Каждый счётчик устанавливается в 0 при достижении значения, хранящегося в регистре периода, см. А.8.2.5. Он также устанавливается в 0 при сбросе.

Рисунок А.8.1 – Регистр счётчика таймера приёма/передачи

А.8.2.5 Регистр периода таймера приёма/передачи

Регистр периода таймера приёма/передачи – 32-разрядный регистр.

Разряды 15-0 относятся к периоду таймера передачи, разряды 31-16 относятся к периоду таймера приёма. Каждый счётчик используется для определения периода таймера.

Он также устанавливается в 0 при сбросе.

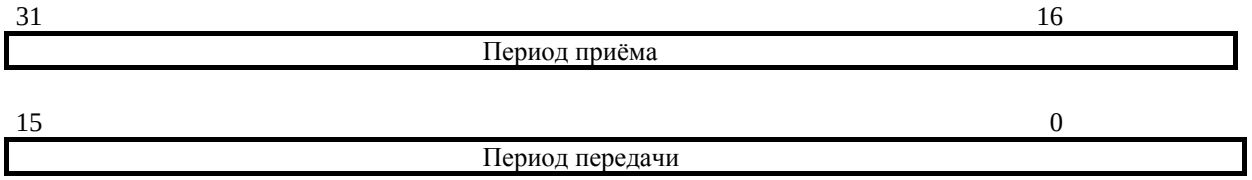


Рисунок А.8.2 – Регистр периода таймера приёма/передачи

А.8.2.6 Регистр передачи данных

Когда регистр передачи данных (DXR) загружен, передатчик загружает слово в передающий регистр сдвига (XSR) и поразрядно выдвигает его наружу. Задержка от момента записи в DXR до выставления FSX (который может быть также установлен извне) составляет два цикла CLKX. Слово не загружается в регистр сдвига, пока не очистится сдвигатель. Когда DXR загружается в XSR, выставляется разряд XRDY, указывающий на то, что буфер готов к приёму следующего слова. Для передачи слова в сдвиговом передающем регистре предусмотрены 4 точки отвода, соответствующие 4 размерам передаваемого слова. Сдвиг происходит влево (от младших разрядов (LSB) к старшим (MSB)) с выдачей данных от старшего разряда в соответствующей точке отвода.

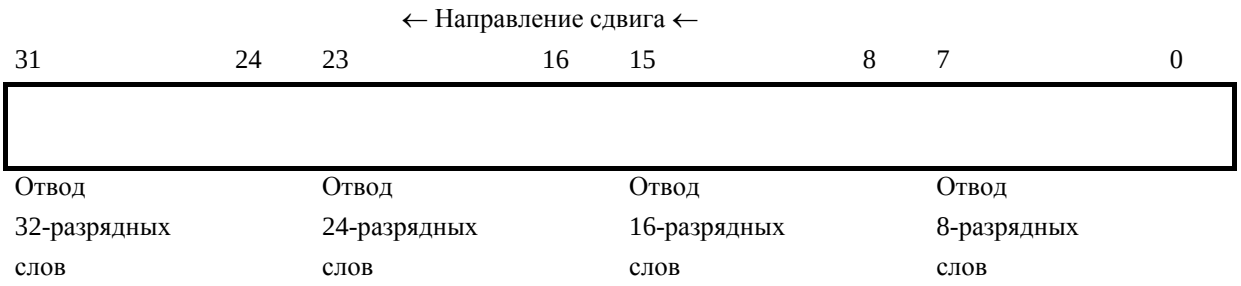


Рисунок А.8.3 – Работа сдвигового регистра передачи

Когда приходят последовательные данные, приёмник сдвигает их в приёмный регистр сдвига (RSR). Когда определённое число разрядов сдвинуто внутрь, содержимое RSR записывается в регистр приёма данных (DRR) и выставляется разряд состояния RRDY. Приёмник обладает двойной буферизацией. Если DRR не считан, и RSR полон, приём останавливается. Новые данные, приходящие на вход DX, игнорируются. Приёмный сдвигатель не будет записываться через DRR. DRR должен быть считан для того, чтобы новые данные из RSR записались в DRR. Если запись в DRR имеет место в то же самое время, что и передача из RSR в DRR, приоритет имеет именно передача.

Данные сдвигаются влево (от LSB к MSB). Предполагается, что принято 8-разрядное слово и старшие три байта буфера приёма изначально не определены. На первой части рисунка А.8.4 байт "a" был принят. Когда принимается байт "b", байт "a" сдвигается влево. Когда считывается регистр приёма данных, считываются оба байта "a" и "b".

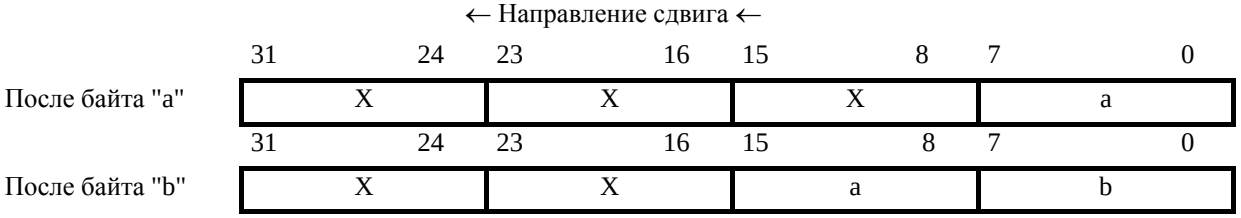


Рисунок А.8.4 – Работа сдвига буфера приёма

А.8.2.7 Установка параметров последовательного порта

Обеспечиваются несколько установок для работы таймера последовательного порта и его синхронизации. Синхроимпульсы для каждого последовательного порта могут вырабатываться как внутри, так и подаваться извне.

A.8.2.8 Временные диаграммы последовательного порта

Формула для вычисления частоты синхронизации последовательного порта с внутренней синхронизацией зависит от режима работы таймеров последовательного порта и может быть определена как:

f(импульсный режим) = f(такт таймера)/регистр периода

f(тактовый режим) = f(такт таймера)/(2 × регистр периода)

Источник внутреннего синхроимпульса f(такт таймера) имеет максимальную частоту f(H1)/2. Внешний источник синхронизации f(такт таймера) (CLKX или CLKR) имеет максимальную частоту меньше, чем f(H1)/2,6. См. временные диаграммы последовательного порта в A.8.1.3 относительно генерации импульс/такт таймера.

Передаваемые данные выдаются по возрастающему фронту выбранного синхросигнала порта. Принимаемые данные записываются в приёмный регистр сдвига по заднему фронту синхросигнала порта. Все данные передаются и загружаются, начиная со старшего разряда и выравненными по правой границе. При передаче менее 32 разрядов, данные выравниваются в 32-разрядных буферах приёма и передачи. Таким образом, младшие разряды буфера передачи – разряды, которые передаются.

Сигнал готовности передачи (XRDY) определяет готовность регистра передачи данных (DXR) к загрузке новых данных. XRDY становится активным, как только данные загружены в сдвиговый регистр передачи (XSR). Последнее слово может ещё выдаваться, когда XRDY становится активным. Если DXR загружается до того, как закончена передача последнего слова, передаваемые разряды данных будут последовательными (непрерывными), т. е. младший разряд первого слова непосредственно предваряет старший разряд второго, с выдачей всех сигналов как при двух независимых передачах. XRDY становится неактивным после загрузки DXR и остаётся неактивным, пока данные загружены в сдвигатель.

Сигнал готовности приёма (RRDY) активен, пока новое слово данных загружено в регистр приёма данных и не считано. Как только данные считаны, разряд RRDY очищается.

Когда FSX определён как выход, активность сигналов определена однозначно внутренним состоянием последовательного порта (п/порт – далее используемое в таблицах сокращение). Если определён фиксированный период данных, FSX становится активным, когда DXR загружен в XSR для передачи.

Через один цикл синхросигнала порта FSX становится неактивным, и начинается передача данных. Если определён переменный период данных, вывод FSX становится активным при начале передачи данных и остаётся активным в течение всей передачи слова. Аналогично данные передаются через один цикл синхросигнала после их загрузки в регистр передачи данных.

Входной FSX в режиме фиксированной скорости передачи данных становится активным в течение, по крайней мере, одного цикла синхросигнала порта и снимается для начала передачи. После этого передатчик пересылает разряды, число которых определено разрядами LEN. В режиме переменного периода данных передатчик начинает передачу с момента установления FSX в активный уровень до того момента, пока определённое число разрядов не будет выдано наружу. В режиме переменной скорости передачи данных при изменении статуса FSX до того, как все разряды будут выданы наружу, передача завершается, и вывод DX устанавливается в высокоимпедансное состояние. Вход FSR полностью комплементарен FSX.

При использовании внешнего FSX, если DXR и XSR пусты, запись в DXR приводит к пересылке DXR в XSR. Данные хранятся в XSR до возникновения FSX. При приёме внешнего FSX XSR начинает сдвиг данных. Если XSR ждёт внешнего FSX, запись в DXR изменяет DXR, но пересылка DXR в XSR не происходит. XSR начинает сдвиг при приёме внешнего FSX или когда он сброшен, используя XRESET.

Непрерывные режимы передачи и приёма

Когда выбран непрерывный режим, последующие записи не генерируются или ожидают сигнала нового синхроимпульса. Только первое слово блока начинается с активной синхронизации. После этого передача данных продолжается, пока загружаются данные в DXR до того, как будет передано последнее слово. Как только TXRDY становится активным, и все данные будут переданы из сдвигового регистра, вывод DX устанавливается в высокоимпедансное состояние и последующая запись в DXR инициирует новый блок и новый FSX.

Подобно FSR, приёмник продолжает сдвиг внутрь новых данных и загрузку DRR.

Если чтение из буфера приёма данных не происходит до того, как следующее слово сдвигается внутрь, последующие входящие данные будут потеряны. Разряд RFSM может быть использован для прерывания непрерывного режима приёма.

Режим квитирования передачи

Режим квитирования передачи (HS=1) обеспечивает прямое соединение между процессорами. В этом режиме все слова данных передаются с ведущей единицей. Например, при передаче 8-разрядного слова, первый переданный разряд будет единица, предваряя 8-разрядное слово данных. В этом режиме, передав слово, последовательный порт не будет передавать следующее слово, пока не примет отдельно переданный нулевой разряд. Таким образом, 1 разряд (leading one), предшествующий каждому слову данных, является, фактически, разрядом запроса.

После того, как последовательный порт принимает слово (с предшествующей 1), и это слово считано из DRR, принимающий последовательный порт передаёт единичный 0 (single zero) в передающий последовательный порт. Таким образом, единичный ноль рассматривается, как разряд подтверждения. Этот единичный разряд подтверждения передаётся каждый раз при чтении DRR, даже если DRR не содержит новые данные.

Когда последовательный порт установлен в режим квитирования передачи, внесение и удаление ведущей 1 для передаваемых данных и передача 0 для подтверждения приёма данных и ожидание этого разряда подтверждения, выполняется автоматически. При использовании этой схемы просто соединить процессоры без дополнительных аппаратных затрат и гарантировать надёжную связь. В режиме квитирования передачи FSX автоматически конфигурируется как выход. Непрерывный режим автоматически отменяется. После системного сброса или XRESET передатчик всегда установлен для передачи. Приёмник и передатчик должны быть сброшены после введения режима квитирования передачи.

А.8.2.9 Источники прерываний последовательного порта

Последовательный порт имеет четыре источника прерываний:

- первый – прерывание передающего таймера: передний фронт XTSTAT вызывает одноцикловый импульс прерывания. Когда XTINT = 0, импульс прерывания запрещён;

- второй – прерывание приёмного таймера: передний фронт RTSTAT вызывает одноцикловый импульс прерывания. Когда RTINT = 0, импульс прерывания запрещён;

- третий – прерывание передатчика: возникает непосредственно после пересылки DXR в XSR. Прерывание передатчика – одноцикловый импульс. Когда разряд регистра глобального управления последовательного порта XINT = 0, этот импульс прерывания запрещён;

- четвёртый – прерывание приёмника возникает непосредственно после пересылки RSR в DRR. Прерывание приёмника – одноцикловый импульс. Когда разряд глобального управления последовательного порта RINT = 0, этот импульс прерывания запрещён.

Импульс прерывания передающего таймера и импульс прерывания передатчика объединяются по логическому ИЛИ для генерации флага прерывания передачи ЦПУ XINT. Импульс прерывания приёмного таймера и импульс прерывания приёмника объединяются по логическому ИЛИ для генерации флага прерывания приёма ЦПУ RINT.

А.8.2.10 Функциональные операции последовательного порта

Далее иллюстрируются временные диаграммы функционирования различных режимов работы последовательного порта. Описание временных диаграмм приведено в предположении, что все полярности сигналов приняты положительными, т. е. CLKXP = CLKRP = DXR = DRP = FSXP = FSRP = 0. Логические временные диаграммы, когда одна или более полярности инвертированы, подобны, за исключением того, что касается противоположной полярности контрольных точек, т. е. передний вместо заднего фронта и т. д.

Эти обсуждения относятся к различным режимам работы и конфигурациям логики последовательного порта. Если необходимо переключить режимы работы или изменить конфигурации последовательного порта, то это надо делать только тогда, когда XRESET или RRESET установлен (низким), как предназначено. Следовательно, при изменении конфигурации передачи, XRESET должен быть низким, при изменении конфигурации приёма RRESET должен быть низким. При использовании режима квитирования передачи, однако, т. к. передатчик и приёмник взаимосвязаны, необходимо делать любые изменения конфигурации с обоими низкими XRESET и RRESET.

Все операции конфигурации последовательного порта могут быть в общих чертах классифицированы по двум категориям: временная диаграмма фиксированного периода данных и временная диаграмма переменного периода данных. Ниже обсуждается работа при фиксированной и переменной скорости передачи данных и всех её вариантах.

Временная диаграмма работы при фиксированной скорости передачи данных

Пересылки последовательного порта при фиксированной скорости передачи данных могут быть в двух вариантах: пакетный режим и непрерывный режим. В пакетном режиме работы пересылки отдельных слов отделены периодами отсутствия активности последовательного порта. В непрерывном режиме нет промежутков между успешными пересылками слов; первый разряд нового слова пересылается по последующему импульсу CLKX/CLKXR, следующим за последним разрядом предыдущего слова. Это происходит непрерывно до прерывания процесса.

В пакетном режиме с фиксированной скоростью передачи данных импульсы FSX/FSR начинают пересылки, и каждая пересылка включает одно слово. При внутренней генерации FSX, передача начинается загрузкой DXR. В этом режиме имеется задержка около 2,5 циклов CLKX (в зависимости от частоты CLKX и H1) с момента загрузки DXR до возникновения FSX. Если FSX – внешний, то импульс FSX инициирует передачу и 2,5-цикловая задержка действительно становится необходимой для обеспечения требования загрузки DXR по FSX. Следовательно, в этом случае DXR должен быть загружен не позднее трёх циклов CLKX до возникновения FSX. Как только XSR загружен из DXR, генерируется XINT.

При операциях приёма, когда пересылка уже началась, FSR игнорируется до последнего разряда. Для пересылок пакетного режима FSR должен быть низким в течение последнего разряда, или начнётся следующая пересылка. После приёма полного слова и пересылки в DRR, генерируется RINT.

В режиме фиксированной скорости передачи данных непрерывные пересылки могут выполняться тогда, когда R/XFSM = 0, пока обеспечен соответствующий кадр синхронизации, или пока DXR перезагружается каждый цикл при внутренней генерации FSX.

При операциях приёма с внешней генерацией FSX после того, как передача началась, кадровые синхроимпульсы требуются только в течение последнего передаваемого разряда для инициализации следующей непрерывной передачи. В противном случае входы кадровых синхроимпульсов игнорируются. Следовательно, непрерывная передача происходит, если кадровый синхроимпульс удерживается в высоком уровне. При внутренней генерации FSX возникает задержка около 2,5 циклов CLKX от загрузки DXR до возникновения FSX. Эта задержка возникает каждый раз при загрузке DXR; следовательно, в течение непрерывной передачи команда, которая загружает DXR, должна быть выполнена на N-3 разрядов для N-разрядной передачи. Т. к. задержки из-за конвейера могут изменяться, необходимо определить жёсткие пределы безопасности, принимая во внимание эту задержку.

После начала процесса в начале каждой пересылки генерируются RINT и XINT.

XINT указывает, что XSR был загружен из DXR и может быть использован для указания, что DXR должен быть перезагружен. Для обеспечения непрерывной передачи в режиме фиксированной скорости передачи с кадровыми синхроимпульсами, особенно при внутренней генерации FSX, DXR должен быть перезагружен в поступающей пересылке.

RINT показывает, что полное слово принято и переслано в DRR. Следовательно, RINT обычно используется для указания необходимости чтения из DRR.

Непрерывные пересылки прерываются путём прерывания кадровых синхроимпульсов или при внутренней генерации FSX, не перезагружая DXR.

Непрерывные пересылки последовательного порта могут быть завершены без использования кадровых синхроимпульсов, если R/XFSM установлен в 1. В этом режиме работа последовательного порта подобна работе с кадрами, за исключением того, что импульс кадровой синхронизации установлен только при передаче первого слова, но больше кадровые синхроимпульсы не используются. В течение передачи первого слова не генерируются внутренние синхроимпульсы и входные синхроимпульсы игнорируются. Кроме того, R/XFSM должен быть установлен до или в течение пересылки первого слова и должен быть установлен не позже, чем пересылается N-1 разряд первого слова, за исключением операций передачи. Для операций передачи в режиме фиксированной скорости передачи данных XFSM должен быть установлен не позднее передачи N-2 разряда. Очистка R/XFSM должна быть произведена не позднее, чем разряд N-1 должен быть распознан в текущем цикле.

Временные диаграммы RINT и XINT и пересылки данных в/из DXR и DRR, соответственно, такие же, как и в непрерывном режиме с фиксированной скоростью передачи данных с кадровыми синхроимпульсами.

В этом режиме работы также имеет место задержка в 2,5 цикла CLKX после загрузки DXR до генерации внутреннего FSX. Как и в случае работы в непрерывном режиме с фиксированной скоростью передачи данных, R/XFSM может быть очищен или установлен, как требуется, в том числе в течение действительных пересылок, для разрешения или запрещения использования кадровых синхроимпульсов, как диктуется системными требованиями. В зависимости от многих условий, эффект изменения состояния R/XFSM обнаруживается в течение пересылки, в которой производилось изменение R/XFSM, при условии, что изменение было сделано заранее в течение пересылки. Однако, для операций передачи с внутренним FSX в режиме фиксированной скорости передачи данных имеет место задержка на одно слово до генерации кадрового синхроимпульса, возникающего при сбросе XFSM в ноль. Следовательно, в этом случае передаётся одно дополнительное слово до генерации следующего FSX. Необходимо обратить также внимание на то, что как обсуждалось выше, очистка XFSM будет распознаваться в течение текущего слова, которое будет передаваться, если только XFSM очищен не позднее, чем на N-1 разряде. Установка XFSM распознается тогда, когда XFSM установлен не позднее N-2 разряда.

Временная диаграмма операции с переменной скоростью передачи данных

Операции с переменной скоростью передачи данных также поддерживают два типа работы: пакетный и непрерывный. Пакетный режим с переменной скоростью передачи данных подобен пакетному с фиксированной скоростью передачи данных. В режиме с переменной скоростью передачи данных, однако, FSX/FSR и синхронизация данных немного отличаются в начале и в конце пересылок. Три основных отличия между временной диаграммой с фиксированной и с переменной скоростями передачи данных:

- импульсы FSX/FSR продолжаются в течение определённого интервала пересылки, хотя FSR и внешний FSX игнорируются после пересылки первого разряда. Импульсы

FSX/FSR в режиме фиксированной скорости передачи данных обычно продолжаются в течение только одного цикла CLKX/CLKR, но могут продолжаться и более долго;

- пересылка данных начинается в течение цикла CLKX/CLKR, в котором возникает FSX/FSR, в отличие от цикла CLKX/CLKR, следующим за FSX/FSR, как в случае фиксированного периода данных;

- при переменной скорости передачи данных кадровые импульсы игнорируются до конца передачи последнего разряда, в отличие от начала передачи последнего разряда, как в случае фиксированного периода данных.

При непрерывной передаче в режиме переменной скорости передачи данных с кадровым синхроимпульсом, временная диаграмма такая же, как для режима фиксированной скорости передачи данных, за исключением различий между этими двумя режимами, которые описывались для пакетного режима. Единственное исключение состоит в том, что DXR должен быть загружен не позднее, чем на N-4 бите данных для обеспечения непрерывной операции в режиме переменной скорости передачи данных, не позднее N-3 для режима фиксированной скорости передачи данных.

Непрерывная операция в переменном режиме без кадровых синхроимпульсов также подобна непрерывной без кадровых синхроимпульсов в режиме фиксированной скорости передачи. Как и в переменном непрерывном режиме с кадровыми синхроимпульсами, DXR должен быть перезагружен не позднее, чем на N-4 бите данных для обеспечения выполнения непрерывной операции. Кроме того, когда R/XFSM установлен или очищен в переменном режиме, изменение должно быть произведено не позднее, чем на N-1 бите, чтобы повлиять на текущую пересылку.

A.8.2.11 Примеры использования интерфейса последовательного порта ПЦОС

При использовании режима квитирования передачи сигналами передачи и приёма данных являются (FSX/DX/CLKX) и (FSR/DR/CLKR), соответственно. Другими словами, когда ПЦОС принимает данные только в режиме квитирования передачи, сигналы передачи также нужны для передачи сигнала подтверждения.

Так как при выборе режима квитирования передачи FSX установлен как выход, и непрерывный режим запрещён, разряды XFSM и RFSM должны быть установлены в 0, разряд FSXOUT в регистре глобального управления должен быть установлен в 1. Для начала связи в режиме квитирования передачи разряды XRESET, RRESET и HS должны быть установлены в 1. Рекомендуется, чтобы полярность выводов последовательного порта была с активным высоким уровнем для простоты. Хотя CLKX, CLKR могут быть установлены и как входы и как выходы, рекомендуется установить CLKX как выход, а CLKR – как вход.

Таймер последовательного порта необходим тогда, когда CLKX или CLKR установлены как выходы. В вышеописанном случае, т. к. только CLKX установлен как выход, регистр управления таймера должен быть установлен в 0Fh. Когда используется таймер порта, регистр периода таймера также должен быть установлен в соответствующее значение для синхронизации скорости. Скорость тактирования таймера последовательного порта устанавливается аналогично таймеру ПЦОС, см. подраздел A.8.1.

Максимальная частота синхронизации для последовательных пересылок равна F(CLKIN)/4, если используется внутренняя синхронизация и равна F(CLKIN)/5,2, если используется внешняя синхронизация. Следовательно, если два ПЦОС имеют общую синхронизацию, как в вышеописанном случае, регистр периода таймера должен быть установлен в значение, большее или равное 1, что устанавливает частоту синхронизации в F(CLKIN)/8.

Примеры установок регистров последовательного порта для вышеописанного случая приведены ниже (предполагается, что два ПЦОС имеют общую синхронизацию).

Установка 1:

Глобальное управление = 0EBC0064h; 32-разрядный, фиксированный период
Управление передачи порта = 0111h; данных, пакетный режим
Управление приёма порта = 0111h; FSX, CLKX (выходы) = F(CLKIN)/8
Управление таймером п/порта = 0Fh ; CLKR (вход), режим квитирования
Управление счётчиком таймером последовательного порта = 0h; передачи, прерывания приёма
Управление периодом таймером п/порта ≥ 01h ; и передачи разрешены.

Установка 2:

Глобальное управление = 0C000364h; 8-разрядный, переменный период
Управление передачи порта = 0111h; данных, пакетный режим
Управление приёма порта = 0111h; FSX, CLKX (выходы) = F(CLKIN)/24
Управление таймером п/порта = 0Fh; CLKR (вход), режим квитирования
Управление счётчиком таймером п/порта = 0h; передачи, прерывания приёма
Управление периодом таймером п/порта ≥01h; и передачи запрещены.

Так как данные имеют предшествующую 1 и сигнал подтверждения в виде 0 в режиме квитирования передачи, последовательный порт ПЦОС может отличать сигналы данных и подтверждения. Следовательно, если при приёме данных последовательный порт принимает данные до сигнала подтверждения, данные не могут быть неверно интерпретированы, как сигнал подтверждения, и потеряны. Кроме того, сигнал подтверждения не генерируется, пока данные читаются из регистра приёма данных DRR. Следовательно, ПЦОС не будет передавать данные и сигнал подтверждения одновременно.

А.8.2.12 Инициализация/реконфигурация последовательного порта

Последовательный порт управляется через картированные в памяти регистры, адресуемые через соответствующую периферийную шину. Общая процедура инициализации и/или реконфигурации последовательного порта следующая:

- остановить последовательный порт, очистив разряды XRESET и/или RRESET регистра глобального управления порта. Чтобы сделать это, необходимо записать 0 в регистр глобального управления порта. Необходимо помнить, что порт останавливается по RESET#;
- сконфигурировать порт через регистр глобального управления порта (с XRESET = RRESET = 0) и регистры управления FSX/DX/CLKX и FSR/DR/CLKR. Если необходимо, определить регистры приёма/передачи: управления таймера (с XHLD=RHLD = 0), счётчика таймера и периода таймера;
- запустить операцию последовательного порта с помощью установки разрядов XRESET и RRESET регистра глобального управления порта и разрядов XHLD и RHLD таймера приёма/передачи последовательного порта, если требуется.

А.8.3 Контроллер ПДП

Процессорное ядро имеет встроенный контроллер прямого доступа к памяти (ПДП), который обеспечивает потребности ЦПУ при выполнении функций ввода-вывода.

Контроллер ПДП может выполнять операции ввода-вывода без взаимодействия операциями ЦПУ. Следовательно, существует возможность взаимодействия с медленными устройствами (внешней памятью и периферией, такой как ЦАП/АЦП, последовательные порты и т. д.) без использования ЦПУ.

Пересылки ПДП состоят из двух операций: чтения из адреса памяти и запись по адресу памяти. Контроллер ПДП может читать из и записывать по любому адресу карты памяти ПЦОС, включая и всю картированную в памяти периферию. Работа ПДП управляется следующим набором картированных в памяти регистров:

- регистром глобального управления ПДП;
- регистром адреса источника ПДП;
- регистром адреса назначения ПДП;
- регистром счётчика пересылок ПДП.

А.8.3.1 Регистр глобального управления ПДП

Регистр глобального управления ПДП управляет состоянием работы контроллера ПДП. Этот регистр также показывает состояние ПДП, которое изменяется каждый цикл. Адреса источника и назначения могут инкрементироваться, декрементироваться и синхронизироваться, используя определённые разряды регистра глобального управления. При сбросе системы все разряды регистра управления ПДП устанавливаются в 0. В таблице А.8.7 приведён список разрядов регистра, их имена и функции.

Таблица А.8.7 – Разряды регистра глобального управления ПДП

Разряд	Имя	Значение при сбросе	Функция
1	2	3	4
0-1	START	0 0	Управляет состоянием, в котором ПДП стартует и останавливается. ПДП может быть остановлен без потери данных (см. таблицу А.8.8)
2-3	STAT	0 0	Эти разряды показывают состояние ПДП и изменяются каждый цикл (см. таблицу А.8.9)
4	INCSRC	0	Если INCSRC=1, адрес источника инкрементируется после каждого чтения
5	DECSRC	0	Если DECSRC=1, адрес источника декрементируется после каждого чтения. Если DECSRC =INCSRC, адрес источника не изменяется после чтения
6	INCDST	0	Если INCDST=1, адрес назначения инкрементируется после каждого чтения
7	DECDST	0	Если DECDST=1, адрес назначения декрементируется после каждого чтения. Если DECDST=INCDST, адрес назначения не изменяется после чтения
9-8	SYNC	0 0	Разряды SYNC определяют синхронизацию между событиями, начинающими пересылки источника и назначения. Интерпретация разрядов SYNC приведена в таблице А.8.10
10	TC	0	Разряд TC влияет на работу счётчика пересылок. Если TC=0, пересылки не прерываются, когда счётчик пересылок становится равным 0. Если TC=1, пересылки не прерываются, когда счётчик пересылок становится равным 0

Окончание таблицы А.8.7

1	2	3	4
11	TCINT	0	Если TCINT=1, устанавливается прерывание ПДП, когда счётчик пересылок переходит через 0. Если TCINT=0, прерывание ПДП не устанавливается, когда счётчик пересылок переходит через 0
31-12	Зарезервированы	0 0	Считываются как 0

Таблица А.8.8 – START разряды для режимов ПДП

START	Функция
0 0	Циклы чтения или записи ПДП, находящиеся в работе, будут завершены; любое чтение данных будет игнорироваться. Любые незавершённые чтения/записи будут прерваны. ПДП сбрасывается в состояние момента старта, новая пересылка начинается; т. е. выполняется чтение (это значение устанавливается при сбросе)
0 1	Если чтение или запись начаты, они завершатся до остановки: например, в середине или в конце пересылок ПДП. Если не начиналось чтение или запись, они не начинаются
1 0	Если пересылка ПДП началась, полная пересылка завершится (включая операции чтения и записи) до останова. Если пересылка не началась, она не начинается
1 1	ПДП стартует со сброса или перестартует с предыдущего значения

Таблица А.8.9 – STAT разряды для режимов ПДП

STAT	Функция
0 0	ПДП остановлен между пересылками ПДП (между записью и чтением). Это значение устанавливается при сбросе.
0 1	ПДП остановлен в середине пересылки ПДП, т. е. между чтением и записью.
1 0	Зарезервировано.
1 1	ПДП занят, т. е. ПДП выполняет чтение или запись, или ждёт прерывания синхронизации источника или назначения.

Таблица А.8.10 – SYNC разряды для режимов ПДП

SYNC	Функция
0 0	Нет синхронизации. Разрешённые прерывания игнорируются (значение при сбросе)
0 1	Синхронизация источника. Чтение выполняется при возникновении разрешённого прерывания
1 0	Синхронизация назначения. Запись выполняется при возникновении разрешённого прерывания
1 1	Синхронизация источника и назначения. Чтение выполняется при возникновении разрешённого прерывания. Запись выполняется при возникновении следующего разрешённого прерывания

А.8.3.2 Регистры адреса источника и назначения

Регистры адреса источника и назначения – 24-разрядные регистры, содержание которых определяет адреса источника и назначения. Как определено управляющими разрядами DECSRC, INCSRC, DECDST и INCDST регистра глобального управления, эти регистры инкрементируются или декрементируются в конце соответствующего обращения к памяти регистра источника для чтения и регистра назначения для записи. При сбросе системы в эти регистры записывается 0.

A.8.3.3 Регистр счётчика пересылок

Регистр счётчика пересылок – 24-разрядный регистр, управляемый 24-разрядным счётчиком, который считает в обратную сторону. Счётчик уменьшается в начале записи памяти ПДП. Таким образом, он может быть использован для управления размером блока передаваемых данных. Регистр счётчика пересылок устанавливается в 0 при системном сбросе. Когда разряд TCINT регистра глобального управления установлен, регистр счётчика пересылок будет определять установку флага прерывания ПДП по установке счётчика в 0.

A.8.3.4 Регистр разрешения прерываний ЦПУ/ПДП

Регистр разрешения прерываний ЦПУ/ПДП (IE) – 32-разрядный регистр, размещённый в регистровом файле ЦПУ. Разряды разрешения прерываний ЦПУ/ПДП размещаются в адресах 10-1. Разряды разрешения прерываний ПДП размещаются в адресах 26-16. 1 в разряде регистра разрешения прерываний ЦПУ/ПДП разрешает соответствующее прерывание, 0 запрещает соответствующее прерывание. При сбросе в этот регистр записывается 0.

В таблице А.8.11 приведён список разрядов, имён и функций регистра разрешения прерываний ЦПУ/ПДП. Приоритеты и декодирование схем прерываний ЦПУ и ПДП идентичны. Следует помнить, что когда ПДП получает прерывание, это прерывание запускается в зависимости от поля SYNC регистра управления ПДП.

Таблица А.8.11 – Разряды регистра разрешения прерываний ЦПУ/ПДП

Разряд	Имя	Функция
0	EINT0	Разрешает внешнее прерывание 0 (ЦПУ)
1	EINT1	Разрешает внешнее прерывание 1 (ЦПУ)
2	EINT2	Разрешает внешнее прерывание 2 (ЦПУ)
3	EINT3	Разрешает внешнее прерывание 3 (ЦПУ)
4	EXINT0	Разрешает прерывание передачи п/порта 0 (ЦПУ)
5	ERINT0	Разрешает прерывание приёма п/порта 0 (ЦПУ)
6	EXINT1	Разрешает прерывание передачи п/порта 1 (ЦПУ)
7	ERINT1	Разрешает прерывание приёма п/порта 1 (ЦПУ)
8	ETINT0	Разрешает прерывание таймера 0 (ЦПУ)
9	ETINT1	Разрешает прерывание таймера 1 (ЦПУ)
10	EDINT	Разрешает прерывание контроллера ПДП (ЦПУ)
11-15	Зарезервированы	Считываются как 0
16	EINT0	Разрешает внешнее прерывание 0 (ПДП)
17	EINT1	Разрешает внешнее прерывание 1 (ПДП)
18	EINT2	Разрешает внешнее прерывание 2 (ПДП)
19	EINT3	Разрешает внешнее прерывание 3 (ПДП)
20	EXINT0	Разрешает прерывание передачи п/порта 0 (ПДП)
21	ERINT0	Разрешает прерывание приёма п/порта 0 (ПДП)
22	EXINT1	Разрешает прерывание передачи п/порта 1 (ПДП)
23	ERINT1	Разрешает прерывание приёма п/порта 1 (ПДП)
24	ETINT0	Разрешает прерывание таймера 0 (ПДП)
25	ETINT1	Разрешает прерывание таймера 1 (ПДП)
26	EDINT	Разрешает прерывание контроллера ПДП (ПДП)
27-31	Зарезервированы	Считываются как 0

А.8.3.5 Операция пересылки памяти ПДП

Каждая операция пересылки памяти ПДП состоит из двух частей:

- чтение данных по адресу, определённому в регистре источника ПДП;
- запись считанных данных по адресу, определённому регистром назначения ПДП.

Циклы	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Внутренний начальный адрес	R		R		R													
Внутренний конечный адрес		W		W		W												
Начальный адрес – основная шина	R.	R.	R.	I		R.	R.	R.	I		R.	R.	R.	I				
Внутренний конечный адрес		C_r					C_r					C_r						
				W						W					W			
Начальный адрес – расширенная шина	R.	R.	R.	I		R.	R.	R.	I		R.	R.	R.	I				
Внутренний конечный адрес		C_r					C_r					C_r						
				W						W					W			

Начальный адрес	Конечный адрес: внутренний
Внутренний	(1+1)T
Основная шина	[(2+Cr)+1]T
Расширенная шина	[(2+Cr)+1]T

Принятые условные обозначения:

- T – число передач;
- C_r – состояние ожидания чтения источника;
- R – одноцикловые чтения;
- W – одноцикловые записи;
- R.R – многоцикловые чтения;
- W.W – многоцикловые записи;
- I – цикл внутреннего регистра.

Рисунок А.8.5 – Временная диаграмма и число циклов передачи данных ПДП во внутренний адрес

Пересылка завершена только после выполнения чтения и записи. Пересылка может быть остановлена установкой разрядов START в требуемое значение. При перезапуске ПДП (STAT = 11), он завершает любую ожидаемую пересылку.

Циклы	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Внутренний начальный адрес	R		R				R											
Конечный адрес – основная шина		W.	W.	W.	W	W.	W.	W.	W	W.	W.	W.	W					
				C_r				C_r				C_r						
Начальный адрес – основная шина	R.	R.	R	I					R.	R.	R	I						
		C_r								C_r								
Конечный адрес – основная шина					W.	W.	W.	W					W.	W.	W.	W		
							C_w								C_w			
Начальный адрес – расширенная шина	R.	R.	R	I		R.	R.	R	I		R.	R.	R	I				
		C_r					C_r					C_r						
Конечный адрес – основная шина					W.	W.	W.	W		W.	W.	W.	W		W.	W.	W.	W
							C_w					C_w					C_w	

Начальный адрес	Конечный адрес: основная шина
Внутренний	$1+(2+C_w)T$
Основная шина	$(2+C_r+2+C_w)T$
Расширенная шина	$(2+C_r+2+C_w)+[(2+C_w+ \max(0,C_r-C_w+1)](T-1)$

Принятые условные обозначения:
- T – число передач;
- C_r – состояние ожидания чтения источника;
- C_w – состояние ожидания записи;
- R – одноцикловые чтения;
- W – одноцикловые записи;
- R.R – многоцикловые чтения;
- W.W – многоцикловые записи;
- I – цикл внутреннего регистра.

Рисунок А.8.6 – Временная диаграмма и число циклов передачи данных ПДП на основную шину

В конце чтения ПДП адрес источника изменяется согласно значению в разрядах SRCINC и SRCDEC регистра глобального управления ПДП. В конце записи ПДП адрес назначения изменяется согласно значению в разрядах DSTINC и DSTDEC регистра глобального управления ПДП. По окончании каждой записи ПДП счётчик пересылок ПДП уменьшается.

Запись и чтение в пределах внутриядерных областей (записи и чтения внутренней памяти и периферии) совершаются за один цикл. Чтение ПДП внешней памяти совершается за два цикла. Первый цикл – внешнее чтение, второй цикл загружает регистр ПДП. Внешний цикл чтения ПДП аналогичен внешнему циклу чтения ЦПУ. Запись ПДП во внешнюю память аналогична записи ЦПУ во внешнюю память. Если ПДП запущен и передаёт данные через одну из внешних шин, регистр управления шиной, связанный с данной шиной, не дол-

жен изменяться. Если регистр управления шиной (см. раздел А.7) должен быть изменён, необходимо остановить ПДП, произвести изменение и перезапустить ПДП. Если это сделать неверно, в результате может возникнуть непредвиденный доступ шины с нулевым состоянием ожидания.

Посредством 24-разрядных регистров источника и назначения ПДП может адресовать любой картированный в памяти адрес карты памяти ПЦОС. Рисунки А.8.5 и А.8.6 иллюстрируют временные диаграммы пересылок ПДП. |R| и |W| – одноцикловые чтение и запись, соответственно |R.R| и |W.W| – многоцикловые чтение и запись. $|C_r|$ и $|C_w|$ показывают число циклов ожидания для чтения и записи.

В таблице А.8.12 приведены максимальные значения скорости пересылок при отсутствии состояний ожидания ($C_r = C_w = 0$). В таблице А.8.13 приведены максимальные значения скорости пересылок при наличии одного состояния ожидания для чтения ($C_r = 1$) и отсутствии состояний ожидания для записи ($C_w = 0$). В таблице А.8.14 приведены максимальные значения скорости пересылок при наличии одного состояния ожидания для чтения ($C_r = 1$) и одного состояния ожидания для записи ($C_w = 1$). В таблицах подразумевается скорость полной пересылки ПДП (чтение и запись). Т. к. требуется одно обращение к шине для чтения и одно для записи, скорость внутренних передач по шине вдвое больше скорости передач ПДП.

Предполагается также, что отсутствуют конфликты с ЦПУ.

Таблица А.8.12 – Максимальные скорости пересылок, когда $C_r = C_w = 0$

Источник	Назначение			Единица измерения
	Внутренняя	Основная	Расширенная	
Внутренняя	40,0	40,0	40,0	Мбайт/с
Основная	26,7	20,0	26,7	Мбайт/с
Расширенная	26,7	26,7	20,0	Мбайт/с

Таблица А.8.13 – Максимальные скорости пересылок, когда $C_r = 1$, $C_w = 0$

Источник	Назначение			Единица измерения
	Внутренняя	Основная	Расширенная	
Внутренняя	40	40	40	Мбайт/с
Основная	20	16	20	Мбайт/с
Расширенная	20	20	16	Мбайт/с

Таблица А.8.14 – Максимальные скорости пересылок, когда $C_r = 1$, $C_w = 1$

Источник	Назначение			Единица измерения
	Внутренняя	Основная	Расширенная	
Внутренняя	40	26,7	26,7	Мбайт/с
Основная	20	13,3	20,0	Мбайт/с
Расширенная	20	20,0	13,3	Мбайт/с

А.8.3.6 Синхронизация каналов ПДП

Канал ПДП может быть синхронизирован через использование прерываний. В таблице А.8.10 приведены соотношения между разрядами SYNC регистра глобального управления ПДП и выполняемой синхронизацией. В данном разделе обсуждаются следующие четыре механизма синхронизации:

- отсутствие синхронизации (SYNC = 0 0);
- синхронизация источника (SYNC = 0 1);
- синхронизация назначения (SYNC = 1 0);
- синхронизация источника и назначения (SYNC = 1 1).

Отсутствие синхронизации

Когда SYNC = 0 0, не выполняется синхронизация. ПДП выполняет чтение и запись всякий раз при отсутствии конфликтов. Все прерывания игнорируются и, следовательно, должны быть глобально запрещены. Однако разряды регистра разрешения прерываний ПДП не изменяются.

Синхронизация источника

Когда SYNC = 0 1, ПДП синхронизирован по источнику. Чтение не будет производиться до того, как ПДП получит прерывание. Тогда все прерывания ПДП глобально запрещаются. Однако разряды регистра разрешения прерываний ПДП не изменяются.

Синхронизация по назначению

Когда SYNC = 1 0, ПДП синхронизирован по назначению. Сначала все прерывания игнорируются до окончания чтения. Предполагается, что все прерывания ПДП глобально запрещены, однако, разряды регистра разрешения прерываний ПДП не изменяются. Запись не будет производиться до того, как ПДП получит прерывание.

Синхронизация источника и назначения

Когда SYNC = 1 1, ПДП синхронизирован как по источнику, так и по назначению. Чтение выполняется по получении прерывания. Запись выполняется по получении следующего прерывания.

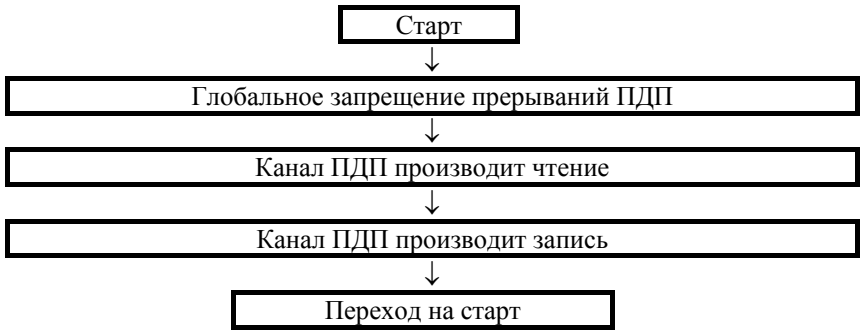


Рисунок А.8.7 – Отсутствие синхронизации ПДП

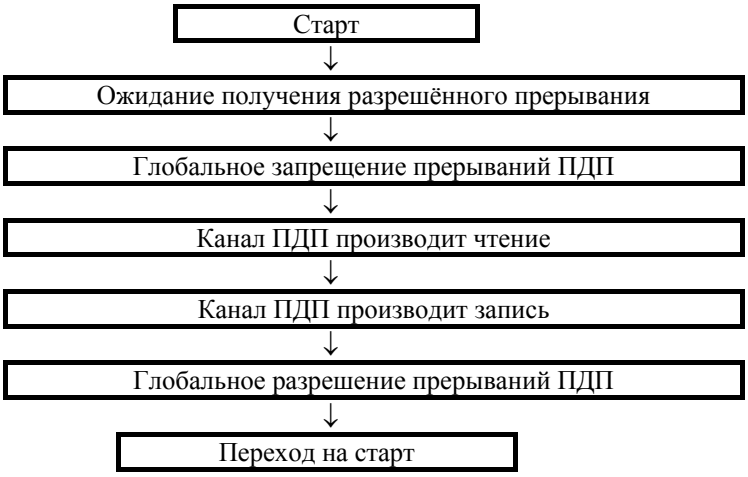


Рисунок А.8.8 – Синхронизация источника

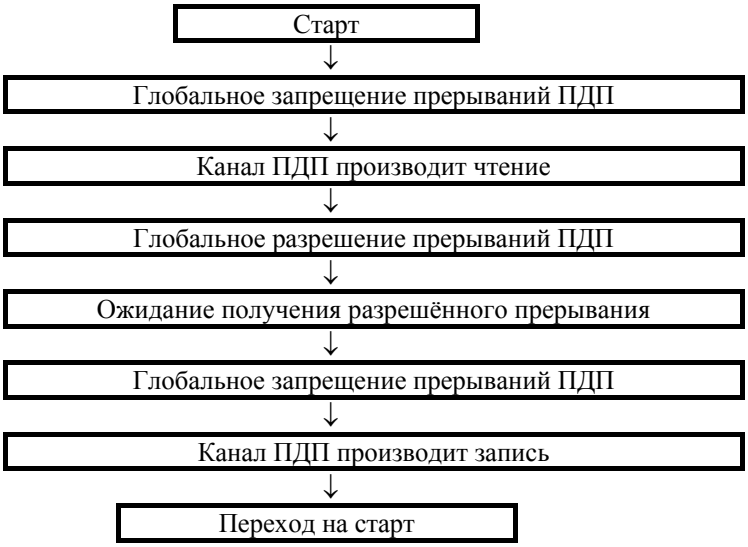


Рисунок А.8.9 – Синхронизация по назначению

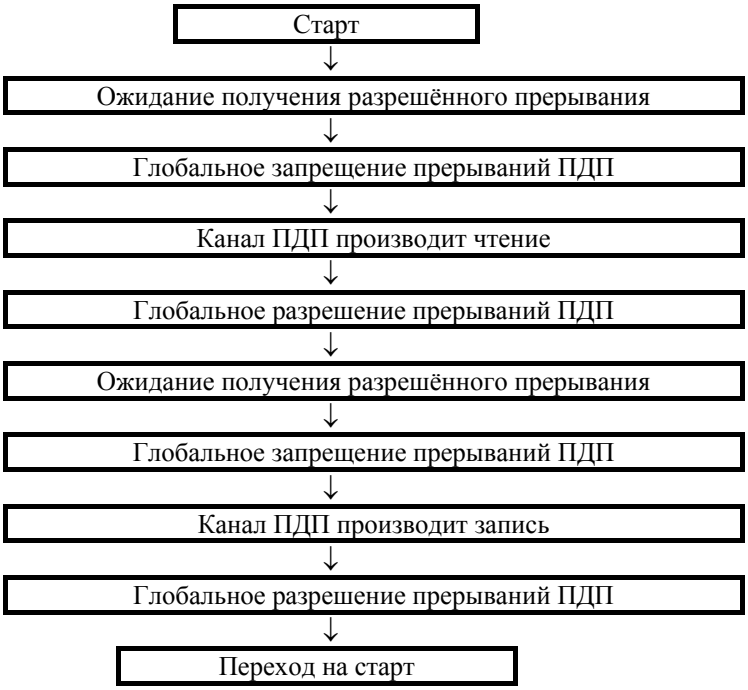


Рисунок А.8.10 – Синхронизация ПДП по источнику и назначению

A.8.3.7 Прерывания ПДП

Прерывания ПДП в ЦПУ могут генерироваться всякий раз, когда счётчик пересылок достигает нуля, показывая, что имеет место последняя пересылка. Разряд TCINT регистра глобального управления ПДП определяет, будет ли генерироваться прерывание. Если TCINT = 1, вырабатывается прерывание ПДП. Если TCINT = 0, прерывание не вырабатывается. Если вырабатываются прерывания ПДП, то 10 разряд в регистре разрешения прерываний EDINT должен быть установлен для разрешения прерывания ЦПУ из ПДП.

Второй разряд регистра глобального управления TC также связан с состоянием разряда TCINT и операцией прерывания. Разряд TC определяет, будут ли пересылки прерываться при равенстве нулю счётчика пересылок, или они будут продолжаться. Если TC = 1, пересылки будут прерываться при равенстве нулю счётчика пересылок. Если TC = 0, пересылки не будут прерываться при равенстве нулю счётчика пересылок.

В основном, если TCINT = 0, TC также должен быть установлен в 0. В противном случае, пересылка ПДП будет прервана, а ЦПУ не будет оповещён. Если TCINT=1, TC в большинстве случаев также должен быть 1. В этом случае ЦПУ будет оповещён при завершении пересылки, и ПДП будет остановлен и готов начать новую пересылку.

A.8.3.8 Примеры установки и использования ПДП

- Переслать блок из 256 слов из внешней во внутреннюю память и генерировать прерывание по окончании. Порядок использования памяти при выполнении.

Адрес источника ПДП: 80_0000h

Адрес назначения ПДП: 80_9800h

Счётчик пересылок ПДП: 0000_0100h

Глобальное управление ПДП: 0000_0C53h

Разрешение прерываний ЦПУ/ПДП (IE): 0000_0400h

- Переслать блок из 128 слов из внутренней во внешнюю память и генерировать прерывание по окончании. Порядок использования памяти должен быть инвертирован; таким образом, старший адресуемый элемент блока становится младшим.

Адрес источника ПДП: 80_9800h

Адрес назначения ПДП: 80_0000h

Счётчик пересылок ПДП: 0000_0080h

Глобальное управление ПДП: 0000_0C93h

Разрешение прерываний ЦПУ/ПДП (IE): 00000400h

- Переслать блок из 200 слов из регистра приёма последовательного порта 0 во внутреннюю память и генерировать прерывание по окончании. Пересылка должна быть синхронизирована с прерыванием приёма последовательного порта 0.

Адрес источника ПДП: 80_804Ch

Адрес назначения ПДП: 80_9C00h

Счётчик пересылок ПДП: 0000_00C8h

Глобальное управление ПДП: 0000_0D43h

Разрешение прерываний ЦПУ/ПДП (IE): 0020_0400h

- Переслать блок из 200 слов из внешней памяти в регистр передачи последовательного порта 0 и генерировать прерывание по окончании. Пересылка должна быть синхронизирована с прерыванием передачи последовательного порта 0.

Адрес источника ПДП: 80_9C00h

Адрес назначения ПДП: 80_8048h

Счётчик пересылок ПДП: 0000_00C8h

Глобальное управление ПДП: 0000_0E13h

Разрешение прерываний ЦПУ/ПДП (IE): 0040_0400h

- Пересылать данные непрерывно между приёмным регистром последовательного порта 0 и регистром передачи последовательного порта 0 для создания цифровой обратной связи. Пересылка должна быть синхронизирована с прерыванием приёма и передачи последовательного порта 0.

Адрес источника ПДП: 80_804Ch

Адрес назначения ПДП: 80_8048h

Счётчик пересылок ПДП: 0000_0000h

Глобальное управление ПДП: 0000_0303h

Разрешение прерываний ЦПУ/ПДП (IE): 0060_0000h

A.8.3.9 Инициализация/реконфигурация ПДП

ПДП управляется через картированные в памяти регистры, расположенные на соответствующей периферийной шине. Ниже дана общая процедура инициализации и/или реконфигурации ПДП:

- остановить ПДП, очистив разряд START регистра глобального управления ПДП. Это может быть сделано при записи 0 в регистр глобального управления ПДП. ПДП останавливается при сбросе RESET#;

- определить ПДП через регистр глобального управления ПДП (со START=00), а также регистры источника, назначения счётчика пересылок, если необходимо, см. A.8.3.8;

- запустить ПДП, установив разряды START регистра глобального управления ПДП как требуется.

A.9 Работа конвейера

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

Пять функциональных элементов управляют работой 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, контроллер ПДП освобождает ЦПУ от выполнения этих операций, снижая нагрузку конвейера и увеличивая вычислительную мощность.

Основные понятия, обсуждаемые в данном разделе:

- структура конвейера (подраздел A.9.1);
- конфликты конвейера (подраздел A.9.2);
- конфликты переходов (A.9.2.1);
- конфликты памяти (A.9.2.3);
- разрешение конфликтов регистров (подраздел A.9.3);
- разрешение конфликтов памяти (подраздел A.9.4);
- синхронизация обращений к памяти (подраздел A.9.5);
- выборки программ;
- загрузка и сохранение данных;
- обращения ПДП.

A.9.1 Структура конвейера

Пять функциональных элементов структуры конвейера ПЦОС и их функции следующие:

- выборка (F): выбирается слово команды из памяти и изменяется счётчик команд (PC);
- декодирование (D): декодируется слово команды и выполняется генерация адреса. Также управляет любыми изменениями вспомогательных регистров и указателем стека;
- чтение (R): если требуется, считывается операнд из памяти;
- выполнение (E): если требуется, считывается операнд из регистрового файла, выполняется необходимая операция и записывается результат в регистровый файл. Если требуется, результат предыдущей операции записывается в память;
- канал прямого доступа к памяти (ПДП): чтение и запись памяти.

Базовая команда имеет четыре уровня выполнения: выборка, декодирование, чтение и выполнение. Уровни индексированы в соответствии с циклом команды и выполнения. Полное перекрытие в конвейере, где все четыре элемента работают параллельно, возникает на цикле (m). Те уровни, которые должны быть почти выполнены на m+1, уже выполнены на уровне m-1. Управление конвейером обеспечивает высокоскоростное выполнение (E) за один цикл, при этом конфликты конвейера прозрачны для пользователя. Нет необходимости предпринимать какие-либо меры для гарантии корректного выполнения операции.

Цикл	F	D	R	E	
m-3	W	-	-	-	
m-2	X	W	-	-	
m-1	Y	X	W	-	
m	Z	Y	X	W	← полное перекрытие
m+1	-	Z	Y	X	
m+2	-	-	Z	Y	
m+3	-	-	-	Z	

Примечания

1 W, X, Y, Z – представляют команды.

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

Рисунок А.9.1 – Структура конвейера ПЦОС

Приоритеты от старшего к младшему, присвоенные для каждого из функциональных элементов, следующие:

- E – выполнение (высший),
- R – чтение,
- D – декодирование,
- F – выборка,
- ПДП – (низший).

Когда процесс обработки команды готов к переходу на следующий высший уровень конвейера, но этот уровень не готов к новому входу, возникает конфликт конвейера. В этом случае элемент нижнего уровня ждёт, пока устройство с высшим приоритетом завершит свою текущую функцию.

Несмотря на низкий приоритет контроллера ПДП, конфликты с ЦПУ могут быть минимизированы путём соответствующего структурирования данных, потому что контроллер ПДП имеет собственную шину данных и адреса.

А.9.2 Конфликты конвейера

Конфликты конвейера ПЦОС могут быть сгруппированы в следующие три категории:

- конфликты переходов: включают большинство команд или операций, которые читают и/или изменяют PC;

- конфликты регистров: включают задержки, которые могут возникнуть при чтении и записи в регистры, которые используются для генерации адреса;

- конфликты памяти: возникают, когда внутренние устройства ПЦОС находятся в состязании за ресурсы памяти.

Каждый из этих трёх типов обсуждается далее. Примеры прилагаются. Необходимо обратить внимание в этих примерах, что, когда данные перевыбираются или, операция повторяется, символ, представляющий стадию конвейера, дополняется номером. Например, если выборка выполняется снова, мнемоника команды повторяется. Когда обращение задерживает несколько циклов из-за отсутствия готовности, используются символы RDY# и RDY для указания отсутствия или наличия готовности, соответственно.

A.9.2.1 Конфликты переходов

Первый тип конфликтов конвейера возникает при стандартных (незадержанных) переходах, т. е. BR, Bcond, DBcond, CALL, IDLE, RPTB, RPTS, RETIcond, RETScond, прерываниях и сбросе. Конфликты возникают при 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 «промывкой» конвейера. «Промывка» конвейера необходима в этих случаях для гарантии, что команды не будут выполнены по частям.

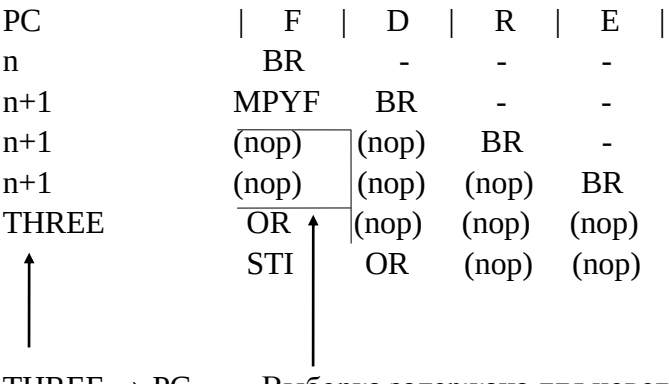
TRAPcond и CALLcond классифицируются отдельно от других типов переходов и рассматриваются позже.

В примере A.9.1 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 A.9.1 – Стандартное ветвление

```
   BR THREE      ; Безусловный переход
   MPYF          ; Не выполняется
   ADD           ; Не выполняется
   SUBF          ; Не выполняется
   AND           ; Не выполняется
   .
   .
THREE OR         ; Выбрана после выборки BR
   STI
   .
   .
```

Работа конвейера:



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 возникает "промывка" 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 быть, необходимо загрузить и сохранить эти регистры прямо во время использования режима повторения. Т. к. до четырёх команд может быть выбрано до введения режима повторения, загрузки должны предшествовать для очистки конвейера. Если RC изменяется при загрузке его командой, прямая загрузка имеет приоритет над изменением, производимым логикой режима повторений.

Задержанные переходы вводятся для гарантии выборки следующих трёх команд.

Задержанные переходы включают BRD, BcondD и DBcondD. В примере А.9.2 приведены программа и работа конвейера для задержанного перехода.

Пример А.9.2 – Задержанный переход

```
    BRD THREE    ; Безусловный задержанный переход
    MPYF         ; Выполняется
    ADD          ; Выполняется
    SUBF         ; Выполняется
    AND          ; Не выполняется
    .
    .
    .
THREE MPYF       ; Выбирается после выборки SUBF.
    .
    .
    .
```

Работа конвейера:

PC	F	D	R	E	
n	BRD	-	-	-	
n+1	MPYF	BRD	-	-	Не выполняется задержка
n+2	ADDF	MPYF	BRD	-	
n+3	SUBF	ADDF	MPYF	BRD	
THREE	MPYF	SUBF	ADDF	MPYF	

THREE → PC

А.9.2.2 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 подразделе А.9.3.

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);

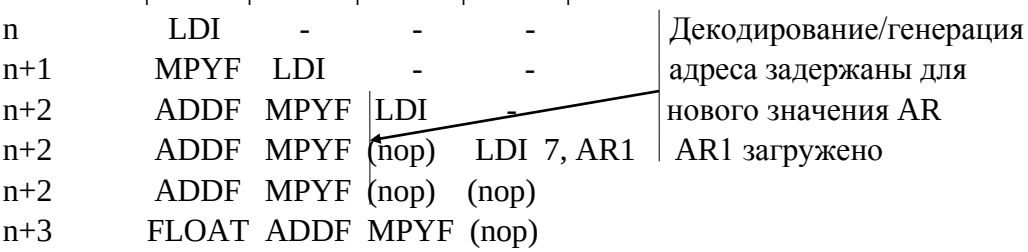
- группа 3: указатель системного стека (SP).

Если команда производит запись в одну из этих групп, элемент декодирования не может использовать любой регистр внутри группы до завершения записи, т. е. до завершения выполнения команды. В примере А.9.3 вспомогательный регистр загружается, а другой вспомогательный регистр используется для следующей команды. Т. к. 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. Т. к. это действительные перевыборки, они могут привести не только к конфликтам с контроллером ПДП, но и к кэш-попаданию и кэш-отсутствию.

Пример А.9.3 – Запись в AR с последующей генерацией адреса AR

```
        LDI 7, AR1; 7 → AR1
NEXT  MPYF *AR2, R0; Декодирование задержано на 2 цикла
        ADDF
        FLOAT
```

Работа конвейера:



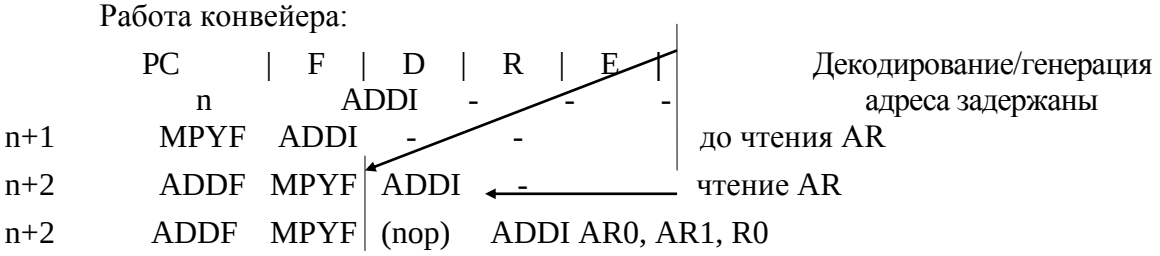
Случай чтения для этих групп аналогичен записи. Если команда должна считать значение в одном из элементов группы, использование той же группы при декодировании следующей команды задерживается до завершения чтения. Регистры считываются в начале цикла исполнения, поэтому требуется только один цикл задержки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

В примере А.9.4 производится сложение содержимого двух вспомогательных регистров с записью в регистр расширенной точности. Следующая команда использует другой вспомогательный регистр как адресный.

Пример А.9.4 – Чтение AR с последующим использованием AR для генерации адреса

```
        ADDI   AR0, AR1, R1          ; AR0 + AR1 → R1
NEXT   MPYF   *++AR2, R0             ; Декодирование, задержанное на 1 цикл
        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.

А.9.2.3 Конфликты памяти

Конфликты памяти могут возникать при превышении границы физической памяти. Например, блоки 0 и 1 ОЗУ могут поддерживать только 2 обращения за один цикл. Внешний интерфейс может поддерживать только одно обращение каждый цикл. Некоторые условия, при которых можно избежать конфликтов памяти, обсуждаются в подразделе А.9.4.

Конфликты конвейера при работе с памятью состоят из следующих четырёх типов:

- «ожидание программы» – выборка программы предотвращена с начала;
- «выборка программы не завершена» – выборка программы начата, но ещё не завершена;
- «только выполнение» – последовательность команды требует трёх обращений ЦПУ к данным за один цикл;
- «задержание всего» – операция основной или расширенной шины должна быть завершена до начала любой другой.

Эти типы конфликтов памяти обсуждены и проиллюстрированы примерами ниже.

«Ожидание программы»

Два условия могут предотвратить выборку программы.

1 Начало доступа ЦПУ к данным когда:

- происходит два обращения ЦПУ к внутренним блокам ОЗУ или ПЗУ и необходима выборка программы из того же блока;
- один из внешних портов начинает доступ к данным ЦПУ и необходима выборка программы из того же порта.

2 Требуется многоцикловое обращение ЦПУ или ПДП к данным через внешнюю шину.

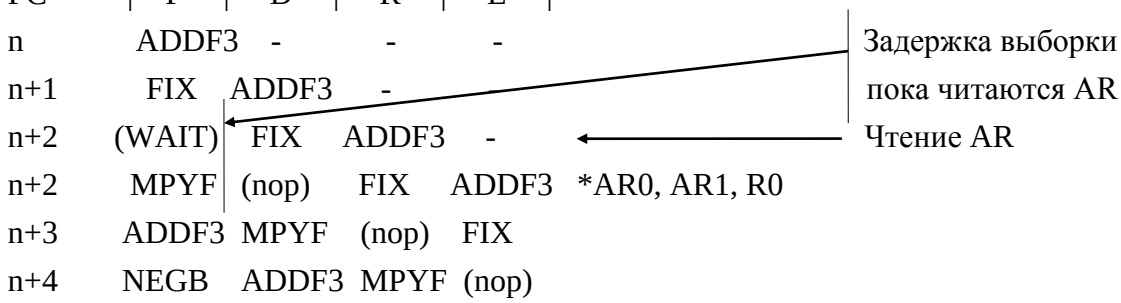
В примере А.9.5 иллюстрируется ожидание завершения обращения ЦПУ к данным.

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. Т. к. 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 А.9.5 – Программа ожидает завершения обращения ЦПУ к данным

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

Работа конвейера:



В примере А.9.6 приведено программное ожидание, связанное с многоцикловым обращением данные-данные или обращением ПДП. ADDF, MPYF и SUBF выбраны из некой области памяти иначе, чем требуется для внешнего порта ПДП. ПДП начинает многоцикловое обращение. Выборка программы, относящаяся к CALL, делается по тому же внешнему порту, который использует ПДП.

Эту ситуацию может вызвать одна из двух причин:

1 Пересечение одной из двух границ памяти:

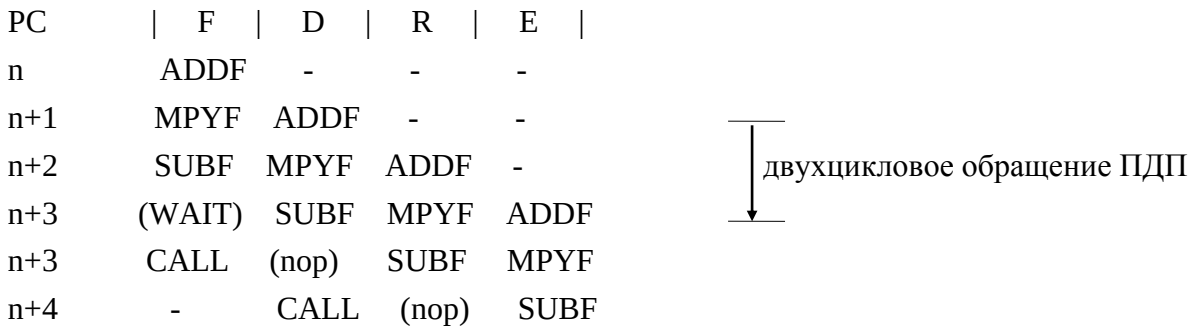
- от 7FFFFFh до 800000h;
- от 809FFFh до 80A000h.

2 Выполняемый код был кэширован и команда до ADDF одна из следующих (условная или безусловная): команда задержанного ветвления или команда задержанного декремента и ветвления.

Хотя ПДП имеет самый низкий приоритет, многоцикловое обращение не может быть прервано. Выборка программы должна ожидать окончания обращения ПДП.

Пример А.9.6 – Ожидание программы в связи с многоцикловым обращением

Работа конвейера:



«Выборка программы не завершена»

Незавершение выборки программы возникает, когда выборка программы требует более одного цикла для завершения из-за состояний ожидания. В примере А.9.7 MPYF и конфликт – выборка через границу банка в основном порте, см. подраздел А.7.4.

Пример А.9.7 – Многоцикловые выборки программной памяти

Работа конвейера:

PC		F	D	R	E	
n		MPYF	-	-	-	
n+1		ADDF	MPYF	-	-	
n+2	RDY	SUBF	ADDF	MPYF	-	требуется 1 состояние
n+2	RDY	SUBF	(nop)	ADDF	MPYF	ожидания
n+3		ADDI	SUBF	(nop)	ADDF	

Тип конфликта конвейера «Только выполнение»

Тип конфликта конвейера «только выполнение» возникает, когда последовательность команд требует три обращения ЦПУ к данным в один цикл или при выполнении блокированной загрузки. Возможны три случая, при которых возникает данный конфликт:

- команда выполняет сохранение и затем следует команда, выполняющая два чтения памяти;

- команда выполняет два сохранения и затем следует команда, выполняющая, по меньшей мере, одно чтение памяти;

- выполняется команда загрузки с блокировкой (LDII или LDFI) и XF= 1.

Т. к. эта последовательность требует трёх обращений к памяти данных, а поддерживаются только два, выполняется только фаза исполнения конвейера. Двойные чтения, требуемые LDF||LDF, задерживаются на один цикл. Обратите внимание, что может произойти перевыборка следующей команды.

Пример А.9.8 – Два чтения следуют за одиночным сохранением

```
STF  R0, *AR1    ; R0 → *AR1
LDF  *AR2, R1    ; *AR2 → R1  параллельно с
|| LDF  *AR3, R2  ; *AR3 → R2
```

Работа конвейера:

PC		F	D	R	E	
n		STF	-	-	-	
n+1		LDF||LDF	STF	-	-	
n+2		W	LDF||LDF	STF	-	Запись должна завершиться
n+3		X	W	LDF||LDF	STF 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 А.9.9 приводится параллельное сохранение, за которым следует одна загрузка или чтение. Т. к. требуются два параллельных сохранения, следующее чтение данных ЦПУ должно ожидать один цикл до начала. Может иметь место одна перевыборка памяти.

Пример А.9.9 – Одно чтение следует за параллельным сохранением

```
STF  R0, *AR0  ; R0 → *AR0  параллельно с
|| STF  R2, *AR1  ; R2 → *AR1
ADDF @SUM,R1  ; R1 + @SUM → R1
IACK
ASH
```

Работа конвейера:

PC	F	D	R	E	
n	STF||STF	-	-	-	Чтение должно ожидать пока
n+1	ADDF	STF||STF	-	-	завершатся записи
n+2	IACK	ADDF	STF||STF	-	
n+3	ASH	IACK	ADDF	STF||STF	R0, *AR0 and R2, *AR1
n+4	ASH	IACK	ADDF	(nop)	
n+4	-	ASH	IACK	ADDF	

Последний случай представляет команды загрузки с блокировкой (LDII или LDFI) и XF1 = 1. Т. к. загрузки с блокировкой используют вывод XF1 для подтверждения того, что чтение завершено, они могут потребовать расширения цикла чтения, как показано в примере А.9.10. Следует помнить, что может возникнуть перевыборка программы.

Пример А.9.10 – Загрузка с блокировкой

```
NOT   R1, R0
LDII  300h, AR2
ADII  *AR2, R2
CMPI  R0, R2
```

Работа конвейера:

PC	F	D	R	E	
n	NOT	-	-	-	
n+1	LDII	NOT	-	-	
n+2	ADDI	LDII	NOT	-	
n+3	CMPI	ADDI	LDII	NOT	XF1 = 1
n+3	-	CMPI	ADDI	LDII	XF1 = 0
n+4	-	CMPI	ADDI	LDII	

Типы конфликтов конвейера «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 А.9.11 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. Т. к. сохранение не завершено, ЦПУ продолжает попытки выполнить LDF, пока порт не станет доступен.

Пример А.9.11 – 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 А.9.12 выполняется LDF из внешней памяти, что требует нескольких циклов для завершения.

Пример А.9.12 – 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	

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 А.9.13. Дополнительный цикл используется для проталкивания адреса возврата после оценки условия вызова.

Пример A.9.13 – Условные вызовы и системные прерывания

Работа конвейера:

PC	F	D	R	E	
n	CALLcond	-	-	-	
n+1	I	CALLcond	-	-	
n+1	(nop)	(nop)	CALLcond	-	
n+1	(nop)	(nop)	(nop)	CALLcond	
n+1	(nop)	(nop)	(nop)	CALLcond	
n+2/CALLaddr	I	(nop)	(nop)	(nop)	Цикл сохранения PC

A.9.3 Разрешение конфликтов регистров

Если осуществляется обращение к вспомогательным (AR7-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 A.9.2.2. Примеры с A.9.14 по A.9.16 демонстрируют общее использование этих регистров, не ведущее к возникновению конфликта или пути, с помощью которых можно избежать этих конфликтов.

Пример A.9.14 – 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
```

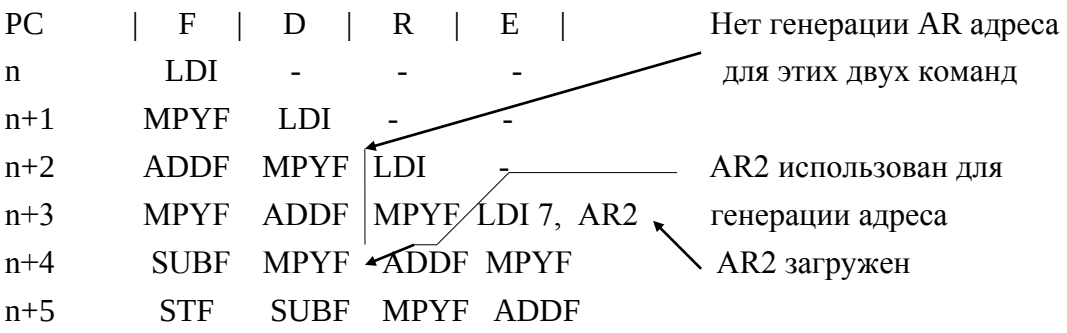
Работа конвейера:

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 A.9.15 – 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
```

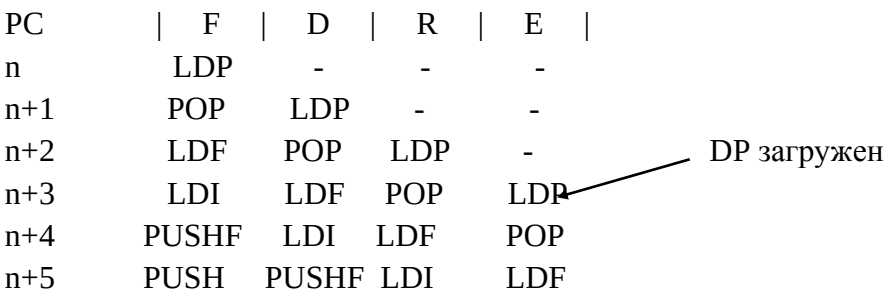
Работа конвейера:



Пример А.9.16 – 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:



А.9.4 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. Определённые конфигурации выборки программ и обращений к данным обеспечивают условия, при которых ПЦОС имеет максимальную производительность.

В таблице А.9.1 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 А.9.1 – Одна выборка программы и одно обращение к данным для максимальной производительности

Случай	Обращения по основной шине	Выборки из внутренней памяти	Обращения по расширенной шине
1	1	1	–
2	1	–	1
3	–	2 из любой комбинации внутренней памяти	–
4	–	1	1

В таблице А.9.2 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).

Таблица А.9.2 – Одна выборка программы и два обращения к данным для максимальной производительности

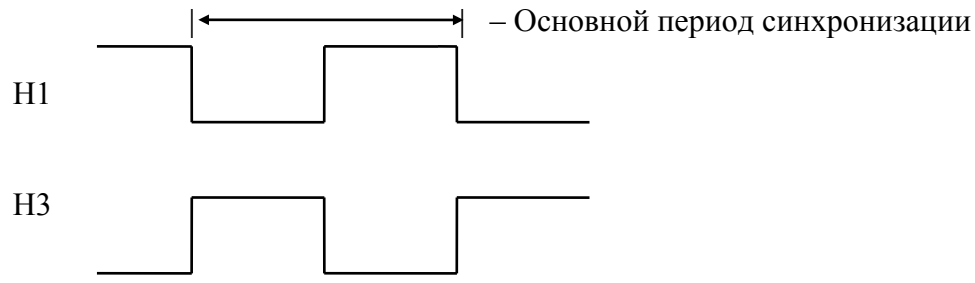
Случай	Обращения по основной шине	Выборки из внутренней памяти	Обращения по расширенной шине
1	1	2 из любой конфигурации памяти	–
2	1 программа	1 данные	1 данные
3	1 данные	1 данные	1 программа
4	–	2 из одного блока внутренней памяти и одна из другого блока внутренней памяти	–
5	–	3 из разных блоков внутренней памяти	–
6	–	2 из любой конфигурации памяти	1

А.9.5 Синхронизация доступа к памяти

Внутренние фазы синхронизации (H1 и H3) и их соотношения для организации доступа к памяти обсуждаются в данном разделе, чтобы показать, каким образом ПЦОС управляет несколькими обращениями к памяти. В то время как в предыдущем разделе обсуждается взаимодействие между последовательностями команд, здесь обсуждается поток данных на основе отдельной команды.

Каждый основной период синхронизации в 60 нс состоит из двух меньших периодов синхросигнала по 30 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.

A.9.5.1 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.

A.9.5.2 Загрузка и сохранение данных

Загрузку, чтение и сохранение данных выполняют 4 типа команд: двухоперандные команды, трёхоперандные команды, операции умножителя/АЛУ с командами сохранения и команды параллельного умножения и сложения. В разделе A.5 «Адресация» приведена более детальная информация о способах адресации.

Как описано в разделе A.7 «Работа с внешней шиной», 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

Двухоперандные команды включают все те команды, которые в разрядах 31-29 имеют значения 000 или 010. В случае чтения данных разряды 15-0 являются операндом src. Внутреннее чтение всегда производится в течение H1. Внешнее чтение данных всегда начинается в начале H3, с адресом, выставляемым на внешней шине, и завершается с защёлкиванием слова данных в конце H1.

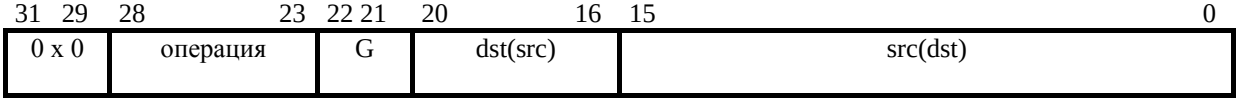


Рисунок A.9.2 – Слово двухоперандной команды

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.

Трёхоперандные команды включают все команды, у которых разряды 31-29 будут 001. Операнды источника src1 и src2 приходят либо из регистра, либо из памяти. Когда один или более операндов источника из памяти, тогда эти команды всегда производят чтение памяти.

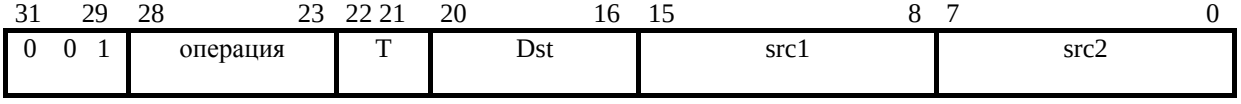


Рисунок A.9.3 – Слово трёхоперандной команды

Если только один из операндов источника из памяти (либо src1, либо src2) и расположен во внутренней памяти, то данные читаются 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 src1 и src2 оба 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.

Следующий тип команд включает все команды, которые производят параллельное сохранение и другую команду. Разряды 31 и 30 для этих команд равны 1 1.

Для тех операций, которые производят умножение или операции АЛУ параллельно с сохранением, формат слова команды приведён на рисунке А.9.4.

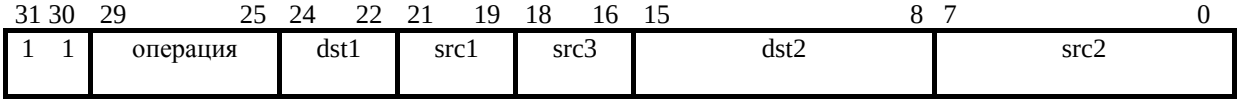


Рисунок А.9.4 – Умножение или операции с параллельным сохранением

Если операция сохранения для внешнего или внутреннего операнда назначения 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. Следует помнить, что чтение памяти выполняется ЦПУ в течение фазы чтения (R) конвейера и сохранение выполняется в течения фазы исполнения (E).

Формат слова команды для тех команд, в которых присутствуют параллельные сохранения в памяти, приведены на рисунке А.9.5.

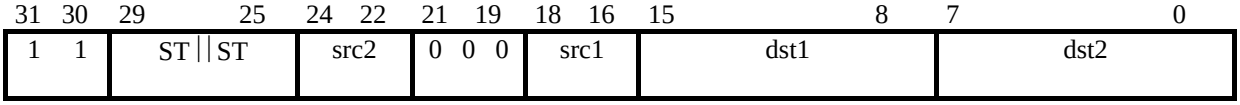


Рисунок А.9.5 – Два параллельных сохранения

Если оба операнда назначения, dst1 и dst2, расположены во внутренней памяти, dst1 сохраняется в течение H3, а dst2 в течение H1, завершая таким образом 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 два цикла шины, 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, то по-прежнему требуется только один цикл ЦПУ для завершения сохранения. В этом случае требуются 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.

Сохранение dst2 начинается в третьем цикле.

Сохранение dst2 завершается на четвёртом цикле внешней шины.

Параллельные умножения и сложения

Адресация памяти для параллельных умножений и сложений подобна адресации для трёхоперандных команд. Параллельное умножение и сложение включает все команды, разряды 31-30 которых равны 10.

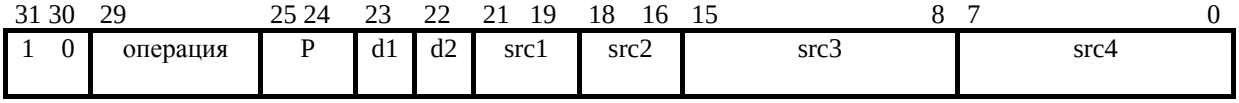


Рисунок A.9.6 – Параллельные умножения и сложения

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.

A.10 Команды языка ассемблер

Система команд языка ассемблер ПЦОС поддерживает высокопроизводительные вычислительные функции и может применяться как для цифровой обработки сигналов, так и для общего назначения. Команды объединены в основные группы, состоящие из команд загрузки и сохранения, двух- или трёхоперандных арифметико-логических, параллельных команд, а также команд программного управления и блокировки (управления межпроцессорным взаимодействием). Режимы адресации, используемые в данных командах, описаны в разделе A.5 «Адресация».

Система команд языка ассемблер ПЦОС может также использовать один из 20 кодов условий с одной из 10 условных команд, таких как LDF cond. Данный раздел описывает коды условий и флаги.

Ассемблер имеет дополнительные синтаксические формы для упрощения языка ассемблер для специальных команд. Список дополнительных форм с описанием приводится.

Каждая отдельная команда описана и приведена в алфавитном порядке. Пример команды демонстрирует специальный использованный формат и объясняет его контекст.

В данном разделе рассматриваются следующие основные положения:

- система команд (подраздел A.10.1);
- команды загрузки и сохранения;
- двухоперандные арифметико-логические команды;
- трёхоперандные арифметико-логические команды;
- команды программного управления;
- команды управления блокировкой;
- команды параллельных операций;
- коды условий и флаги (подраздел A.10.2);
- отдельные команды (подраздел A.10.3);
- символы и аббревиатура, используемые в командах;
- дополнительный синтаксис ассемблера;
- описание отдельных команд в алфавитном порядке, включая:
- синтаксис;
- описание работы;
- операнды;
- код;
- описание;
- количество циклов выполнения;
- разряды состояния;
- режимный разряд;
- примеры.

A.10.1 Система команд языка ассемблер

Система команд ПЦОС максимально подходит для решения задач цифровой обработки сигналов и других приложений, требующих высокопроизводительных вычислений. Все команды имеют длину в одно машинное слово, и большинство команд выполняется за один цикл. В дополнение к командам умножения и накопления, ПЦОС обладает развитой системой команд общего назначения.

Система команд содержит 113 команд, организованных в следующие функциональные группы:

- команды загрузки и сохранения;
- двухоперандные арифметико-логические команды;
- трёхоперандные арифметико-логические команды;
- команды программного управления;
- команды управления блокировкой;
- команды параллельных операций.

Каждая из этих групп обсуждается ниже.

A.10.1.1 Команды загрузки и сохранения

ПЦОС поддерживает 12 команд загрузки и сохранения (см. таблицу A.10.1).
Эти команды могут:

- загрузить слово из памяти в регистр;
- сохранить слово из регистра в памяти;
- манипулировать данными в системном стеке.

Две из этих команд могут загружать данные по условию. Это используется для нахождения минимального или максимального значения из набора данных. См. подраздел A.10.2 с детальным описанием кодов условий.

Таблица A.10.1 – Команды загрузки и сохранения

Команда	Описание	Команда	Описание
LDE	Загружает экспоненту с ПЗ	POP	Выталкивает целое из стека
LDF	Загружает значение с ПЗ	POPF	Выталкивает значение с ПЗ
LDFcond	Загружает значение с ПЗ из стека по условию	PUSH	Загружает целое в стек
LDI	Загружает целое	PUSHF	Загружает значение с ПЗ в стек
LDIcond	Загружает целое по условию	STF	Сохраняет значение с ПЗ
LDM	Загружает мантиссу с ПЗ	STI	Сохраняет целое

A.10.1.2 Двухоперандные команды

ПЦОС поддерживает полную систему из 35 двухоперандных арифметических и логических команд. Два операнда являются исходным операндом и операндом результата. Исходным операндом может быть слово памяти, регистр или часть слова команды. Операнд результата – всегда регистр. Эти команды обеспечивают целочисленную арифметику, арифметику с ПЗ или логические операции и арифметику повышенной точности. В таблице A.10.2 приведено краткое описание двухоперандных команд.

Таблица А.10.2 – Двухоперандные команды

Команда	Описание	Команда	Описание
ABSF	Абсолютное значение числа с ПЗ	NORM	Нормализовать значение с ПЗ
ABSI	Абсолютное значение целого	NOT	Поразрядное логическое дополнение
ADDC*	Сложить целое с переносом	OR*	Поразрядное логическое ИЛИ
ADDF*	Сложить значение с ПЗ	RND	Округлить значение с ПЗ
ADDI*	Сложить целые	ROL	Циклический сдвиг влево
AND*	Поразрядное логическое И	ROLC	Левый циклический сдвиг с переносом
ANDN*	Поразрядное логическое И с дополнением	ROR	Циклический сдвиг вправо
ASH*	Арифметический сдвиг	RORC	Циклический сдвиг вправо через перенос
CMPF*	Сравнить значения с ПЗ	SUBB*	Вычитание целых с заёмом
CMPI*	Сравнить целые	SUBC	Вычитание целых с условием
FIX	Преобразовать ПЗ в целое	SUBF	Вычитание значений с ПЗ
FLOAT	Преобразовать целое в ПЗ	SUBI	Вычесть целое
LSH	Логический сдвиг	SUBRB	Вычесть обратное целое с заёмом
MPYF*	Умножить значения с ПЗ	SUBRF	Вычесть обратное ПЗ с заёмом
MPYI*	Умножить целые	SUBRI	Вычесть обратное целое
NEGB	Отрицание целого с заёмом	TSTB*	Тестировать разрядные поля
NEGF	Отрицание числа в формате с ПЗ	XOR*	Поразрядное исключающее ИЛИ
NEGI	Отрицание целого		

* Двух- и трёхоперандные версии команд.

А.10.1.3 Трёхоперандные команды

Большинство команд имеет только два операнда, однако, некоторые арифметические и логические команды имеют трёхоперандные версии. 17 трёхоперандных команд позволяют ПЦОС считывать два операнда из памяти или регистрового файла ЦПУ в один цикл и сохранять результаты в регистре. Ниже описываются различия между двух- и трёхоперандными командами:

- двухоперандные команды имеют один исходный операнд (или сдвиг счётчика) и один операнд результата;

- трёхоперандные команды могут иметь два исходных операнда (или один исходный операнд и операнд счётчика) и один операнд результата. Исходным операндом является слово памяти или регистр. Операнд результата в трёхоперандных командах всегда регистр.

В таблице А.10.3 приведён список трёхоперандных команд. Необходимо отметить, что суффикс 3 может быть опущен в мнемониках трёхоперандных команд (см. А.10.3.2).

Таблица А.10.3 – Трёхоперандные команды

Команда	Описание	Команда	Описание
ADDC3	Сложение с переносом	MPYF3	Умножить значения с ПЗ
ADDF3	Сложить значения с ПЗ	MPYI3	Умножить целые
ADDI3	Сложить целые	OR3	Поразрядное логическое ИЛИ
AND3	Поразрядное логическое И	SUBB3	Вычитание целых с заёмом
AND3N	Поразрядное логическое И с дополнением	SUBF3	Вычитание значений с ПЗ
ASH3	Арифметический сдвиг	SUBI3	Вычитание целых
CMPF3	Сравнить значения с ПЗ	TSTB3	Тестирование разрядных полей
CMPI3	Сравнить целые	XOR3	Поразрядное ИСКЛЮЧАЮЩЕЕ ИЛИ
LSH3	Логический сдвиг		

А.10.1.4 Команды программного управления

Группа команд программного управления состоит из 16 команд, влияющих на выполнение программы. Режим повторения обеспечивает повторение блока команд (RPTB) или отдельной команды (RPTS). Поддерживаются как стандартные, так и задержанные (одноцикловые) переходы. Некоторые из команд программного управления могут зависеть от кодов условий (см. подраздел А.10.2 для получения более детальной информации о кодах условий).

В таблице А.10.4 приведены команды программного управления.

Таблица А.10.4 – Команды программного управления

Команда	Описание	Команда	Описание
Bcond	Переход по условию (стандартный)	IACK	Подтверждение прерывания
BcondD	Переход по условию (задержанный)	IDLE	Холостая работа до прерывания
BR	Безусловный переход (стандартный)	NOP	Нет операции
BRD	Безусловный переход (задержанный)	RETIcond	Возврат из прерывания по условию
CALL	Вызов подпрограммы	RETScond	Возврат из подпрограммы по условию
CALLcond	Вызов подпрограммы по условию	RPTB	Повтор блока команд
DBcond	Декремент и переход по условию (стандартный)	RPTS	Повтор отдельной команды
DBcondD	Декремент и переход по условию (задержанный)	SWI	Программное прерывание
		TRAPcond	Условное системное прерывание

А.10.1.5 Команды операций блокировки

Команды операций блокировки поддерживают мультипроцессорные связи и используют внешние сигналы для обеспечения мощного механизма синхронизации. Они также гарантируют целостность связи и результатов в высокоскоростных операциях, см. в разделе А.6 «Управление выполнением программы» примеры использования команд блокировки.

Таблица А.10.5 – Команды операций блокировки

Команда	Описание	Команда	Описание
LDFI	Загрузить значение с ПЗ, с блокировкой	STFI	Сохранить значение с ПЗ, с блокировкой
LDII	Загрузить целое, с блокировкой		
SIGI	Сигнализация с блокировкой	STII	Сохранить целое, с блокировкой

A.10.1.6 Команды параллельных операций

Группа команд параллельных операций делает возможным высокую степень параллелизма. Некоторые команды ПЦОС могут объединяться парами, которые выполняются параллельно. Данные команды обеспечивают следующие возможности:

- параллельная загрузка регистров,
- параллельные арифметические операции,
- арифметико-логические команды, выполняемые параллельно с командой сохранения.

Каждая команда в паре вводится как отдельный исходный оператор. Вторая команда в паре должна быть отделена двумя вертикальными чёрточками (||). В таблице А.10.6 приведён список параллельных команд.

Таблица А.10.6 – Параллельные команды

Мнемоника	Описание
Команды 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
NEG||STI	Обратное целое и сохранить целое
NOT||STI	Логическое дополнение (поразрядная инверсия) значения и сохранить целое
OR3||STI	Поразрядное логическое ИЛИ и сохранить целое
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
TOIEEE||STF	Преобразование в IEEE формат и сохранение
SUBI3||STI	Вычесть целое и сохранить целое
XOR3||STI	Поразрядное ИСКЛЮЧАЮЩЕЕ ИЛИ значений и сохранить целое
Команды параллельной загрузки	
LDF||LDF	Загрузить значение в формате с ПЗ
LDI||LDI	Загрузить целое
Команды 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

A.10.2 Коды условий и флаги

В ПЦОС используются 20 кодов условий (с 00000 по 10100, за исключением 01011) которые могут быть использованы с условными командами, 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 команды могут включать суффикс U для обозначения безусловной операции.

Семь флагов условий содержат информацию о свойствах результата арифметических или логических команд. Флаги условий хранятся в регистре состояния (ST) и изменяются командой в одном из следующих двух случаев:

- регистр назначения является одним из регистров с повышенной точностью (R7-R0). Это делается для модификации регистров, используемых для адресации, но не влияет на флаги условий в ходе вычисления;

- команда является одной из команд сравнения (CMPF, CMPF3, CMPI, CMPI3, TSTB или TSTB3). Это даёт возможность установить флаги условий в соответствии с содержимым регистров ЦПУ.

Флаги условий могут изменяться многими командами либо когда предшествующие условия установлены, либо при выполнении следующих условий:

- результат получен, когда определённая операция выполнена с бесконечной точностью. Это подходит для случая команд сравнения и тестирования, которые не сохраняют результат в регистре. Это также может подходить для арифметических команд, которые приводят к переполнению или отрицательному переполнению (потере значимости).

Более детальное описание влияния отдельных команд на флаги условий см. A.10.3.3.

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., $-2^{32} \leq$ результат $\leq 2^{32}-1$). 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 команд сдвига этот флаг устанавливается по значению последнего сдвигаемого вовне разряда; при нулевом значении сдвига флаг устанавливается в ноль.

В таблице А.10.7 приведена мнемоника условия, код, описание и флаг для каждого из 19 кодов условий.

Таблица А.10.7 – Коды условий и флаги

Условие	Код	Описание	Флаг*
1	2	3	4
Безусловные сравнения			
U	00000	Безусловный	Безусловное
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
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
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
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

Окончание таблицы А.10.7

1	2	3	4
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
* Знак ~ означает логическое дополнение (состояние «ложь»).			

А.10.3 Отдельные команды

Этот подраздел содержит отдельные команды языка ассемблер для ПЦОС. Команды приведены в алфавитном порядке. Информация о каждой команде включает синтаксис ассемблера, производимую операцию, операнды, код, описание, количество циклов, разряды состояния, режимные разряды и примеры.

Определение символов и аббревиатур также, как и дополнительных синтаксических форм, поддерживаемых ассемблером, находится в начале раздела описания отдельных команд. Также приводится пример команды с описанием используемого специального формата и объясняется её контекст.

Группировка по функциям команд и полный список команд приведены выше в подразделе А.10.1. Информация по режимам адресации содержится в разделе А.6 «Управление выполнением программы».

А.10.3.1 Символы и аббревиатура

В таблице А.10.8 приведены символы и аббревиатура, использованные в описании отдельных команд.

Таблица А.10.8 – Символы и аббревиатура, использованные в описании команд

Обозначение	Назначение
1	2
src	Операнд источника (исходный)
src1	Операнд источника 1 (исходный)
src2	Операнд источника 2 (исходный)
src3	Операнд источника 3 (исходный)
src4	Операнд источника 4 (исходный)
dst	Операнд назначения (результата)
dst1	Операнд назначения 1 (результата)
dst2	Операнд назначения 2 (результата)
disp	Смещение
cond	Условие
count	Счётчик сдвига

Окончание таблицы А.10.8

1	2
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 команд
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 разряд
x >> y	Сдвиг x справа на y разряд
*++SP	Инкрементировать SP и использовать инкрементированный SP как адрес
*SP– –	Использовать SP как адрес и декрементировать SP

А.10.3.2 Дополнительный синтаксис ассемблера

Ассемблер позволяет использовать упрощённую форму записи некоторых команд. Эти дополнительные формы упрощают синтаксис ассемблера, т. к. специфические формы синтаксиса могут игнорироваться.

Регистр назначения может быть опущен в одинарных арифметических и логических операциях, когда тот же самый регистр используется в качестве источника.

Например, ABSI R0, R0 может быть записана как ABSI R0.

Используемые команды: ABSI, ABSF, FIX, FLOAT, NEGB, NEGF, NEGI, NORM, NOT, RND.

Все трёхоперандные команды могут быть записаны без суффикса 3.

Например, ADDI3 R0, R1, R2 может быть записана как ADDI R0, R1, R2.

Используемые команды: ADDC3, ADDF3, ADDI3, AND3, ANDN3, ASH3, LSH3, MPYF3, MPYI3, OR3, SUBB3, SUBF3, SUBI3, XOR3.

Все трёхоперандные команды сравнения могут быть записаны без суффикса 3.

Например, CMPI3 R0, *AR0 может быть записана как CMPI R0, *AR0.

Используемые команды: CMPI3, CMPF3, TSTB3.

Разрешены непрямые операнды с явно заданным нулевым смещением. В трёхоперандных или параллельных командах операнды с нулевым смещением автоматически преобразуются в режим с отсутствием смещения. Например, разрешено: LDI *+AR0 (0), R1.

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 команды включают в себя суффикс U для обозначения безусловной операции. Также, суффикс U может быть исключён из команды короткого безусловного перехода. Например, BU label (метка) может быть записана как B label (метка).

Метки могут быть записаны как с последующим символом (:) так и без него. Например:

label0: NOP

label1: NOP

label2: (метка относится к следующей строке).

Пустые выражения запрещены для указания смещения в косвенном режиме:

LDI *+AR0 (), R0 – запрещено.

Операнды длинного непосредственного режима (назначение операторов BR и CALL) могут быть написаны со знаком @:

BR метка может быть записана как BR @метка.

Псевдооперация LDP может быть использована для загрузки регистра (обычно DP) 8 младшими разрядами перемещаемого адреса следующим образом:

LDP addr, REG или LDP @addr, REG.

Знак @ является дополнительным. Если регистр назначения это DP, он может быть опущен в операнде. LDP генерирует команду LDI с непосредственным операндом и специальным типом смещения.

Параллельные команды могут быть написаны в другом порядке. Например,

ADDI может быть записано как STI

|| STI || ADDI

Символы (||), определяющие 2 часть параллельной команды, могут быть написаны в любом месте строки, начиная с 0 столбца. Например,

ADDI может быть записано как ADDI

|| STI || STI

Если второй операнд параллельной команды такой же, как и третий (регистр назначения), третий операнд может быть опущен. Т. е. это даёт возможность написания трёхоперандной команды, которая выглядит как нормальная двухоперандная команда. Например,

ADDI *AR0, R2, R2 может быть записана как ADD *AR0, R2, R2

|| MPYI *AR1, R0, R0 || MPYI *AR1, R0

Команды, для которых применимо вышесказанное: ADDI, ADDF, AND, MPYI, MPYF, OR, SUBI, SUBF, XOR (для параллельных команд, имеющих в качестве второго операнда регистр).

Все коммутируемые операнды в параллельных командах могут быть записаны в другом порядке. Например, часть параллельной команды ADDI может быть записана одним из двух способов:

ADDI *AR0, R1, R2 или ADDI R1, *AR0, R2

Описанное выше относится к параллельным командам, содержащим одну из перечисленных команд:

ADDI, ADDF, MPYI, MPYF, AND, OR, XOR.

A.10.3.3 Описание отдельных команд

Далее описана каждая команда ассемблера ПЦОС в алфавитном порядке.

Информация о каждой команде включает синтаксис ассемблера, производимую операцию, операнды, код, описание, количество циклов, разряды состояния, режимные разряды и примеры.

Таблица А.10.9 – Синтаксис регистров ЦПУ

Синтаксис ассемблера	Альтернативный синтаксис регистров	Назначение регистров
R0	R0	Регистр повышенной точности 0
R1	R1	Регистр повышенной точности 1
R2	R2	Регистр повышенной точности 2
R3	R3	Регистр повышенной точности 3
R4	R4	Регистр повышенной точности 4
R5	R5	Регистр повышенной точности 5
R6	R6	Регистр повышенной точности 6
R7	R7	Регистр повышенной точности 7
AR0	R8	Вспомогательный регистр 0
AR1	R9	Вспомогательный регистр 1
AR2	R10	Вспомогательный регистр 2
AR3	R11	Вспомогательный регистр 3
AR4	R12	Вспомогательный регистр 4
AR5	R13	Вспомогательный регистр 5
AR6	R14	Вспомогательный регистр 6
AR7	R15	Вспомогательный регистр 7
DP	R16	Указатель страницы данных
IR0	R17	Индексный регистр 0
IR1	R18	Индексный регистр 1
BK	R19	Регистр размера блока
SP	R20	Указатель системного стека
ST	R21	Регистр состояния
IE	R22	Регистр разрешения прерывания ЦПУ/ПДП
IF	R23	Флаги прерываний ЦПУ
IOF	R24	Флаги ввода-вывода
RS	R25	Регистр адреса начала повторения
RE	R26	Регистр адреса конца повторения
RC	R27	Счётчик повторений

Синтаксис: INST src, dst

или

INST1 src2, dst1

||INST2 src3, dst2

Каждая команда начинается с синтаксического выражения ассемблера. Метки могут размещаться либо до команды (мнемонического выражения) на той же строке, либо на предыдущей строке в первом столбце. Дополнительное поле комментария, которое завершает синтаксис, не включается в синтаксическое выражение. Пробелы требуются между всеми полями (метка, команда, операнд и поля комментария).

Примеры синтаксиса иллюстрируют общий одностроковый и двухстроковый синтаксис, используемый в параллельной адресации. Необходимо обратить внимание, что символ (||), обозначающий параллельную адресную пару, может быть размещён, где угодно перед мнемоникой во второй строке. Первая команда в паре может иметь метку, но вторая иметь метки не может.

Операция: |src| → dst

или

|src2| → dst1

|| src3 → dst2

Последовательность работы команды описывает процесс, который имеет место при выполнении команды. Для параллельных команд рабочая последовательность выполняется параллельно. Условные эффекты, определённые режимами регистра состояния, приводятся для условных команд, таких как Bcond.

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

или

src2 - косвенная (смещение = 0, 1, IR0, IR1)

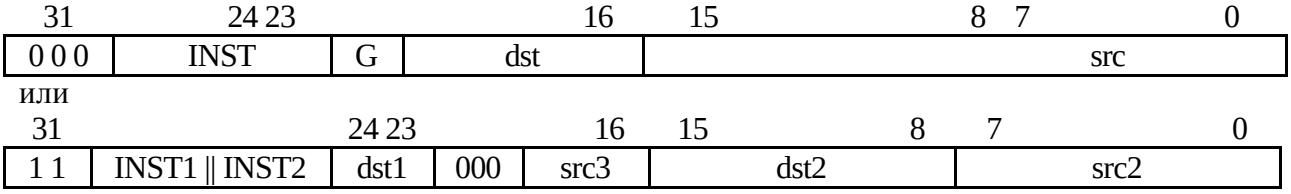
dst1 - регистр (Rn1, $0 \le n1 \le 7$)

src3 - регистр (Rn2, $0 \le n2 \le 7$)

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 (см. А.5.1.3).

Код:



Примеры кодов приведены с использованием основной и параллельной адресации. Пара команд для примера параллельной адресации состоит из INST1 и INST2.

Описание:

Описано выполнение команды и её влияние на состояние процессора и содержимое памяти. Описаны также ограничения, накладываемые на операнды процессором или ассемблером. Описание идёт параллельно и дополняет информацию, приведённую в описании работы.

Циклы: 1

Число описывает количество циклов, требуемых для выполнения команды.

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 возникновении отрицательного переполнения UF устанавливается в 1, в противном случае 0.

N – флаг отрицательного условия. Для некоторых операций N имеет значение старшего значащего разряда выходного значения; 1 – если результат отрицательный и 0 – в противном случае.

Z – флаг нулевого условия. 1 – при равенстве выходного результата нулю и 0 – в противном случае. Для логических команд и команд сдвига: 1 – при нулевом выходном результате, 0 – в противном случае.

V – флаг условия переполнения. 1 – при переполнении, в 0 – в противном случае.

C – устанавливается в 1 при возникновении заём или переноса, 0 – в противном случае. Для команд сдвига этот флаг устанавливается по значению последнего сдвигаемого вовне разряда; при нулевом сдвиге – устанавливается в ноль.

Семь флагов условий, сохраняемые в регистре состояния (ST), изменяются большинством команд только если регистр назначения представляет собой один из R7-R0. Они выдают 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 (описано в таблице А.3.2).

Пример: INST @98AEh, R5

До команды:

DP = 80h

R5 = 0766900000h = 2.30562500e+02

Значение в ячейке памяти по адресу 8098AEh = 5CDFh = 1.00001107e+00

LUF LV UF N Z V C = 0 0 0 0 0 0 0

После команды:

DP = 80h

R5 = 0066900000h = 1.80126953e+00

Значение в ячейке памяти по адресу 8098AEh = 5CDFh = 1.00001107e+00

LUF LV UF N Z V C = 0 0 0 0 0 0 0

Примерный код, представленный в вышеописанном формате, показывает влияние команды 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 ассемблере или симуляторе (см. таблицу А.10.7).

ABSF Абсолютное значение числа с плавающей запятой

Синтаксис: ABSF src, dst

Операция: |src| → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 7$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 7$)

Опкод:

31 29	28 23	22 21	20 16	15 0
0 0 0	0 0 0 0 0 0	G	dst	src

Описание: Абсолютное значение операнда src загружается в регистр dst. Операнды src и dst являются числами в формате с ПЗ.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 0

N 0

Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении ПЗ-переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример:

```
ABSF      R4,R7
```

Before Instruction			After Instruction		
R4	5C 8000 F971	-9.90337307e+27	R4	5C 8000 F971	-9.90337307e+27
R7	7D 2511 00AE	5.48527255e+37	R7	5C 7FFF 068F	9.90337307e+27
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	

ABSF||STF Абсолютное значение числа в формате с ПЗ и сохранить значение с ПЗ

Синтаксис: ABSF src2, dst1

|| STF src3, dst2

Операция: |src2| → dst1

|| src3 → dst2

Операнды: src2 - косвенная (смещение = 0, 1, IR0, IR1)

dst1 - регистр (Rn1, 0 ≤ n1 ≤ 7)

src3 - регистр (Rn2, 0 ≤ n2 ≤ 7)

dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 30	29 25	24 23	22 20	19 16	15 8	7 0
0 0	0 0 1 0 0	dst1	0 0 0	src3	dst1	src3

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 командой 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 меняется.

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)
```

	Before Instruction			After Instruction	
R4	07 33C0 0000	1.79750e+02	R4	05 74C0 0000	6.118750e+01
AR3	80 9800		AR3	8098AF	
AR7	80 98C5		AR7	8098C5	
IR1	0AF		IR1	0AF	
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	
Data Memory					
8098AF	58B4000	-6.118750e+01	8098AF	58B4000	-6.118750e+01
8098C4	0		8098C4	733C000	1.79750e+02

ABSI Абсолютное значение целого числа

Синтаксис: ABSI src, dst

Операция: |src| → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, 0 ≤ n ≤ 27)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 29	28 24	23	22 16	19 16	15 0
0 0 0	0 0 0 0 0 1	G	dst	src3	src

Описание: Абсолютное значение операнда src загружается в регистр dst. Операнды src и dst являются целыми со знаком.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 0

Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример 1:

```
ABSI    R0,R0
or
ABSI    R0
```

Before Instruction			After Instruction		
R0	00 FFFF FFCB	-53	R0	00 0000 0035	53

Пример 2:

```
ABSI    *AR1,R3
```

Before Instruction			After Instruction		
R3	00 0000 0000		R3	00 0000 0035	53
AR1	00 0020		AR1	00 0020	
Data memory					
20	0FFFFFFCB	-53	20	0FFFFFFCB	-53

ABSI||STI Абсолютное значение целого числа и сохранить целое

Синтаксис: ABSI src2, dst1
|| STI src3, dst2

Операция: |src2| → dst1
|| src3 → dst2

Операнды: src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn1, 0 ≤ n1 ≤ 7)
src3 - регистр (Rn2, 0 ≤ n2 ≤ 7)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 30	29 25	24 23	22 20	16	15 8	7 0
1 1	0 0 1 0 1	dst1	0 0 0	src3	dst1	src2

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 командой 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.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7-R0.
LUF Не изменяется.
LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF 0
N 0
Z 1 при генерации нулевого результата, иначе 0.
V 1 при возникновении целочисленного переполнения, иначе 0.
C Не изменяется.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример:

```
        ABSI    *-AR5(1),R5
||      STI     R1,*AR2--(IR1)
```

	Before Instruction		After Instruction	
R1	00 0000 0042	66	00 0000 0042	66
R5	00 0000 0000		00 0000 0035	53
AR2	80 98FF		80 98F0	
AR5	80 99E2		80 99E2	
IR1	0F		0F	
LUF	0		0	
LV	0		0	
UF	0		0	
N	0		0	
Z	0		0	
V	0		0	
C	0		0	
Data memory				
8098FF	2	-53	42	-53
8099E1	0FFFFFFCB	2	0FFFFFFCB	66

ADDC Сложение целых с переносом

Синтаксис: ADDC src, dst

Операция: dst + src + C → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 29	28 23	24 22	16	15 8	7 0
0 0 0	0 0 0 0 1 0	G	dst	src	

Описание: Сумма операндов dst и src и флаг C (перенос) загружается в регистр dst. Операнды src и dst являются целыми со знаком.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7-R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

C 1 при возникновении переноса, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример:

```
ADDC    R1,R5
```

Before Instruction			After Instruction		
R1	00 FFFF 5C25	-41,947	R1	00 FFFF 5C25	-41,947
R5	00 FFFF 019E	-65,122	R5	00 FFFE 5DC4	-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 Сложение целых с переносом (3 операнда)

Синтаксис: ADDC3 src2, src1, dst

Операция: src1 + src2 + C → dst

Операнды: Трёхоперандные режимы адресации src1 (T):
00 - регистровая (Rn1, 0 ≤ n1 ≤ 27)
01 - косвенная (смещение = 0, 1, IR0, IR1)
10 - регистровая (Rn1, 0 ≤ n1 ≤ 27)
1 - косвенная (смещение = 0, 1, IR0, IR1)
Трёхоперандные режимы адресации src2 (T):
00 - регистровая (Rn2, 0 ≤ n2 ≤ 27)
01 - косвенная (смещение = 0, 1, IR0, IR1)
10 - регистровая (Rn2, 0 ≤ n2 ≤ 27)
11 - косвенная (смещение = 0, 1, IR0, IR1)
dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 29	28 23	24	22 16	15 8	7 0
0 0 1	0 0 0 0 0 0	T	dst	src3	src

Описание: Сумма операндов src1 и src2, и флаг C (перенос) загружается в регистр dst. Операнды src1, src2 и dst являются целыми со знаком.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7-R0.
LUF Не изменяется.
LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF 0
N 1 при генерации отрицательного результата, иначе 0.
Z 1 при генерации нулевого результата, иначе 0.
V 1 при возникновении целочисленного переполнения, иначе 0.
C 1 при возникновении переноса, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример 1:

```
ADDC3     *AR5++(IR0),R5,R2
  or
ADDC3     R5,*AR5++(IR0),R2
```

Before Instruction			After Instruction		
R2	00 0000 0000		R2	00 0000 0032	50
R5	00 0000 0066	102	R5	00 0000 0066	102
AR5	80 9908		AR5	80 9918	
IR0	10		IR0	10	
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UF	0	
N	0		N	0	
Z	0		Z	0	
V	0		V	0	
C	1		C	1	
Data memory					
809908	0FFFFFCB	-53	809908	0FFFFFCB	-53

Пример 2:

```
ADDC3     R2, R7, R0
```

Before Instruction			After Instruction		
R0	00 0000 0000		R0	00 0000 123F	4671
R2	00 0000 02BC	700	R2	00 0000 02BC	700
R7	00 0000 0F82	3970	R7	00 0000 0F82	3970
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	

ADDF Сложение значений с плавающей запятой

Синтаксис: ADDF src, dst

Операция: dst + src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 7$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 7$)

Опкод:

31 29	28 23	24 22	16	15 8	7 0
0 0 0	0 0 0 0 1 1	G	dst	src	

Описание: Сумма операндов dst и src загружается в регистр dst.

Операнды src и dst – числа в формате с ПЗ.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7-R0.

LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 1 при возникновении отрицательного переполнения, иначе 0.

N 1 при генерации отрицательного результата, иначе 0.

Z 1 при генерации нулевого результата, иначе 0.

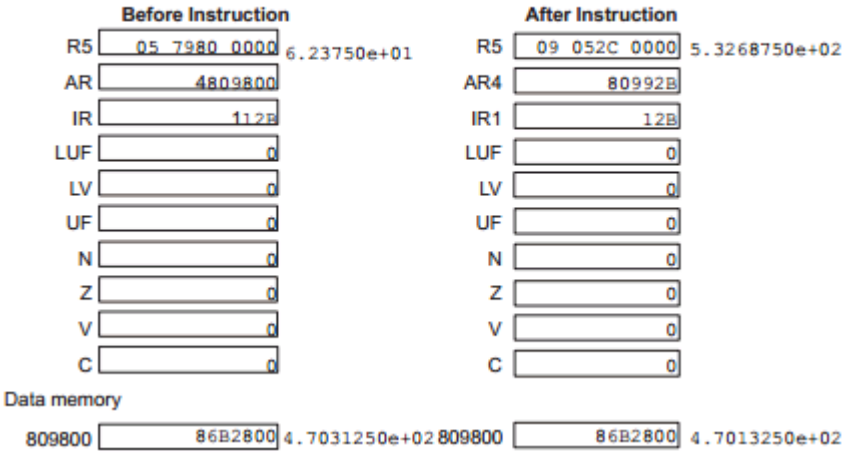
V 1 при возникновении ПЗ-переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример:

```
ADDF *AR4++(IR1),R5
```



ADDF3 Сложение значений с плавающей запятой (3 операнда)

Синтаксис: ADDF3 src2, src1, dst

Операция: src1 + src2 → dst

Операнды: Трёхоперандные режимы адресации src1 (T):

00 - регистровая (Rn1, 0 ≤ n1 ≤ 7)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn1, 0 ≤ n1 ≤ 7)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Трёхоперандные режимы адресации src2 (T):

00 - регистровая (Rn2, 0 ≤ n2 ≤ 7)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn2, 0 ≤ n2 ≤ 7)

11 - косвенная (смещение = 0, 1, IR0, IR1)

dst - регистр (Rn, 0 ≤ n ≤ 7)

Опкод:

31 29	28 23	24	22 16	15 8	7 0
0 0 1	0 0 0 0 0 1	T	dst	src1	src2

Описание: Сумма операндов src1 и src2 загружается в регистр dst. Операнды src1, src2 и dst являются числами в формате с ПЗ.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 1 при возникновении отрицательного переполнения, 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.

Пример 1:

```
ADDF3     R6,R5,R1
or
ADDF3     R5,R6,R1
```

	Before Instruction		After Instruction	
R1	00 0000 0000		09 052C 0000	5.3268750e+02
R5	05 7980 0000	6.23750e+01	05 7980 0000	6.23750e+01
R6	08 6B28 0000	4.7031250e+02	08 6B28 0000	4.7031250e+02
LUF	0		0	
LV	0		0	
UF	0		0	
N	0		0	
Z	0		0	
V	0		0	
C	0		0	

Пример 2:

```
ADDF3     *+AR1(1),*AR7++(IR0),R4
```

	Before Instruction		After Instruction		
R4	00 0000 0000		R4	07 0DB2 0000	1.41695313e+02
AR1	80 9820		AR1	80 9820	
AR7	80 99F0		AR7	80 99F8	
IR0	8		IR0	8	
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UV	0	
N	0		N	0	
Z	0		Z	0	
V	0		V	0	
C	0		C	0	
Data memory					
809821h	700F000	1.28940e+02	809821h	700F000	1.28940e+02
8099F0h	34C2000	1.27590e+01	8099F0h	34C2000	1.27590e+01

ADDF3||STF Сложить значения в формате с ПЗ и сохранить значение с ПЗ

Операция: src1 + src2 → dst1

|| src3 → dst2

Операнды: src1 - регистр (Rn1, 0 ≤ n1 ≤ 7)

src2 - косвенная (смещение = 0, 1, IR0, IR1)

dst1 - регистр (Rn2, 0 ≤ n2 ≤ 7)

src3 - регистр (Rn3, 0 ≤ n3 ≤ 7)

dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 30	29 24	23 16	15 8	7 0
0 0	0 0 1 1 0	dst1 src1	src3	dst2	src2

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 командой ADDF3. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если назначение – один из R7 - R0.

LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 1 при возникновении отрицательного переполнения, 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.

Пример

```
        ADDF3   *+AR3(IR1),R2,R5
||      STF     R4,*AR2
```

Before Instruction			After Instruction		
R2	07 0C80 0000	1.4050e+02	R2	07 0C80 0000	1.4050e+02
R4	05 7B40 0000	6.281250e+01	R4	05 7B40 0000	6.281250e+01
R5	00 0000 0000		R5	08 2020 0000	3.20250e+02
AR2	80 98F3		AR2	80 98F3	
AR3	80 9800		AR3	80 9800	
IR1	0A5		IR1	0A5	
LUF	0		LUF	0	
LV	0		LV	0	
UF	0		UV	0	
N	0		N	0	
Z	0		Z	0	
V	0		V	0	
C	0		C	0	
Data memory					
8098A5h	733C000	1.79750e+02	8098A5h	733C000	1.79750e+02
8098F3h	0		8098F3h	57B4000	6.28125e+01

ADDI Сложение целых

Синтаксис: ADDI src, dst

Операция: dst + src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 29	28 24	23 22	16	15 8	7 0
0 0 0	0 0 0 0 1 1	G	dst	src	

Описание: Сумма операндов dst и src загружается в регистр dst. Операнды dst и src являются целыми со знаком.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

C 1 при возникновении переноса, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример: ADDI R3, R7

Before Instruction			After Instruction		
R3	00 FFFF FFCB	-53	R3	00 FFFF FFCB	-53
R7	35	53	R7	00 0000 0000	
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	

ADDI3 Сложение целых (3 операнда)

Синтаксис: ADDI3 <src2>, <src1>, <dst>

Операция: src1 + src2 → dst

Операнды: Трёхоперандные режимы адресации src1 (T):
00 - регистровая (Rn1, 0 ≤ n1 ≤ 27)
01 - косвенная (смещение = 0, 1, IR0, IR1)
10 - регистровая (Rn1, 0 ≤ n1 ≤ 27)
11 - косвенная (смещение = 0, 1, IR0, IR1)
Трёхоперандные режимы адресации src2 (T):
00 - регистровая (Rn2, 0 ≤ n2 ≤ 27)
01 - косвенная (смещение = 0, 1, IR0, IR1)
10 - регистровая (Rn2, 0 ≤ n2 ≤ 27)
11 - косвенная (смещение = 0, 1, IR0, IR1)
dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 29	28 24	23	16	15 8	7 0
0 0 1	0 0 0 0 1 0	T	dst	src1	src2

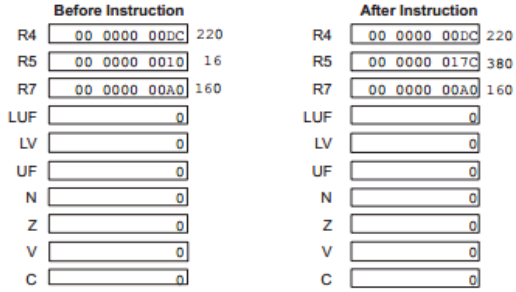
Описание: Сумма операндов src1 и src2 загружается в регистр dst. Операнды src1, src2 и dst являются целыми со знаком.

Циклы: 1

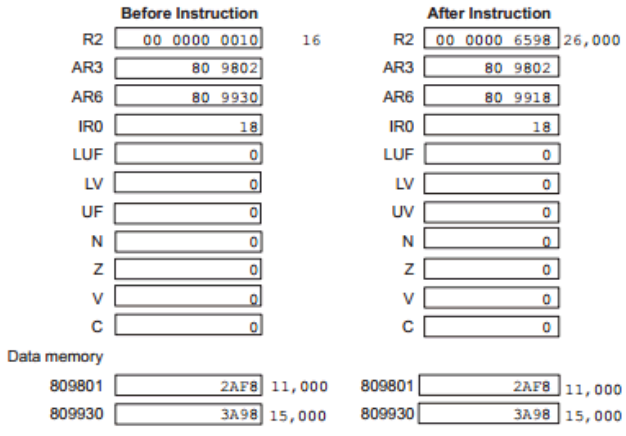
Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7 - R0.
LUF Не изменяется.
LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF 0
N 1 при генерации отрицательного результата, иначе 0.
Z 1 при генерации нулевого результата, иначе 0.
V 1 при возникновении целочисленного переполнения, иначе 0.
C 1 при возникновении переноса, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример 1: ADDI3 R4, R7, R5



Пример 2: ADDI3 * – AR3(1), * – – AR6(IR0), R2



ADDI3||STI Сложить целые и сохранить целое

Синтаксис: ADDI3 src2, src1, dst1
|| STI src3, dst2

Операция: src1 + src2 → dst1
|| src3 → dst2

Операнды: src1 - регистр (Rn1, 0 ≤ n1 ≤ 7)
src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn2, 0 ≤ n2 ≤ 7)
src3 - регистр (Rn3, 0 ≤ n3 ≤ 7)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод

31 24	23 16	15 8	7 0
0 1 | 0 0 1 1 1	dst1 | src1	src3	dst2 | src2

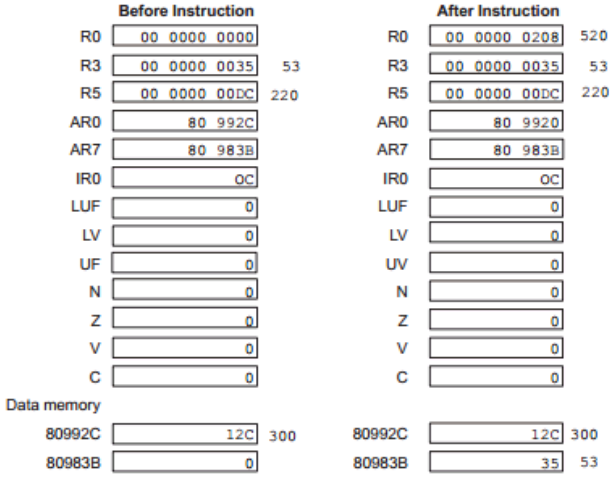
Описание: Целочисленное сложение и сохранение целого производится параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STI) считывает из регистра, и операция, которая производится параллельно (ABSI), записывает в тот же регистр, STI имеет на входе содержимое регистра до того, как оно было изменено командой ABSI. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если назначение – один из R7 - R0.
LUF Не изменяется.
LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF 0
N 1 при генерации отрицательного результата, иначе 0.
Z 1 при генерации нулевого результата, иначе 0.
V 1 при возникновении целочисленного переполнения, иначе 0.
C 1 при возникновении переноса, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример
```
ADDI3   *AR0 –(IR0), R5, R0
|| STI   R3, *AR7
```



AND Поразрядное логическое И (AND)

Синтаксис: AND src, dst

Операция: dst AND src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, 0 ≤ n ≤ 27)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 29	28 24	23 16	15 8	7 0
0 0 0	0 0 0 1 0 1	G	dst	src

Описание: Результат поразрядного логического И между операндами dst и src записывается в регистр dst. Предполагается, что src и dst являются целыми без знака.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

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.

Пример: AND R1, R2

Before Instruction	
R1	00 0000 0080
R2	00 0000 0AFF
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	1

After Instruction	
R1	00 0000 0080
R2	00 0000 0080
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	1

AND3 Поразрядное логическое И (AND) (3 операнда)

Синтаксис: AND3 src2, src1, dst

Операция: src1 AND src2 → dst

Операнды: Трёхоперандные режимы адресации src1 (T):

00 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Трёхоперандные режимы адресации src2 (T):

00 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

11 - косвенная (смещение = 0, 1, IR0, IR1)

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 29	28 24	23 16	15 8	7 0
0 0 1	0 0 0 0 1 1	T	dst	src1	src2

Описание: Результат поразрядного логического И операндов src1 и src2 загружается в регистр dst.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если назначение – один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N Старший значащий разряд выходного значения.

Z 1 при генерации нулевого результата, иначе 0.

V 0

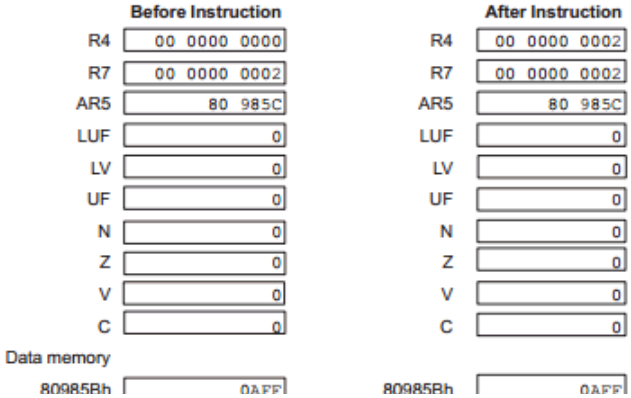
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример 1: AND3 *AR0– – (IR0), *+AR1, R4

	Before Instruction	After Instruction
R4	00 0000 0000	00 0000 0020
AR0	80 98F4	80 98A4
AR1	80 9951	80 9951
IR0	50	50
LUF	0	0
LV	0	0
UF	0	0
N	0	0
Z	0	0
V	0	0
C	0	0
Data memory		
8098F4h	30	30
809952h	123	123

Пример 2: AND3 *-AR5, R7, R4



AND3||STI Поразрядное логическое И и сохранить целое

Синтаксис: AND src2, src1, dst1
||STI src3, dst2

Операция: src1 AND src2 → dst1
|| src3 → dst2

Операнды: src1 - регистр (Rn1, 0 ≤ n1 ≤ 7)
src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn2, 0 ≤ n2 ≤ 7)
src3 - регистр (Rn3, 0 ≤ n3 ≤ 7)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод

31 24	23 16	15 8	7 0
0 1 | 0 0 1 1 1 | dst1	src1 | src3	dst2	src2

Описание: Поразрядное логическое И и сохранение целого параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STI) считывает из регистра, и операция, которая производится параллельно (AND3), записывает в тот же регистр, STI имеет на входе содержимое регистра до того, как оно было изменено командой AND3. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

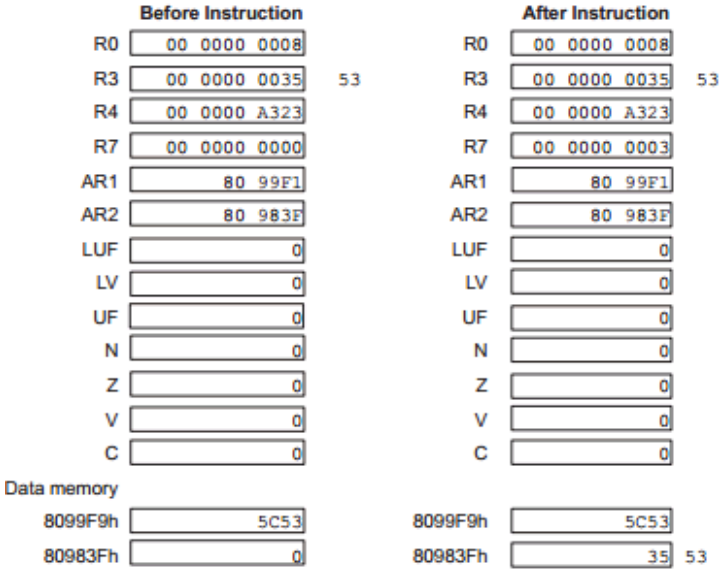
Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если назначение – один из R7 - R0.
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



ANDN Поразрядное логическое И (AND) с дополнением

Синтаксис: ANDN src, dst

Операция: dst AND ~src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, 0 ≤ n ≤ 27)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод

31 24	23 16	15 8	7 0
0 0 0 0 0 0 0 1 0 G	dst	src	

Описание: Поразрядное логическое И между операндом dst и поразрядным логическим дополнением (~) операнда src (→) загружается в регистр dst. Предполагается, что операнды dst и src являются беззнаковыми целыми.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N Старший значащий разряд выходного результата.

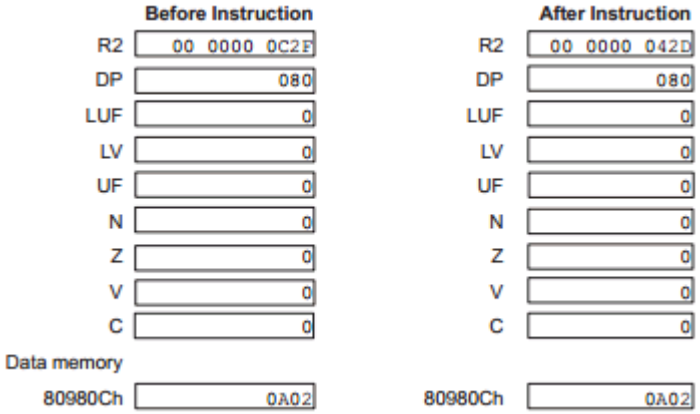
Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: ANDN @980Ch, R2



ANDN3 Поразрядное логическое И (AND) (с дополнением (3 операнда))

Синтаксис: ANDN3 src2, src1, dst

Операция: src1 AND ~src2 → dst

Операнды: Трёхоперандные режимы адресации src1 (T):

00 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Трёхоперандные режимы адресации src2 (T):

00 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

11 - косвенная (смещение = 0, 1, IR0, IR1)

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 29	28 24	23 16	15 8	7 0
0 0 1	0 0 0 1 0 0	T	dst	src1	src2

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.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения- один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N Старший значащий разряд выходного результата.

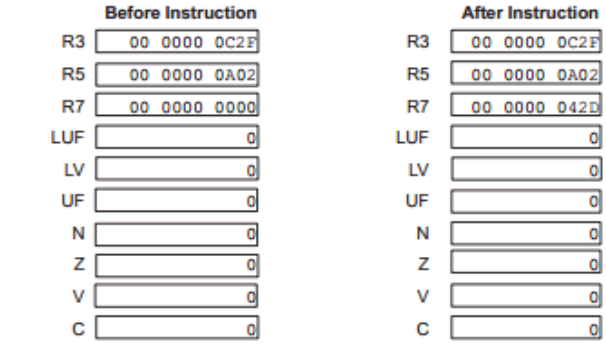
Z 1 при генерации нулевого результата, иначе 0.

V 0

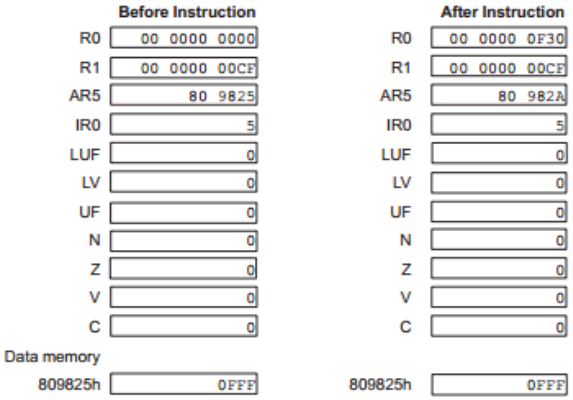
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример 1: ANDN3 R5, R3, R7



Пример 2: ANDN3 R1, *AR5++ (IR0), R0



ASH Арифметический сдвиг

Синтаксис: ASH count, dst

Операция: Если (count ≥0):

dst << count → dst.

В противном случае:

dst >> |count| → dst.

Операнды: основные режимы адресации count (по счётчику) (G):

00 - регистровая (Rn, 0 ≤ n ≤ 27)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод

31 24	23 16	15 8	7 0
0 0 0 0 0 0 1 1 1	G	dst	count

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 в разряд переноса C регистра состояния. Арифметический левый

сдвиг: C← dst ← 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 в разряд переноса C регистра состояния. Арифметический сдвиг право: знак dst → dst → C. Если операнд count равен 0, сдвиг не происходит, и разряд C (перенос) устанавливается в 0. Операнды count и dst являются целыми со знаком.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N Старший значащий разряд выходного значения.

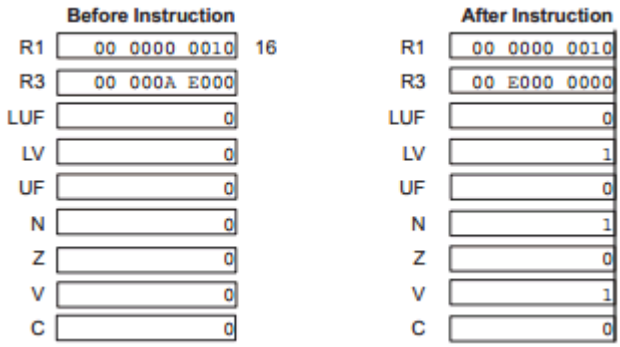
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

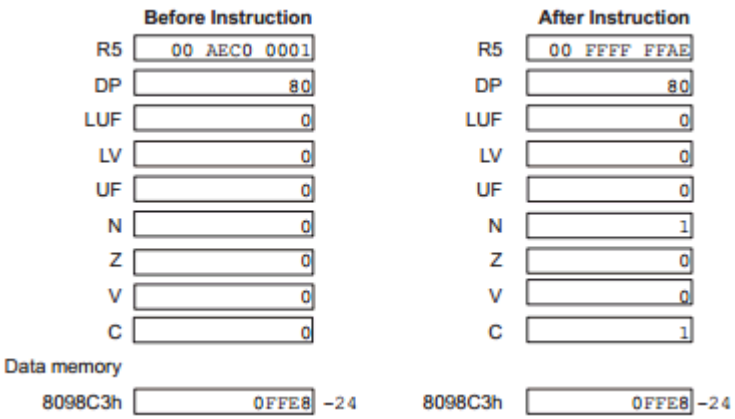
C Устанавливается по значению последнего сдвигаемого вовне разряда. Равен 0, если счётчик сдвига 0.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример 1: ASH R1, R3



Пример 2: ASH @98C3h, R5



ASH3 Арифметический сдвиг (3 операнда)

Синтаксис: ASH3 count, src, dst

Операция: Если (count ≥ 0):

src << count → dst

В противном случае:

src >> |count| → dst

Операнды: Трёхоперандные режимы адресации count (T):

00 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

01 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

10 - косвенная (смещение = 0, 1, IR0, IR1)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Трёхоперандные режимы адресации src (T):

00 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

11 - косвенная (смещение = 0, 1, IR0, IR1)

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 29	28 24	23 16	15 8	7 0
0 0 1	0 0 0 1 0 1 T	dst	src	count

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 в разряд переноса C регистра состояния. Арифметический левый сдвиг: C ← src ← 0.

Если операнд count меньше 0, операнд src сдвигается справа на число разрядов, определённых абсолютным значением операнда count. Старшие разряды операнда src сдвигаются вправо с расширением знака. Младшие разряды сдвигаются в разряд переноса C регистра состояния.

Арифметический сдвиг вправо: знак src → src → C.

Если операнд count равен 0, сдвиг не происходит, и разряд C (перенос) устанавливается в 0. Операнды count, src и dst являются целыми со знаком.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, иначе не изменяется.

UF 0

N Старший значащий разряд выходного результата.

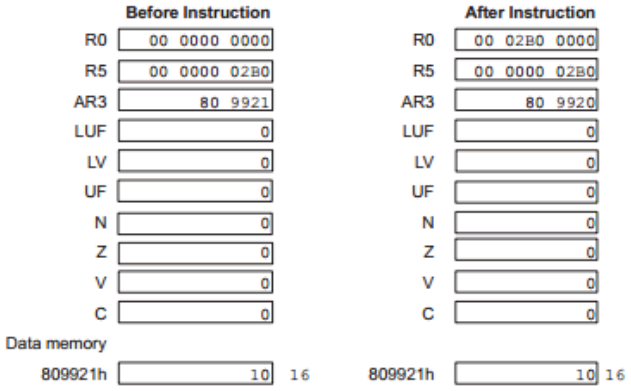
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

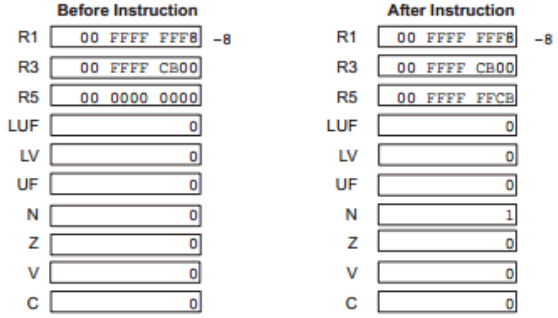
C Устанавливается по значению последнего сдвигаемого вовне разряда. Равен 0, если count = 0.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример 1: ASH3 *AR3– – (1), R5, R0



Пример 2: ASH3 R1, R3, R5



ASH3||STI Арифметический сдвиг и целое

Синтаксис: ASH3 count, src2, dst1
|| STI src3, dst2

Операция: Если (count ≥ 0):
src2 << count → dst1
В противном случае:
src2 >> |count| → dst1
|| src3 → dst2

Операнды: count - регистр (Rn1, 0 ≤ n1 ≤ 7)
src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn2, 0 ≤ n2 ≤ 7)
src3 - регистр (Rn3, 0 ≤ n3 ≤ 7)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод

31 24	23			16 15 8	7 0
1 1	0 1 0 0 1	dst1	Count	src3	dst2	src2

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 в разряд переноса C регистра состояния. Арифметический левый сдвиг: C←src2←0. Если операнд count

меньше 0, операнд src сдвигается справа на число разрядов, определённых абсолютным значением операнда count. Старшие разряды операнда src сдвигаются вправо с расширением знака. Младшие разряды сдвигаются в разряд переноса C регистра состояния. Арифметический сдвиг вправо: знак src2 → src2 → 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 командой ASH3. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7-R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, иначе не изменяется.

UF 0

N Старший значащий разряд выходного результата

Z 1 при генерации нулевого результата, иначе 0.

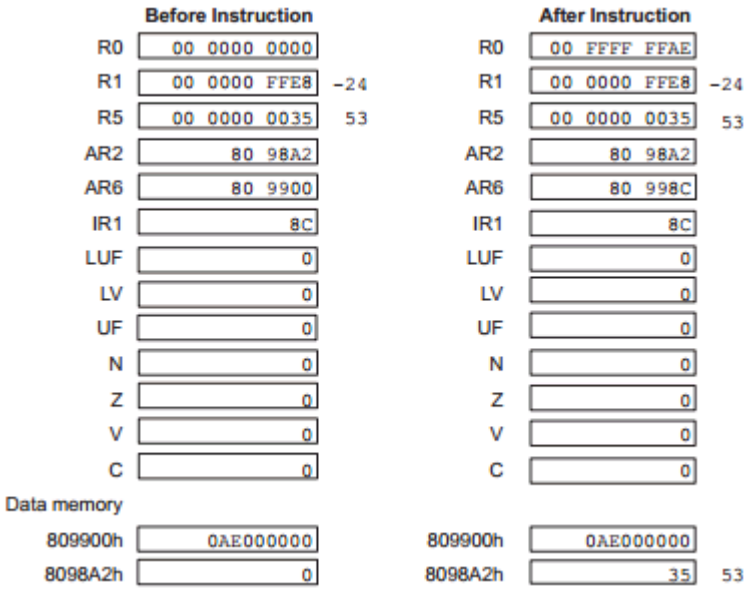
V 1 при возникновении целочисленного переполнения, иначе 0.

C Устанавливается по значению последнего сдвигаемого вовне разряда. Равен 0, если count = 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример:

```
ASH3 R1, *AR6++ (IR1), R0
|| STI R5, *AR2
```



Bcond Условный переход (стандартный)

Синтаксис: Bcond src

Операция: Если cond - истина: Если src в регистровом режиме адресации (Rn, 0≤n≤27), src → PC.
Если src в режиме адресации относительно PC (метка или адрес), смещение + PC + 1 → PC. В противном случае продолжение.

Операнды: Режимы адресации условного перехода src (B):
0 - регистровая
1 - относительно PC

Опкод:

31 24	23 16	15 8	7 0
0 1 1 0 1 0 B 0 0 0 0	cond	регистр или смещение	

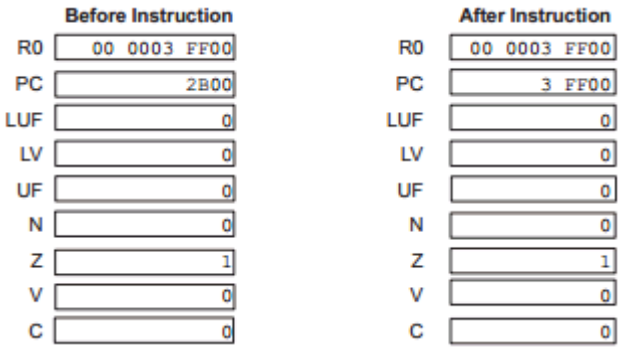
Описание: Bcond это стандартный переход, который выполняется за четыре цикла. Переход осуществляется, если условие истинно (т. к. конвейер при этом должен быть очищен; см. подраздел A.9.2). Если операнд src установлен в режим регистровой адресации, содержимое указанного регистра загружается в PC. Если операнд src установлен в режим адресации относительно PC, ассемблер генерирует смещение;
смещение = метка - (PC команды перехода + 1).
Это смещение сохраняется как 16-разрядное целое со знаком в 16 младших значащих разрядах слова команды перехода. Смещение добавляется к PC команды перехода + 1 для генерации нового PC.
В ПЦОС имеются 20 кодов условий, которые могут быть использованы с этой команды (в подразделе A.10.2 приведён список мнемоник условий, коды и флаги). Флаги условий устанавливаются предыдущей командой, только когда регистром назначения является один из регистров повышенной точности (R7-R0), либо когда выполняется одна из команд сравнения (CMPF, CMPF3, CMPI, CMPI3, TSTB или TSTB3).

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: BZ R0



BcondD Условный переход (с задержкой)

Синтаксис: BcondD src

Операция: Если cond - истина:
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.

Операнды: Режимы адресации условного перехода src (B):
0 – регистровая
1 - относительно PC

Опкод:

31 24	23	22 16	15 8	7 0
0 1 1 0 1 0	B	0 0 0	1	cond	регистр или смещение

Описание: BcondD это задержанный переход, при котором обеспечивается выборка трёх команд до изменения PC. В результате получается одноцикловый переход, и три команды, следующие за BcondD, не влияют на cond. 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 команды перехода + 3). Это смещение сохраняется как 16-разрядное целое со знаком в 16 младших значащих разрядах слова команды перехода. Смещение добавляется к PC команды перехода + 3 для генерации нового PC. В ПЦОС имеются 20 кодов условий, которые могут быть использованы с этой команды (в подразделе A.10.2 приведён список мнемоник условий, коды и флаги). Флаги условий устанавливаются предыдущей командой, только когда регистром назначения является один из регистров повышенной точности (R7-R0), либо когда выполняется одна из команд сравнения (CMPF, CMPF3, CMPI, CMPI3, TSTB или TSTB3).

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: BNZD 36 (36 = 24h)

Before Instruction		After Instruction	
PC	0050	PC	0077
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

BR Безусловный переход (стандартный)

Синтаксис: BR src

Операция: src → PC

Операнды: src - режим длинной непосредственной адресации

Опкод:

31 24	23 16	15 8	7 0
0 1 1 0 0 0 0	0		src

Описание: BR обеспечивает стандартный переход, который выполняется за четыре цикла, т. к. в течение выполнения перехода происходит очистка конвейера (см. подраздел A.9.2). Операнд src является 24-разрядным целым без знака.

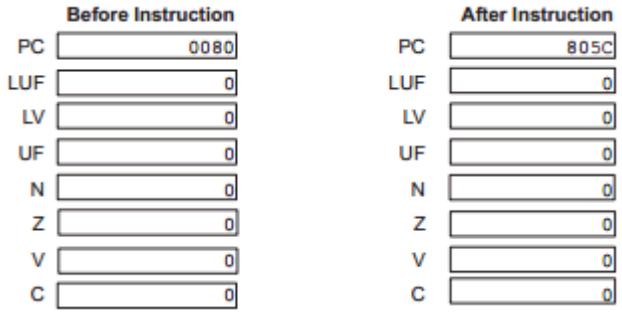
Примечание – Для стандартного перехода разряд 24 равен 0.

Циклы: 4

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: BR 805Ch



BRD Безусловный переход (с задержкой)

Синтаксис: BRD src

Операция: src → PC

Операнды: src - режим длинной непосредственной адресации

Опкод:

31 24	23 16	15 8	7 0
0 1 1 0 0 0 0	1		src

Описание: BRD это задержанный переход, при котором обеспечивается выборка трёх команд после задержанного перехода до изменения PC. В результате получается одноцикловый переход. Операнд src является 24-разрядным целым без знака.

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: BRD 2Ch

Before Instruction		After Instruction	
PC	001B	PC	002C
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

CALL Вызов подпрограммы

Синтаксис: CALL src

Операция: Следующий PC → *++SP

Операнды: src - режим длинной непосредственной адресации

Опкод:

31 24	23 16	15 8	7 0
0 1 1 0 0 0 1	0	src	

Описание: Осуществляется вызов подпрограммы. Следующее значение PC записывается в системный стек. Операнд src загружается в PC. Операнд src является 24-разрядным непосредственным операндом без знака.

Циклы: 4

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: CALL 123456h

Before Instruction		After Instruction	
PC	0005	PC	123456
SP	809801	SP	809802
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
Data memory			
		809802h	6

CALLcond Условный вызов подпрограммы

Синтаксис: CALLcond src

Операция: Если cond - истина:
Следующий PC → *++SP
Если src в регистровом режиме адресации (Rn, 0≤n≤27), src → PC.
Если src в режиме адресации относительно PC (метка или адрес), смещение + PC + 1 → PC.
В противном случае продолжение.

Операнды: Режимы адресации условного перехода src (B):
0 - регистровая
1 - относительно PC

Опкод:

31 24	23	22 19	18 16	15 8	7 0
0 1 1 1 0 0	B	0 0 0	0	cond	регистр или смещение

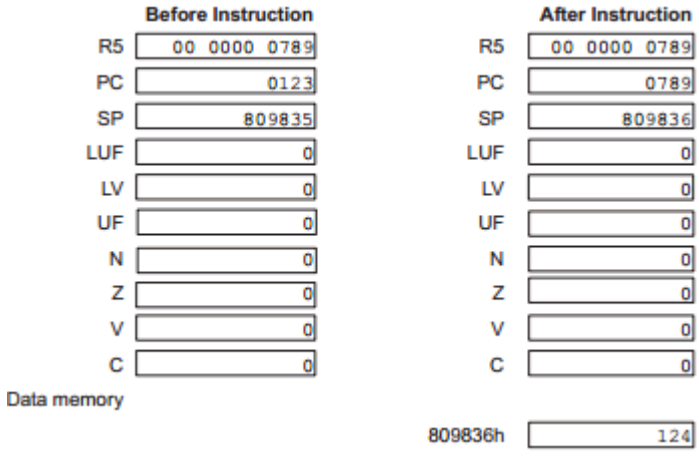
Описание: Вызов осуществляется, если условие истинно. Если условие истинно, следующее значение PC загружается в системный стек. Если операнд src установлен в режим регистровой адресации, содержимое указанного регистра загружается в PC. Если операнд src установлен в режим адресации относительно PC, ассемблер генерирует смещение; смещение = метка – (PC команды перехода + 1). Это смещение сохраняется как 16-разрядное целое со знаком в 16 младших значащих разрядах слова команды вызова. Смещение добавляется к PC команды перехода + 1 для генерации нового PC. В ПЦОС имеются 20 кодов условий, которые могут быть использованы с этой командой (в подразделе A.10.2 приведён список мнемоник условий, коды и флаги). Флаги условий устанавливаются предыдущей командой, только когда регистром назначения является один из регистров повышенной точности (R7 - R0), либо когда выполняется одна из команд сравнения (CMPF, CMPF3, CMPI, CMPI3, TSTB или TSTB3).

Циклы: 5

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: CALLNZ R5



CMPF Сравнение значений с плавающей запятой

Синтаксис: CMPF src, dst

Операция: dst → src

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 7$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 7$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 0 1 0 0 0	G	dst	src

Описание: Операнд src вычитается из операнда dst. Результат никуда не заносится, обеспечивая, таким образом, возможность сравнения без изменения значений каких бы то ни было операндов. Предполагается, что операнды dst и src являются числами в формате с ПЗ.

Циклы: 1

Разряды состояния: Флаги состояния изменяются для всех регистров назначения (R27 - R0).

LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 1 при возникновении отрицательного переполнения, иначе 0.

N 1 при генерации отрицательного результата, иначе 0.

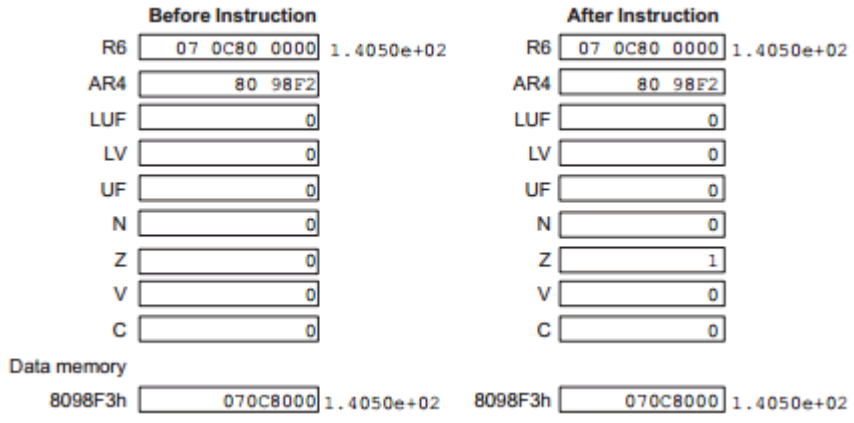
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении ПЗ-переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: CMPF *+AR4, R6



CMPF3 Сравнение значений с плавающей запятой (3 операнда)

Синтаксис: CMPF3 src2, src1

Операция: src1 - src2

Операнды: Трёхоперандные режимы адресации src1 (T):
00 - регистровая (Rn1, 0 ≤ n1 ≤ 7)
01 - косвенная (смещение = 0, 1, IR0, IR1)
10 - регистровая (Rn1, 0 ≤ n1 ≤ 7)
11 - косвенная (смещение = 0, 1, IR0, IR1)
Трёхоперандные режимы адресации src2 (T):
00 - регистровая (Rn2, 0 ≤ n2 ≤ 7)
01 - косвенная (смещение = 0, 1, IR0, IR1)
10 - регистровая (Rn2, 0 ≤ n2 ≤ 7)
11 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

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

Описание: Операнд src2 вычитается из операнда src1. Результат никуда не заносится, обеспечивая, таким образом, возможность сравнения без изменения значений каких бы то ни было операндов. Предполагается, что операнды dst и src являются числами в формате с ПЗ. Хотя команда имеет только два операнда, она имеет вид трёхоперандной, т. к. операнды определены в трёхоперандном формате.

Циклы: 1

Разряды состояния: Флаги состояния изменяются для всех регистров назначения (R27 - R0).
LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.
LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.
UF 1 при возникновении отрицательного переполнения, 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.

Пример: CMPF3 *AR2, *AR3– – (1)

	Before Instruction		After Instruction	
AR2	80 9831		AR2	80 9831
AR3	80 9852		AR4	80 9851
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
Data memory				
809831h	77A7000	2.5044e+02	809831h	77A7000 2.5044e+02
809852h	57A2000	6.253125e+01	809852h	57A2000 6.253125e+01

CMPI Сравнение целых

Синтаксис: CMPI src, dst

Операция: dst – src

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 0 1 0 0 1	G	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.

Циклы: 1

Разряды состояния: Флаги состояния изменяются для всех регистров назначения (R27 - R0).

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

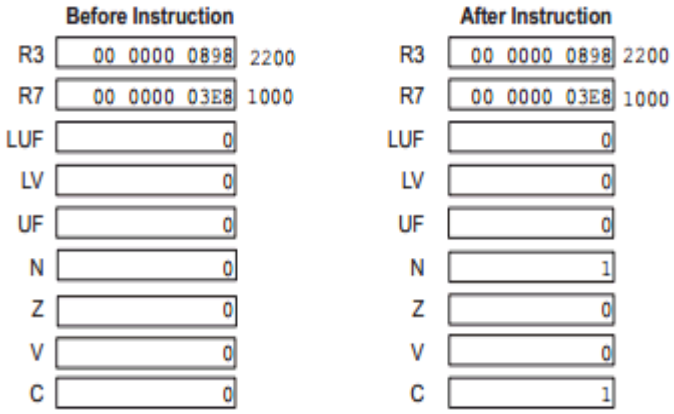
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

C 1 при возникновении заёма, иначе 0.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: CMPI R3, R7



CMPI3 Сравнение целых (3 операнда)

Синтаксис: CMPI3 src2, src1

Операция: src1 - src2

Операнды Трёхоперандные режимы адресации src1 (T):

00 - регистровая (Rn1, $0 \le n1 \le 27$)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn1, $0 \le n1 \le 27$)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Трёхоперандные режимы адресации src2 (T):

00 - регистровая (Rn2, $0 \le n2 \le 27$)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn2, $0 \le n2 \le 27$)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 24	23 16	15 8	7 0
0 0 1 0 0 0 1 1 1 T	0 0 0 0 0	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 команда имеет только два операнда, она имеет вид трёхоперандной, т. к. операнды определены в трёхоперандном формате.

Циклы: 1

Разряды состояния: Флаги состояния изменяются для всех регистров назначения (R27-R0).

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

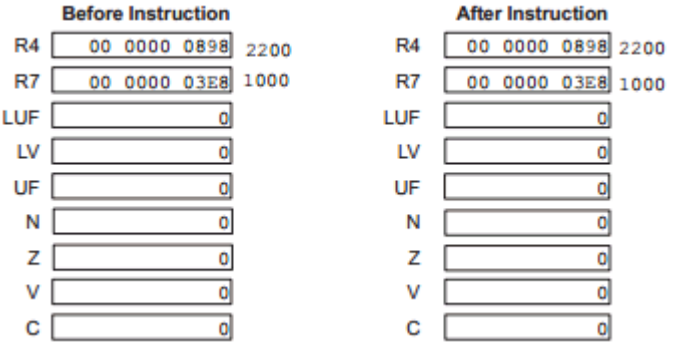
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

C 1 при возникновении заёма, иначе 0.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: CMPI3 R7, R4



Dbcond Декремент и условный переход (стандартный)

Синтаксис: DBcond ARn, src

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 + 1 → PC.

В противном случае продолжение.

Операнды: Режимы адресации условного перехода src (B):

0 - регистровая

1 - относительно PC

ARn - регистр (0 ≤ n ≤ 7)

Опкод:

31 24	23	22 21	20 16	15 8	7 0
0 1 1 0 1 1	B	ARn	0	Cond	регистр или смещение

Описание: DBcond это стандартный переход, который выполняется за четыре цикла, т. к. конвейер должен быть очищен, если cond – «истина». Определённый вспомогательный регистр декрементируется, и выполняется переход, если cond – «истина», и, если значение в определённом вспомогательном регистре больше или равно 0. Флаги условий те же самые, которые установлены последней командой, влияющей на разряды состояния. Вспомогательный регистр интерпретируется как 24-разрядное знаковое целое. Самые старшие 8 разрядов не изменяются при декрементировании. При сравнении вспомогательного регистра используется 24 младших разряда вспомогательного регистра. Обратите внимание, что условие перехода не зависит от декремента вспомогательного регистра.

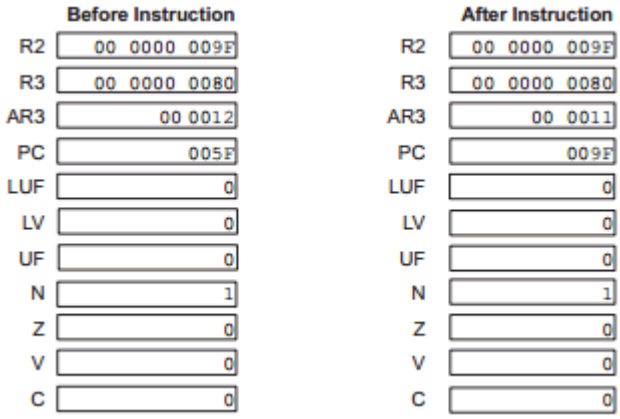
Если операнд src установлен в режим регистровой адресации, содержимое указанного регистра загружается в PC. Если операнд src установлен в режим адресации относительно PC, ассемблер генерирует смещение; смещение = метка – (PC команды перехода + 1). Это смещение сохраняется как 16-разрядное целое со знаком в 16 младших значащих разрядах слова команды перехода. Смещение добавляется к PC команды перехода + 1 для генерации нового PC. В ПЦОС имеются 20 кодов условий, которые могут быть использованы с этой команды (в подразделе A.10.2 приведён список мнемоник условий, коды и флаги). Флаги условий устанавливаются предыдущей командой, только когда регистром назначения является один из регистров повышенной точности (R7 - R0), либо когда выполняется одна из команд сравнения (CMPF, CMPF3, CMPI, CMPI3, TSTB или TSTB3).

Циклы: 4

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:
```
CMPI    200, R3
DBLT    AR3, R2
```



DBcondD Декремент и условный переход (задержанный)

Синтаксис: DBcondD ARn, src

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. Иначе: продолжение.

Операнды: Режимы адресации условного перехода src (B):

0 - регистровая

1 - относительно PC

ARn - регистр (0 ≤ n ≤ 7)

Опкод:

31 24	23	22 16	15 8	7 0
0 1 1 0 1 1	B	ARn	1	cond	регистр или смещение

Описание: DBcondD означает это переход, который выполняется за один цикл в результате того, что позволяется осуществить выборку трёх команд 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. Флаги условий те же самые, которые установлены последней командой, влияющей на разряды состояния. Три команды, следующие за DBcondD, не влияют на разряды состояния. Вспомогательный регистр интерпретируется как 24-разрядное знаковое целое. Самые старшие 8 разрядов не

изменяются при декрементировании. При сравнении вспомогательного регистра используется 24 младших разряда вспомогательного регистра. 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 команды перехода + 1). Это смещение сохраняется как 16-разрядное целое со знаком в 16 младших значащих разрядах слова команды перехода. Смещение добавляется к PC команды перехода + 1 для генерации нового PC. В ПЦОС имеются 20 кодов условий, которые могут быть использованы с этой команды (в подразделе A.10.2 приведён список мнемоник условий, коды и флаги). Флаги условий устанавливаются предыдущей командой, только когда регистром назначения является один из регистров повышенной точности (R7 - R0), либо когда выполняется одна из команд сравнения (CMPF, CMPF3, CMPI, CMPI3, TSTB или TSTB3).

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:
```
CMPI    0, R2
DBZD AR5, $+110h
```

Before Instruction		After Instruction	
R2	00 0000 0026	R2	00 0000 0026
AR5	00 0067	AR5	00 0066
PC	0100	PC	0210
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

FIX Преобразование значений с плавающей запятой в целое

Синтаксис: FIX src, dst

Операция: fix(src) → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 7$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 0 1 0 1 0	G	dst	src

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.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения - один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

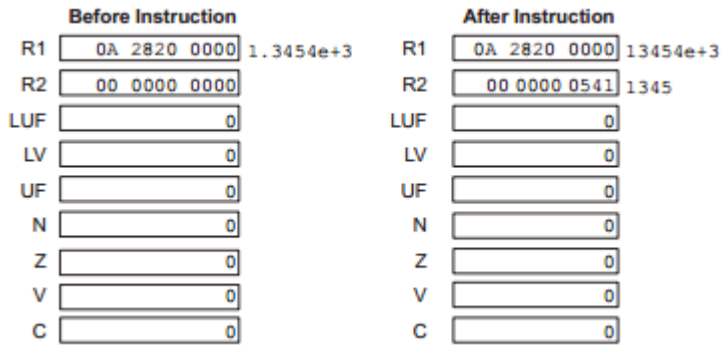
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: FIX R1, R2



FIX||STI Преобразовать значение числа в формате с ПЗ в целое и сохранить целое

Синтаксис: FIX src2, dst1
|| STI src3, dst2

Операция: fix(src2) → dst1
|| src3 → dst2

Операнды: src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn1, 0 ≤ n1 ≤ 7)
src3 - регистр (Rn2, 0 ≤ n2 ≤ 7)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод

31 24	23		16 15 8	7 0
1 1	0 1 0 1 0	dst1	0 0 0	src3	dst2	src2

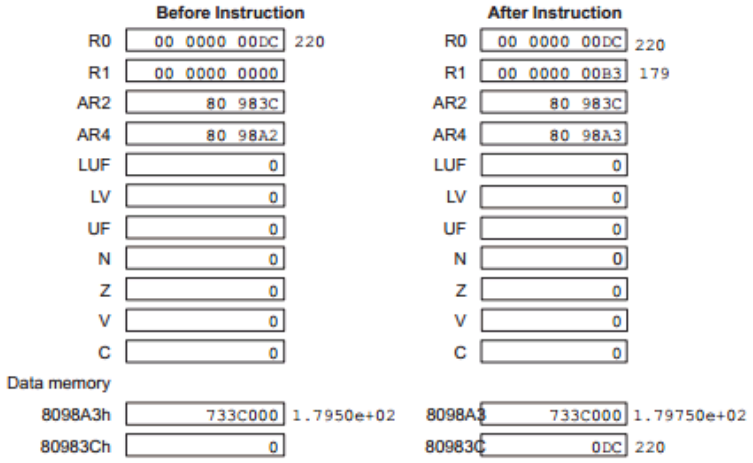
Описание: Преобразование числа в формате с ПЗ в целое. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что, если одна из параллельных операций (STI) считывает из регистра, и операция, которая производится параллельно (FIX), записывает в тот же регистр, STI имеет на входе содержимое регистра до того, как оно было изменено командой 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.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7 - R0.
LUF Не изменяется.
LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF 0
N 1 при генерации отрицательного результата, иначе 0.
Z 1 при генерации нулевого результата, иначе 0.
V 1 при возникновении целочисленного переполнения, иначе 0.
C Не изменяется.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример:
```
FIX     *++AR4 (1), R1
|| STI R0, *AR2
```



FLOAT Преобразование целых значений в значения с плавающей запятой

Синтаксис: FLOAT src, dst

Операция: float(src) → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 – косвенная

11 – непосредственная

dst - регистр (Rn, $0 \le n \le 7$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 0 1 0 1 1	G	dst	src

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.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения - один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

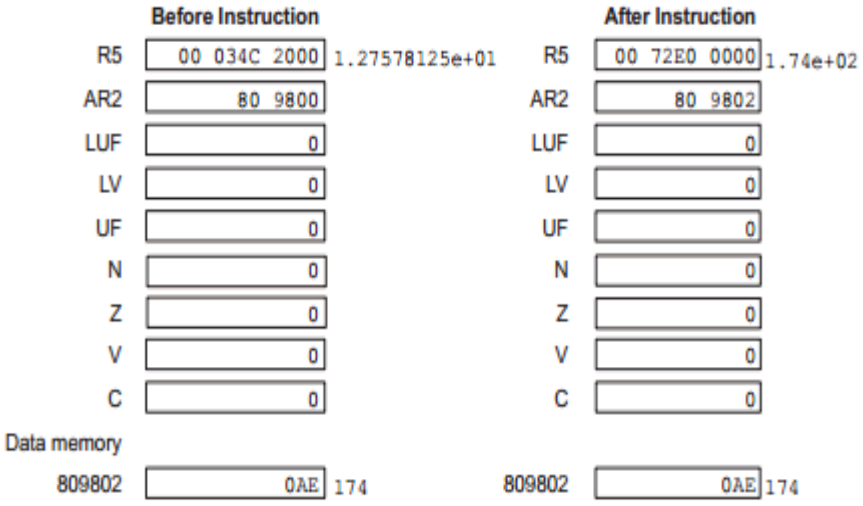
Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: FLOAT *++AR2 (2), R5



FLOAT||STF Преобразовать целое в значение числа в формате с ПЗ и сохранить в формате с ПЗ

Синтаксис: FLOAT src2, dst1
|| STF src3, dst2

Операция: float(src2) → dst1
|| src3 → dst2

Операнды: src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn1, 0 ≤ n1 ≤ 7)
src3 - регистр (Rn2, 0 ≤ n2 ≤ 7)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод

31 24	23 16	15 8	7 0
1 1 | 0 1 0 1 1	dst1 | 0 0 0	src3	dst2 | src2

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 командой FLOAT. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Циклы: 1

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 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:
```
FLOAT *+AR2 (IR0), R6
|| STF R7, *AR1
```

	Before Instruction		After Instruction	
R6	00 0000 0000		07 2E00 0000	1.740e+02
R7	03 4C20 0000	1.27578125e+01	03 4C20 0000	1.27578125e+01
AR1	80 9933		80 9933	
AR2	80 98C5		80 98C5	
IR0	8		8	
LUF	0		0	
LV	0		0	
UF	0		0	
N	0		0	
Z	0		0	
V	0		0	
C	0		0	
Data memory				
8098CD	0AE	174	0AE	174
809933	0		034C2000	1.27578125e+01

IACK Подтверждение прерывания

Синтаксис: IACK src

Операция: Выполняет пустую операцию чтения с IACK# = 0.
По завершении пустого чтения устанавливает IACK# в 1.

Операнды: основные режимы адресации src (G):
01 - прямая
10 - косвенная

Опкод:

31 24	23 16	15 8	7 0
0 0 0 1 1 0 1 1 0	G	0 0 0 0 0	src

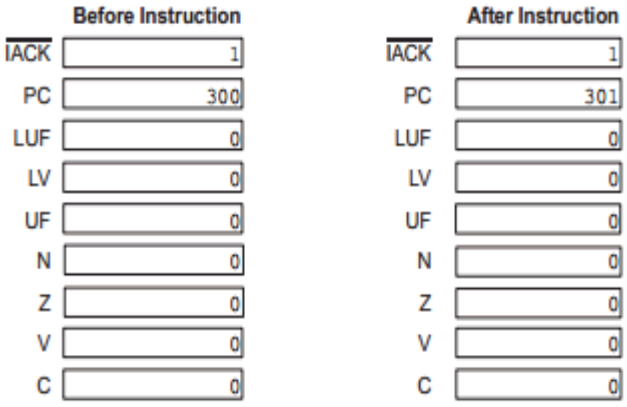
Описание: Выполняется пустая операция чтения с IACK# = 0. По завершении пустого чтения IACK# устанавливается в 1. Эта команда может быть использована для подтверждения внешнего прерывания. Если указанный адрес относится к внешней памяти, осуществляется операция чтения по данному адресу. Сигнал IACK# и адрес могут использоваться для сигнализации подтверждения прерывания внешним устройствам. Чтение данных процессором не используется.

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



IDLE Ожидание прерывания

Синтаксис: IDLE

Операция: 1 → ST (GIE)
Следующий PC → PC
Ожидание до прерывания

Операнды: Нет

Опкод:

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	

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.

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.

LDE Загрузка значения экспоненты с плавающей запятой

Синтаксис: LDE src, dst

Операция: src(exp) → dst(exp)

Операнды: основные режимы адресации src (G):
00 - регистровая (Rn, $0 \le n \le 7$)
01 - прямая
10 - косвенная
11 - непосредственная
dst - регистр (Rn, $0 \le n \le 7$)

Опкод:

31 … 24	23 … 16	15 … 8	7 … 0
0 0 0	0 0 1 1 0 1	G	dst	src

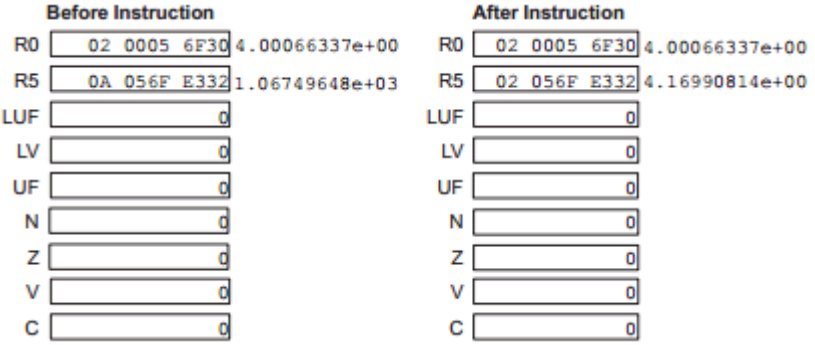
Описание: Поле экспоненты операнда src загружается в поле экспоненты регистра dst. Поле мантиссы регистра dst не изменяется, если только значение загруженной экспоненты не является зарезервированным для нуля значением экспоненты, что определяется точностью операнда src. Тогда поле мантиссы операнда dst устанавливается в ноль. Операнды src и dst являются числами в формате с ПЗ. Непосредственные значения выражаются в коротком формате с ПЗ.

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: LDE R0, R5



LDF Загрузка значения с плавающей запятой

Синтаксис: LDF src, dst

Операция: src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 7$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 7$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 0 1 1 1 0	G	dst	src

Описание: Операнд src загружается в регистр dst. Операнды src и dst являются числами в формате с плавающей запятой.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

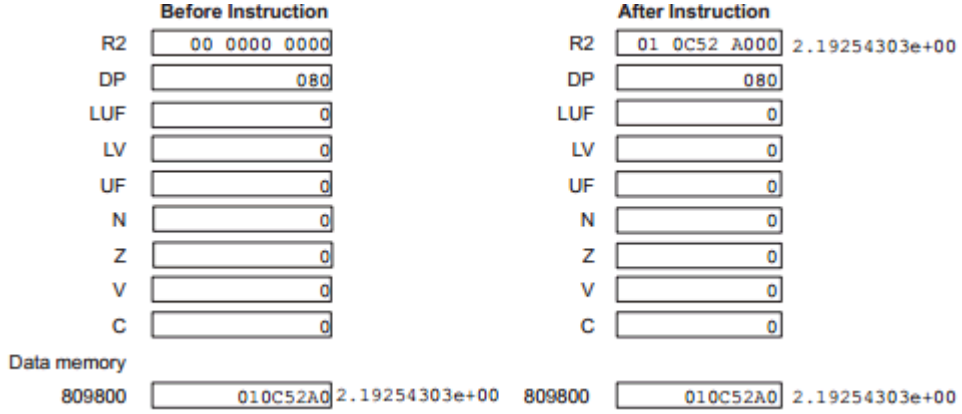
Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: LDF @9800h, R2



LDFcond Условная загрузка значения с плавающей запятой

Синтаксис: LDFcond src, dst

Операция: Если cond – «истина»:
src → dst.
Иначе:
dst не изменяется

Операнды: основные режимы адресации src (G):
00 - регистровая (Rn, 0 ≤ n ≤ 7)
01 - прямая
10 - косвенная
11 - непосредственная
dst - регистр (Rn, 0 ≤ n ≤ 7)

Опкод:

31 24	23 16	15 8	7 0
0 1 0 0 Cond	G Dst		src

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 ПЦОС имеются 20 кодов условий, которые могут быть использованы с этой команды (в подразделе A.10.2 приведён список мнемоник условий, коды и флаги). Обратите внимание, что команда LDFU (загрузить ПЗ – число безусловно) используется для загрузки R7-R0 без влияния на флаги условий. Флаги условий устанавливаются предыдущей командой, только когда регистром назначения является один из регистров повышенной точности (R7 - R0), либо когда выполняется одна из команд сравнения (CMPF, CMPF3, CMPI, CMPI3, TSTB или TSTB3).

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: LDFZ R3, R5

Before Instruction			After Instruction		
R3	2C FF2C D500	1.77055560e+13	R3	2C FF2C D500	1.77055560e+13
R5	5F 0000 003E	3.96140824e+28	R5	2C FF2C D500	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 Загрузка значения с плавающей запятой, блокировка. Сигнализирует об операции блокировки

Синтаксис: LDFI src, dst

Операция: Сигнализирует об операции блокировки.

src → dst

Операнды: основные режимы адресации src (G):

01 - прямая

10 - косвенная

dst - регистр (Rn, $0 \leq n \leq 7$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 0 1 1 1 1 1	G	dst	src

Описание: Операнд src загружается в регистр dst. Операция блокировки (межпроцессорного взаимодействия) сигнализируется через XF0 и XF1. Операнды src и dst являются числами в формате с ПЗ. Обратите внимание, что определена только прямая и косвенная адресация. Детальное описание см. в подразделе A.6.4.

Циклы: 1, если XF1 = 0 (см. подраздел A.6.4)

Разряды состояния: LUF Не изменяется.

LV Не изменяется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

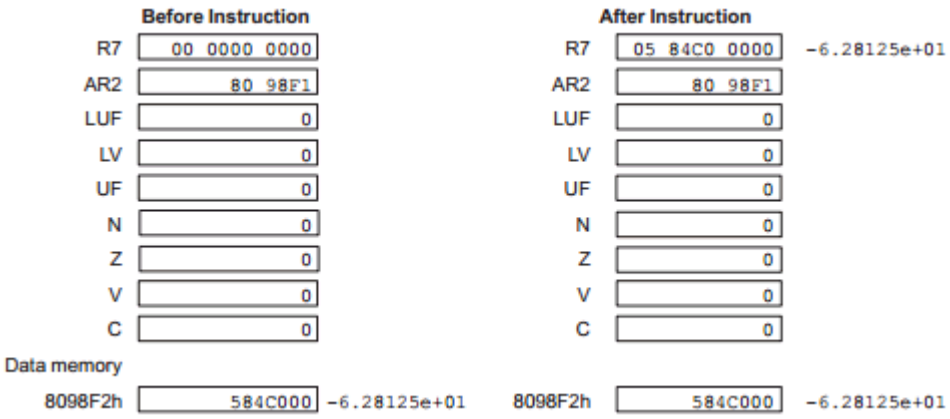
Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: LDFI *+AR2, R7



LDF||LDF Загрузить значение в формате с П3

Синтаксис: LDF src2, dst2

|| LDF src1, dst1

Операция: src2 → dst2

|| src1 → dst1

Операнды: src1 - косвенная (смещение = 0, 1, IR0, IR1)

dst1 - регистр (Rn1, $0 \le n1 \le 7$)

src2 - косвенная (смещение = 0, 1, IR0, IR1)

dst2 - регистр (Rn2, $0 \le n2 \le 7$)

Опкод

31		24	23		16	15		8	7		0
1 1	0 0 0 1 0		dst2	dst1	0 0 0		src1			src2	

Описание: Загрузка двух чисел выполняется параллельно. Если обе LDF загружают в один регистр, ассемблер выдаёт предупреждение. В результате этого получается LDF src2, dst2

Циклы: 1

Разряды состояния: LUF Не изменяется.

LV Не изменяется.

UF Не изменяется.

N Не изменяется.

Z Не изменяется.

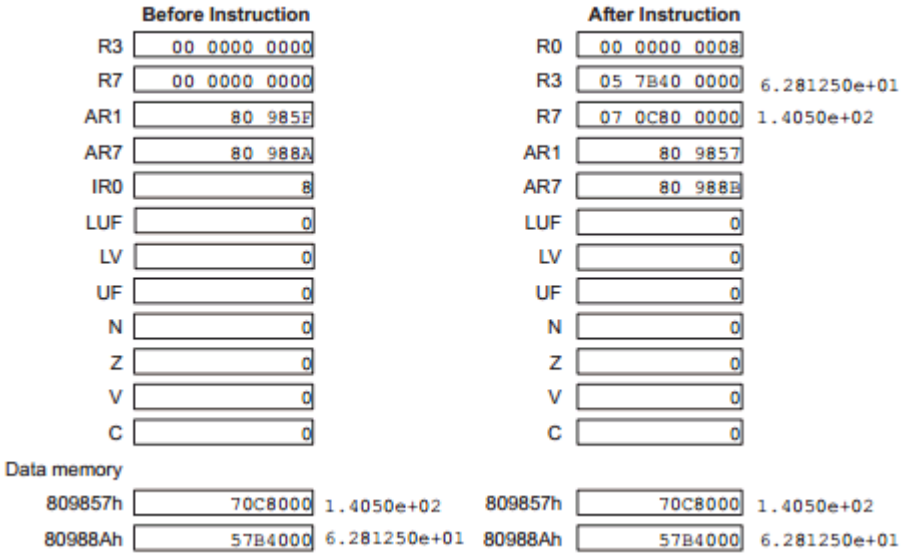
V Не изменяется.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: LDF *– –AR1 (IR0), R7

|| LDF *AR7++ (1), R3



LDF||STF Параллельное выполнение LDF и STF

Синтаксис: LDF src2, src1
|| STF src3, dst2

Операция: src2 → dst1
|| src3 → dst2

Операнды: src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn1, 0 ≤ n1 ≤ 7)
src3 - регистр (Rn2, 0 ≤ n2 ≤ 7)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод

31 24	23 16	15 8	7 0
1 1 | 0 1 1 0 0	dst1 | 0 0 0	src3	dst2 | src2

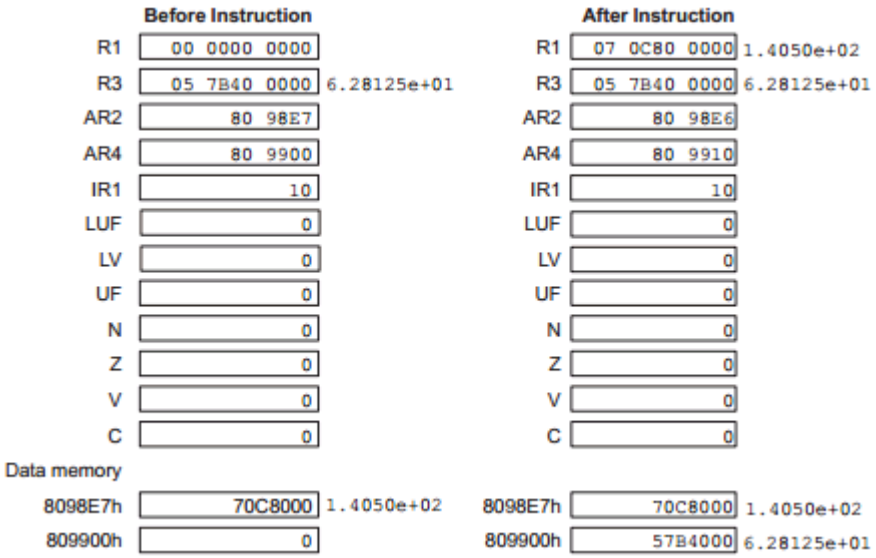
Описание: Параллельно выполняются операции загрузки и сохранения чисел в формате с ПЗ. Если операнды src2 и dst2 указывают на один и тот же адрес, src2 считывается до того, как будет записан dst2.

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:
```
LDF    *AR2– – (1), R1
|| STF R3, *AR4++ (IR1)
```



LDI Загрузка целого

Синтаксис: LDI src, dst

Операция: src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 1 0 0 0 0 0	G	Dst	src

Описание: Операнд src загружается в регистр dst. Операнды src и dst являются целыми со знаком. Альтернативная форма LDI, LDP, используется для загрузки регистра указателя страницы памяти (DP) или любого другого регистра восьмью старшими разрядами перемещаемого адреса. Смотри команду LDP и A.10.3.2.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

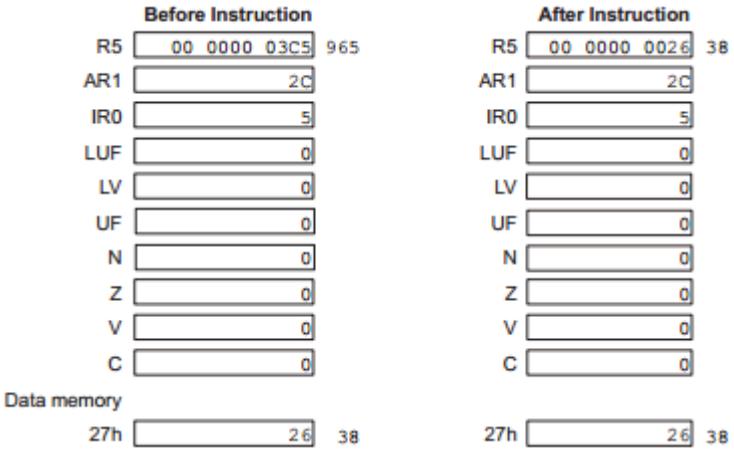
Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: LDI *-AR1 (IR0), R5



LDIcond Условная загрузка целых

Синтаксис: LDIcond src, dst

Операция: Если cond - "истина":
src → dst
Иначе:
dst не изменяется

Операнды: основные режимы адресации src (G):
00 - регистровая (Rn, $0 \le n \le 27$)
01 - прямая
10 - косвенная
11 - непосредственная
dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 24	23 16	15 8	7 0
0 1 0 1 Cond	G dst	src	

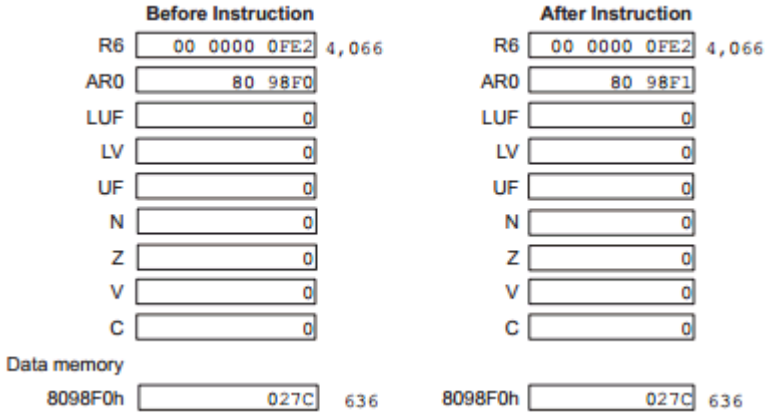
Описание: Если cond – «истина», операнд src загружается в регистр dst. В противном случае регистр dst не изменяется. Вне зависимости от условия, имеет место чтение регистра src. Операнды src и dst являются целыми со знаком. В ПЦОС имеются 20 кодов условий, которые могут быть использованы с этой команды (в подразделе A.10.2 приведён список мнемоник условий, коды и флаги). Обратите внимание, что команда LDIU (безусловная загрузка целого) используется для загрузки R7-R0 без изменения флагов условий. Флаги условий устанавливаются предыдущей командой, только когда регистром назначения является один из регистров повышенной точности (R7 - R0), либо когда выполняется одна из команд сравнения (CMPF, CMPF3, CMPI, CMPI3, TSTB или TSTB3).

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 зависит от значения разряда OVM.

Пример: LDIZ R4, R6



LDII Загрузка целых, блокировка. Сигнализирует о команде блокировки

Синтаксис: LDII src, dst

Операция: Сигнализирует об операции блокировки.

src → dst

Операнды: основные режимы адресации src (G):

01 - прямая

10 - косвенная

dst - регистр (Rn, $0 \leq n \leq 27$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 1 0 0 0 1	G	dst	src

Описание: Операнд src загружается в регистр dst. Операция блокировки (межпроцессорного взаимодействия) сигнализируется через XF0 и XF1. Операнды src и dst являются целыми со знаком. Необходимо обратить внимание, что определены только прямая и косвенная адресации. Детальное описание см. в подразделе A.6.4.

Циклы: 1, если XF1 = 0 (см. подраздел A.6.4)

Разряды состояния: LUF Не изменяется.

LV Не изменяется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

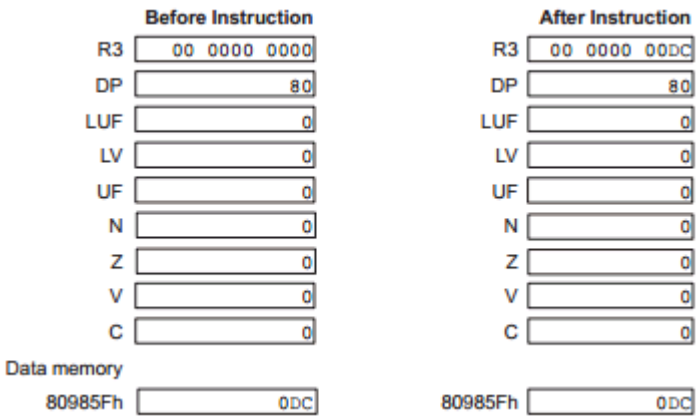
Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: LDII @985Fh, R3



LDI||LDI Загрузить целое

Синтаксис: LDI src2, dst2
|| LDI src1, dst1

Операция: src2 → dst2
|| src1 → dst1

Операнды: src1 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn1, 0 ≤ n1 ≤ 7)
src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst2 - регистр (Rn2, 0 ≤ n2 ≤ 7)

Опкод

31 24	23 16	15 8	7 0
1 1 0 0 0 1 1	dst2	dst1	0 0 0	src1	src2

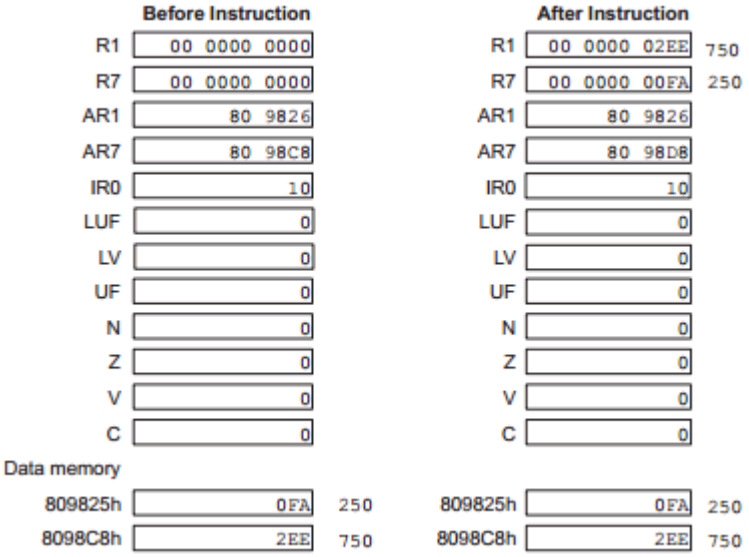
Описание: Загрузка двух целых выполняется параллельно. Если обе LDI загружают в один регистр, ассемблер выдаёт предупреждение. В результате этого получается LDI src2, dst2.

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:
```
LDI *– –AR1 (1), R7
|| LDI *AR7++ (IR0), R1
```



LDI||STI Загрузить целое и сохранить целое

Синтаксис: LDI src2, src1
|| STI src3, dst2

Операция: src2 → dst1
|| src3 → dst2

Операнды: src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn1, $0 \le n1 \le 7$)
src3 - регистр (Rn2, $0 \le n2 \le 7$)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод

31 24	23 16	15 8	7 0
1 1 | 0 1 1 0 1	dst1 | 0 0 0	src3	dst2 | src2

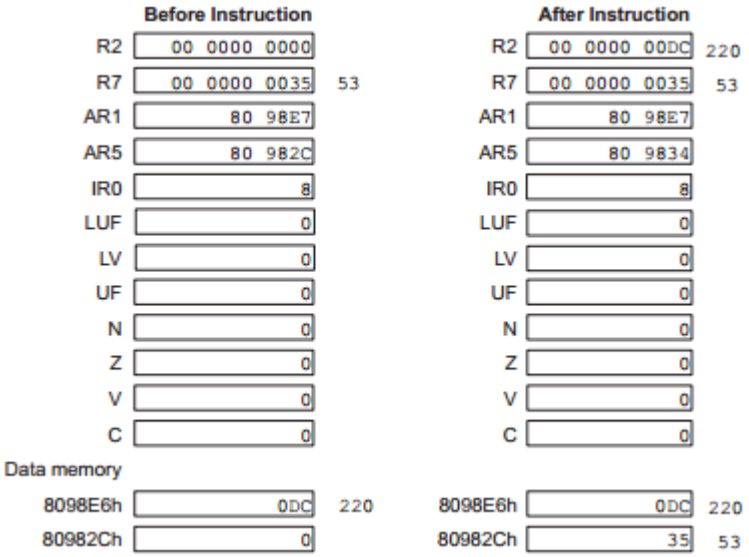
Описание: Параллельно выполняются операции загрузки и сохранения целых. Если операнды src2 и dst2 указывают на один и тот же адрес, src2 считывается до того, как будет записан dst2.

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:
```
LDI     *-AR1 (1), R2
|| STI   R7, *AR5++ (IR0)
```



LDM Загрузка мантиссы с плавающей запятой

Операция: src(man) → dst(man)

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 7$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 7$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 1 0 0 1 0	G	dst	src

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. Если операнд src загружается из памяти, соответствующее содержимое памяти загружается как мантисса. При использовании непосредственного режима адресации разряды 15-12 слова команды устанавливаются в 0 ассемблером.

Циклы: 1

Разряды состояния: LUF Не изменяется.

LV Не изменяется.

UF Не изменяется.

N Не изменяется.

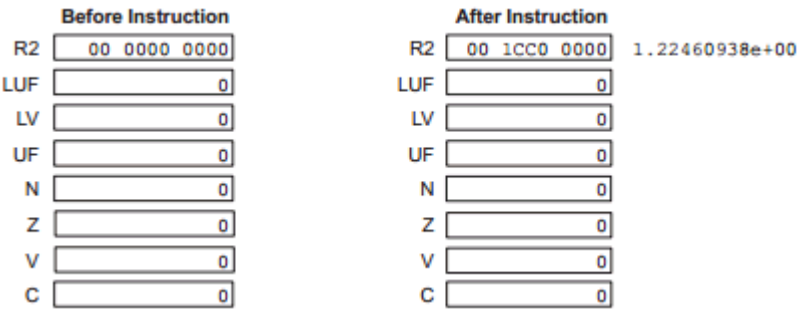
Z Не изменяется.

V Не изменяется.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: LDM 156.75, R2 (156.75 = 071CC00000h)



LDP Указатель загрузки данных страницы

Синтаксис: LDP src, DP

Операция: src → Указатель страницы данных.

Операнды: src - 8 старших разрядов абсолютного 24-разрядного адреса источника (src). "DP" в операнде является дополнительным.

dst регистр (Rn, 0 ≤ n ≤ 7)

Опкод

31 24	23 16	15 8	7 0
0 0 0 0 1 0 0 0 0 0	1 1 1 0 0 0 0	0 0 0 0 0 0 0 0	src2

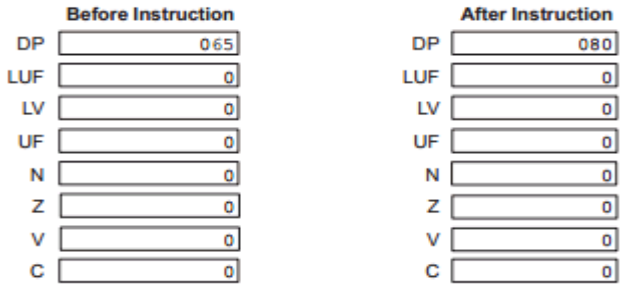
Описание: Эта псевдооперация является альтернативной формой команды LDI, за исключением того, что команда LDP определена только для непосредственного режима адресации. Поле операнда src содержит восемь старших разрядов абсолютного 24-разрядного адреса src (т. е. только 24-16 разряды src используются). Эти восемь разрядов загружены в младшие восемь разрядов указателя страницы данных. Младшие восемь разрядов указателя используются в прямой адресации как указатели страницы данных, к которой будет осуществляться адресация. Всего имеется 256 страниц, каждая из которых размером 64K слов. Разряды 31-8 указателя зарезервированы и хранятся как 0.

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:
```
LDP @809900h,      DP
или
LDP @809900h
```



LSH Логический сдвиг

Синтаксис: LSH count, dst

Операция: Если count ≥0:

dst << count → dst.

Иначе:

dst >> |count| → dst.

Операнды: основные режимы адресации count (G):

00 - регистровая (Rn, 0 ≤ n ≤ 27)

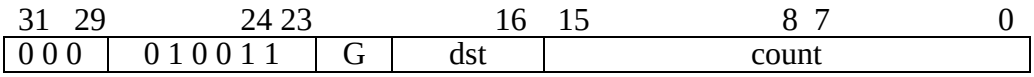
01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:



Описание: Семь младших разрядов операнда count используются для генерации двоичного дополнения счётчика сдвига. Если операнд count больше 0, операнд dst сдвигается влево на количество разрядов, определённое значением операнда count. Младшие разряды заполняются нулями, старший разряд сдвигаем переносится в разряд C (carry) регистра состояния. Логический сдвиг влево: C ← dst ←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 → dst → C. Если операнд count = 0, сдвиг не происходит и разряд переноса C устанавливается в 0. Предполагается, что операнд count является целым со знаком, а оператор dst – беззнаковое целое.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N Старший разряд выходного значения.

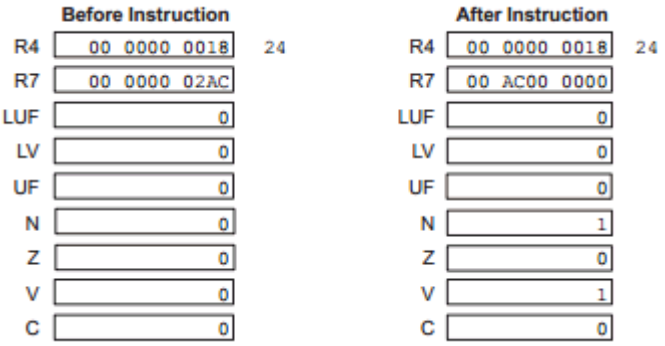
Z 1 при генерации нулевого результата, иначе 0.

V 0

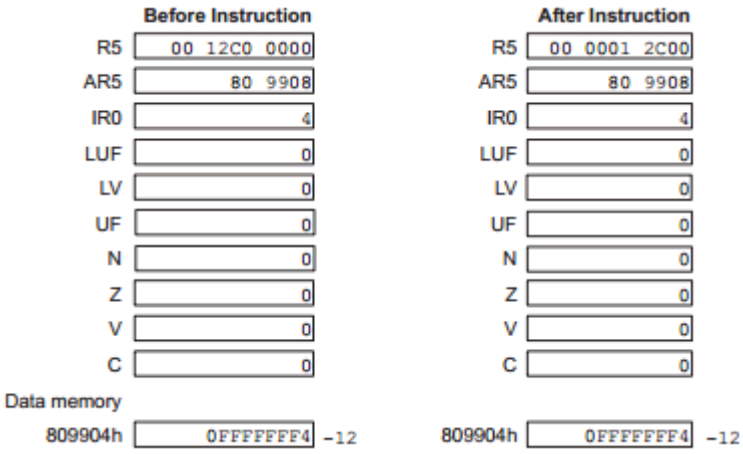
C Устанавливается по значению последнего сдвинутого вовне разряда.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример 1: LSH R4, R7



Пример 2: LSH *-AR5 (IR1), R5



LSH3 Логический сдвиг (3 операнда)

Синтаксис: LSH3 count, src, dst

Операция: Если count ≥0:

src << count → dst.

Иначе:

src >> |count| → dst.

Операнды: Трёхоперандные режимы адресации src (T):

00 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Трёхоперандные режимы адресации count (T):

00 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

11 - косвенная (смещение = 0, 1, IR0, IR1)

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 29	24 23	16	15 8	7 0
0 0 1	0 0 1 0 0 0	T	dst	src	count

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 в разряд переноса C регистра состояния. Логический сдвиг влево: C← dst← 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 в разряд переноса C регистра состояния. Логический сдвиг вправо: 0 → dst → 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.

Циклы: 1

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 Младший разряд выходного значения.

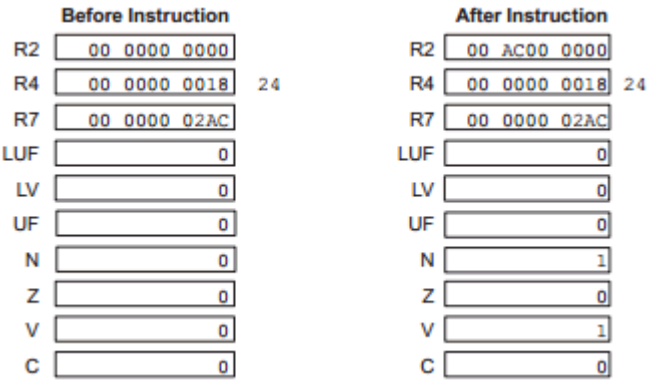
Z 1 при генерации нулевого выхода, иначе 0.

V 0

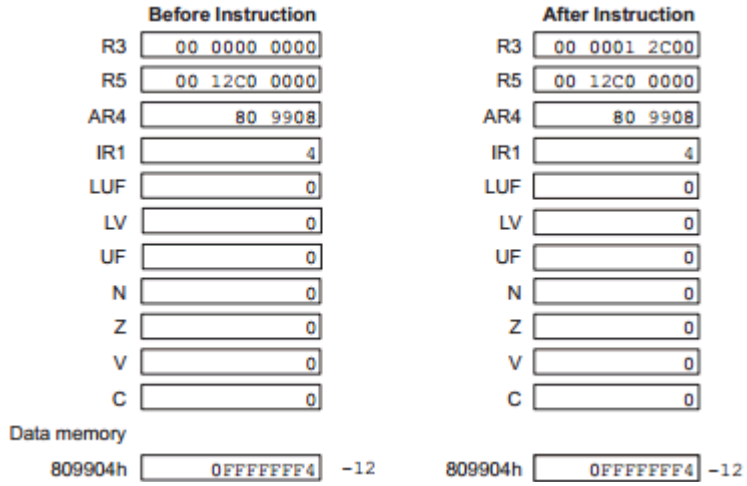
C Устанавливается по значению последнего сдвинутого в разряд переноса C регистра состояния. 0 для счётчика сдвига, равного 0. Не изменяется, если dst не является одним из R7-R0.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример 1: LSH3 R4, R7, R2



Пример 2: LSH3 *-AR4 (IR1), R5, R3



LSH3||STI Логический сдвиг и сохранение целого

Синтаксис: LSH3 count, src2, dst1
|| STI src3, dst2

Операция: Если count ≥0:
src2 << count → dst1.
Иначе:
src2 >> |count| → dst1
|| src3 → dst2

Операнды: count - регистр (Rn1, 0 ≤ n1 ≤ 7)
src1 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn3, 0 ≤ n3 ≤ 7)
src2 - регистр (Rn4, 0 ≤ n4 ≤ 7)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31		24	23		16	15	8	7	0
1 1	0 1 1 1 0	dst1	count	src3	dst2	src2			

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 в разряд переноса C регистра состояния. 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 в разряд переноса C регистра состояния. Логический сдвиг вправо: 0 → dst2 → C. Если операнд count = 0, сдвиг не происходит и разряд переноса C устанавливается в 0. Предполагается, что операнд 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 проводится параллельно (LSH3), записывает в тот же регистр, STI имеет на входе содержимое регистра до того, как оно было изменено командой LSH3. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Циклы: 1

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 Младший разряд выходного значения.

Z 1 при генерации нулевого выхода, иначе 0.

V 0

C Устанавливается по значению последнего сдвинутого во вне разряда. 0 для счётчика сдвига, равного 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример 1:

```
LSH3 R2, *++AR3 (1), R0
|| STI R4, *-AR5
```

	Before Instruction		After Instruction	
R0	00 0000 0000		00 AC00 0000	
R2	00 0000 0018	24	00 0000 0018	24
R4	00 0000 00DC	220	00 0000 00DC	220
AR3	80 98C2		80 98C3	
AR5	80 98A3		80 98A3	
LUF	0		0	
LV	0		0	
UF	0		0	
N	0		0	
Z	0		0	
V	0		0	
C	0		0	
Data memory				
8098C3h	0AC		0AC	
8098A2h	0		0DC	220

Пример 2:

```
LSH3 R7, *AR2– – (1), R2
```

	Before Instruction		After Instruction	
R0	00 0000 012C	300	00 0000 012C	300
R2	00 0000 0000		00 0002 C000	
R7	00 FFFF FFF4	-12	00 FFFF FFF4	-12
AR0	80 98B7		80 98B7	
AR2	80 9863		80 9862	
LUF	0		0	
LV	0		0	
UF	0		0	
N	0		0	
Z	0		0	
V	0		0	
C	0		0	
Data memory				
809863h	2C000000		2C000000	
8098B8h	0		12C	300

MPYF Умножение значений с плавающей запятой

Синтаксис: MPYF src, dst

Операция: dst × src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 7$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 7$)

Опкод:

31 … 24	23 … 16	15 … 8	7 … 0
0 0 0 | 0 1 0 1 0 0 | G	dst	src	

Описание: Результат умножения операндов dst и src загружается в регистр dst. Операнд src является числом в формате с ПЗ с одинарной точностью, операнд dst является числом в формате с ПЗ с расширенной точностью.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 1 при возникновении отрицательного переполнения, иначе 0.

N 1 при генерации отрицательного результата, иначе 0.

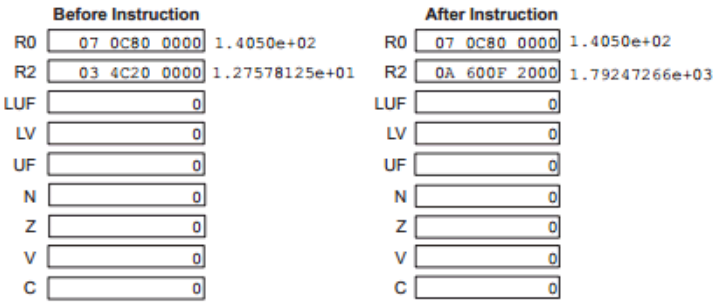
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении ПЗ-переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: MPYF R0, R2



MPYF3 Умножение значений с плавающей запятой (3 операнда)

Синтаксис: MPYF3 src2, src1, dst

Операция: src1 ×src2 → dst

Операнды: Трёхоперандные режимы адресации src1 (T):

00 - регистровая (Rn1, $0 \le n1 \le 7$)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn1, $0 \le n1 \le 7$)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Трёхоперандные режимы адресации src2 (T):

00 - регистровая (Rn2, $0 \le n2 \le 7$)

01 - регистровая (Rn2, $0 \le n2 \le 7$)

10 - косвенная (смещение = 0, 1, IR0, IR1)

11 - косвенная (смещение = 0, 1, IR0, IR1)

dst - регистр (Rn, $0 \le n \le 7$)

Опкод:

31 24	23 16	15 8	7 0
0 0 1 0 0 1 0 0 1	T	Dst	src1	src2

Описание: Результат умножения операндов dst1 (src1) и src загружается в регистр dst. Операнды src1 и src2 являются числом в формате с ПЗ с одинарной точностью, операнд dst является числом в формате с ПЗ с расширенной точностью.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 1 при возникновении отрицательного переполнения, иначе 0.

N 1 при генерации отрицательного результата, иначе 0.

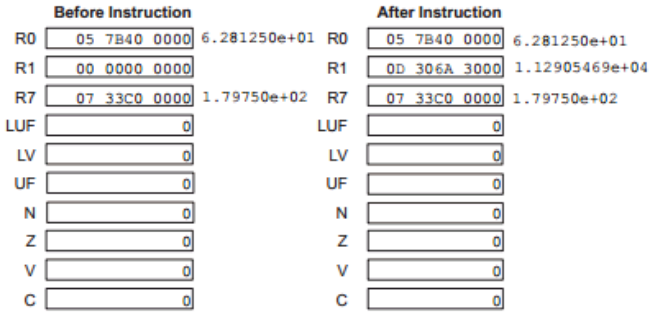
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении ПЗ-переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

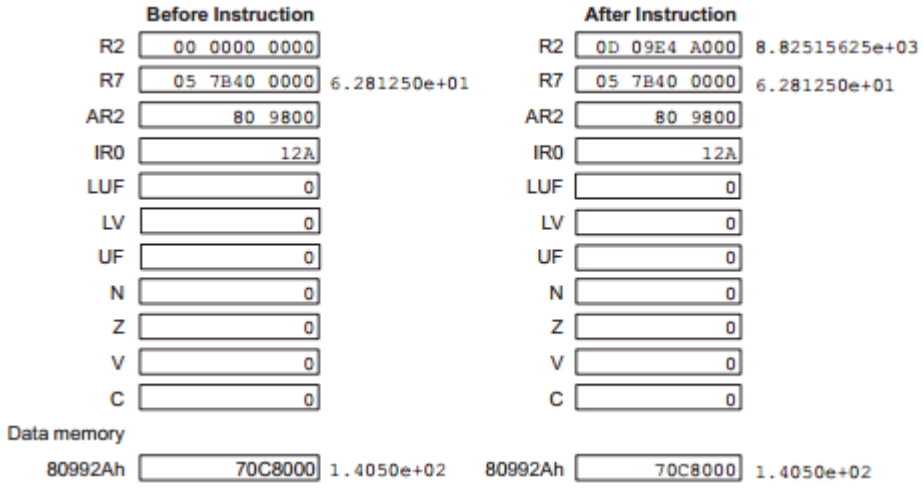
Пример 1: MPYF3 R0, R7, R1



Пример 2: MPYF3 *+AR2 (IR0), R7, R2

или

MPYF3 R7, *+AR2 (IR0), R2



MPYF3||ADDF3 Умножить и прибавить значение в формате с ПЗ

Синтаксис: MPYF3 srcA, srcB, dst1

|| ADDF3 srcC, srcD, dst2

Операция: srcA × srcB → dst1

|| srcC + srcD → dst2

Операнды: srcA |

srcB | Любые два регистры ($0 \le Rn \le 7$)

srcC | Любые два косвенная (смещение = 0, 1, IR0, IR1)

srcD |

dst1 - регистр (d1):

0 = R0

1 = R1

dst2 - регистр (d2):

0 = R2

1 = R3

src1 - регистр (Rn, $0 \le n \le 7$)

src2 - регистр (Rn, $0 \le n \le 7$)

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

Опкод:

31 24	23 16	15 8	7 0
1 0 | 0 0 0 0 | P | d1 | d2 | src1	src2	src3	src4

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 командой 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. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 1 при возникновении отрицательного переполнения, иначе 0.

N 0

Z 0

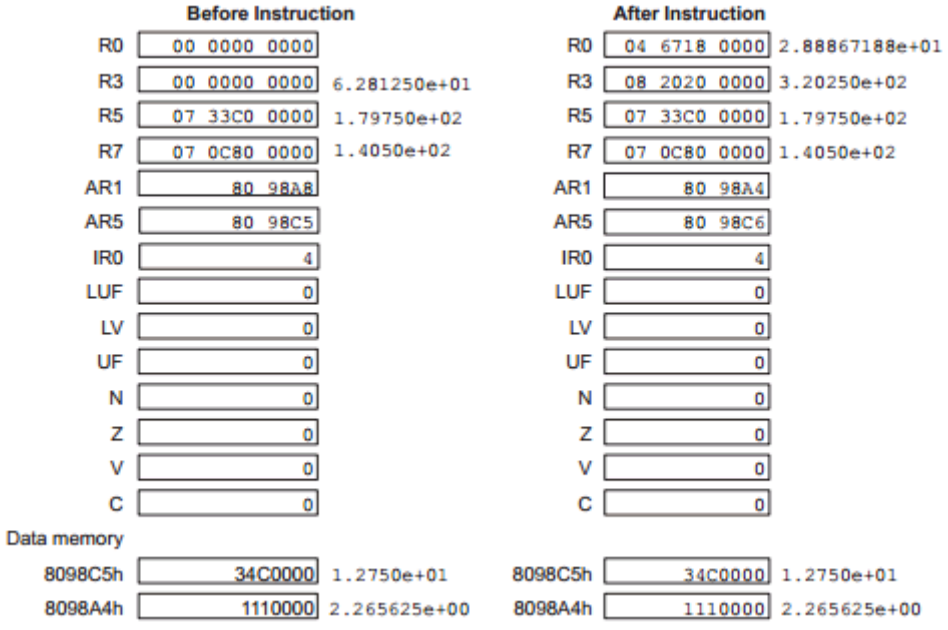
V 1 при возникновении ПЗ-переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример: MPYF3 *AR5++ (1): – –AR1 (IR0), R0

|| ADDF3 R5, R7, R3



MPYF3||STF Умножить значения в формате с ПЗ и сохранить значение с ПЗ

Синтаксис: MPYF3 src2, src1, dst
|| STF src3, dst2

Операция: src1× src2 → dst1
|| src3 → dst2

Операнды: src1 - регистр (Rn1, 0 ≤ n1 ≤ 7)
src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn3, 0 ≤ n3 ≤ 7)
src3 - регистр (Rn4, 0 ≤ n4 ≤ 7)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 24	23 16	15 8	7 0
1 1 | 0 1 1 1 1 | dst1	src1 | src3	dst2	src2

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 командой MPYF3. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

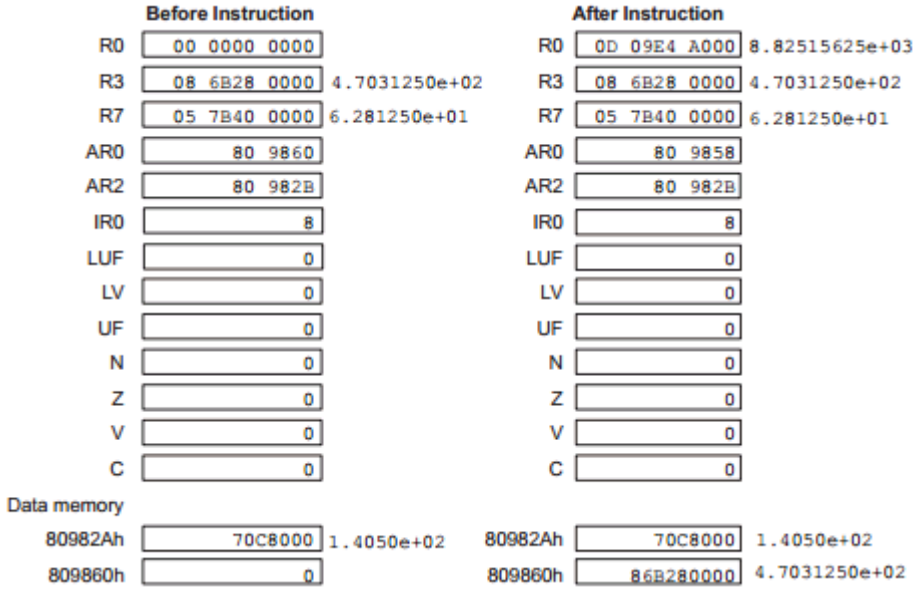
Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.
LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.
LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.
UF 1 при возникновении отрицательного переполнения, 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.

Пример: MPYF3 *-AR2 (1), R7, R0

|| STF R3, *AR0– – (IR0)



MPYF3||SUBF3 Умножить и вычесть значение в формате с ПЗ

Синтаксис: MPYF3 srcA, srcB, dst1

|| SUBF3 srcC, srcD, dst2

Операция: srcA ×srcB → dst1

|| srcD - srcC → dst2

Операнды: srcA |

srcB | Любые два регистры (0 ≤ Rn ≤ 7)

srcC | Любые два косвенная (смещение = 0, 1, IR0, IR1)

srcD |

dst1 - регистр (d1):

0 = R0

1 = R1

dst2 - регистр (d2):

0 = R2

1 = R3

src1 - регистр (Rn, 0 ≤ n ≤ 7)

src2 - регистр (Rn, 0 ≤ n ≤ 7)

src3 - косвенная (смещение = 0, 1, IR0, IR1)

src4 - косвенная (смещение = 0, 1, IR0, IR1)

P - параллельные режимы адресации (0 ≤ P ≤ 3)

Операция (Поле P)

00 - src3 × src4, src1 - src2

01 - src3 × src1, src4 - src2

10 - src1× src2, src3 - src4

11 - src3 × src1, src2 - src4

Опкод:

31 24	23 16	15 8	7 0
1 0 | 0 0 0 1 | P | d1 | d2 | src1	src2	src3	src4

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 командой 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.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.
LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.
LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.
UF 1 при возникновении отрицательного переполнения, иначе 0.
N 0
Z 0
V 1 при возникновении ПЗ-переполнения, иначе 0.
C Не изменяется.

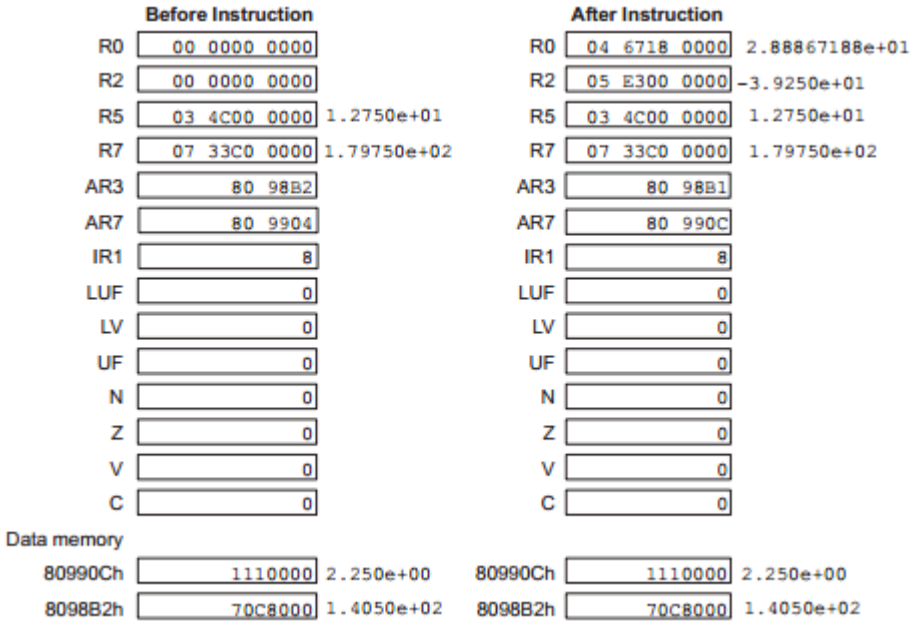
Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример:

```
MPYF3 R5, *++AR7 (IR1), R0
|| SUBF3 R7, *AR3– – (1), R2
```

или

```
MPYF3 *++AR7 (IR1), R5, R0
|| SUBF3 R7, *AR3– – (1), R2
```



MPYI Умножение целых

Синтаксис: MPYI src, dst

Операция: dst × src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, 0 ≤ n ≤ 27)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 1 0 1 0 1 1	G	dst	src

Описание: Результат произведения операндов dst и src загружается в регистр dst. При чтении операнды src и dst являются 24-разрядными целыми со знаком. Результат является 48-разрядным целым со знаком. В регистр dst помещается 32 младших значащих разряда результата. Целочисленное переполнение возникает, когда хотя бы один из старших 16 разрядов 48-разрядного результата отличается от старшего разряда 32-разрядного выходного значения.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

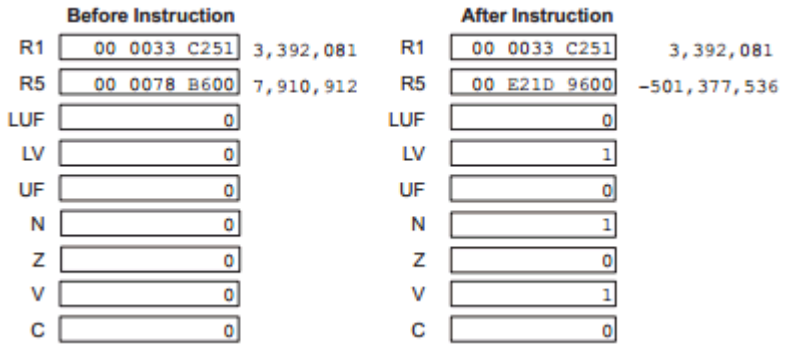
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример: MPYI R1, R5



MPYI3 Умножение целых (3 операнда)

Синтаксис: MPYI3 src2, src1, dst

Операция: src1 × src2 → dst

Операнды: Трёхоперандные режимы адресации src1 (T):

00 - регистровая (Rn1, n1 ≤ 27)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn1, n1 ≤ 27)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Трёхоперандные режимы адресации src2 (T):

00 - регистровая (Rn2, n2 ≤27)

01 - регистровая (Rn2, n2 ≤ 27)

10 - косвенная (смещение = 0, 1, IR0, IR1)

11 - косвенная (смещение = 0, 1, IR0, IR1)

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 24	23	16	15 8	7 0
0 0 1 0 0 1 0 1 0	T	dst	src1	src2

Описание: Результат произведения операндов src1 и src2 загружается в регистр dst. Операнды src1 и src2 являются 24-разрядными целыми со знаком. Результат является 48-разрядным целым со знаком. В регистр dst помещается 32 младших значащих разряда результата. Целочисленное переполнение возникает, когда хотя бы один из старших 16 разрядов 48-разрядного результата отличается от старшего разряда 32-разрядного выходного значения.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

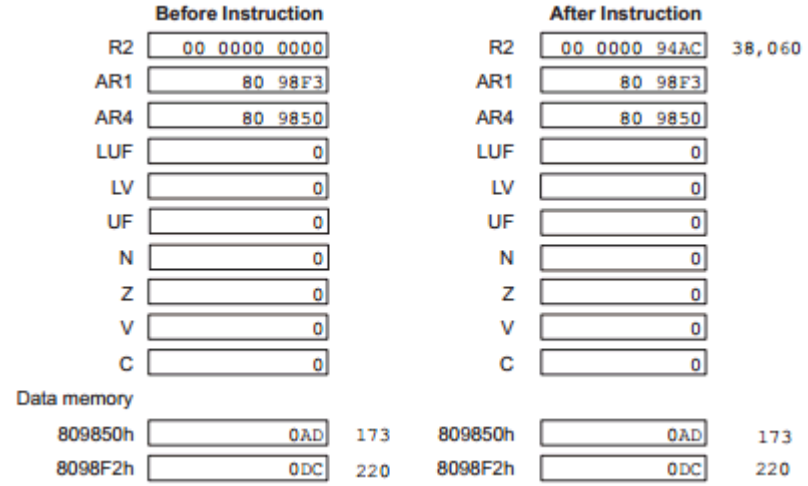
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

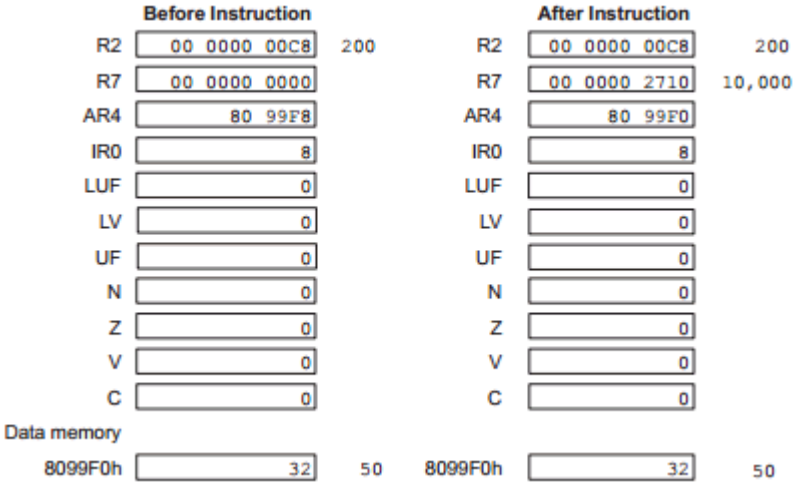
C Не изменяется.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример 1: MPYI3 *AR4, *-AR1 (1), R2



Пример 2: MPYI3 *– –AR4 (IR0), R2, R7



MPYI3||ADDI3 Умножить и прибавить целое

Синтаксис: MPYI3 srcA, srcB, dst1
|| ADDI3 srcC, srcD, dst2

Операция: srcA × srcB → dst1
|| srcD + srcC → dst2

Операнды: srcA |
srcB | Любые два регистры ($0 \le Rn \le 7$)
srcC | Любые два косвенная (смещение = 0, 1, IR0, IR1)
srcD |

dst1 - регистр (d1):
0 = R0
1 = R1
dst2 - регистр (d2):
0 = R2
1 = R3
src1 - регистр (Rn, $0 \le n \le 7$)

src2 - регистр (Rn, $0 \le n \le 7$)

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)

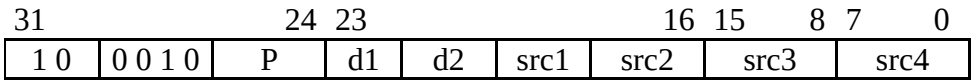
00 - src3 × src4, src1 + src2

01 - src3 × src1, src4 + src2

10 - src1× src2, src3 + src4

11 - src3 × src1, src2 + src4

Опкод:



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 командой 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.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 0

Z 0

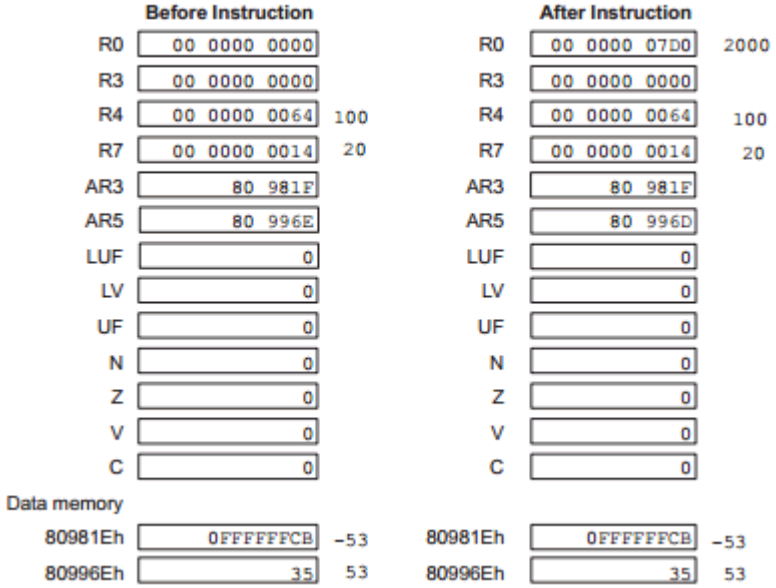
V 1 при возникновении целочисленного переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример: MPYI3 R7, R4, R0

|| ADDI3 *-AR3, *AR5– – (1), R3



MPYI3||STI Умножить целое и сохранить целое

Синтаксис: MPYI3 src2, src1, dst

|| STI src3, dst2

Операция: src1× src2 → dst1

|| src3 → dst2

Операнды: src1 - регистр (Rn1, 0 ≤ n1 ≤ 7)

src2 - косвенная (смещение = 0, 1, IR0, IR1)

dst1 - регистр (Rn3, 0 ≤ n3 ≤ 7)

src3 - регистр (Rn4, 0 ≤ n4 ≤ 7)

dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 24	23 16	15 8	7 0
1 1 | 1 0 0 0 0	dst1 | src1	src3	dst2 | src2

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 командой MPYI3. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.
LUF Не изменяется.
LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF 0
N 1 при генерации отрицательного результата, иначе 0.
Z 1 при генерации нулевого результата, иначе 0.
V 1 при возникновении целочисленного переполнения, иначе 0.
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример:
```
MPYI3      *-AR0 (1), R5, R7
|| STI R2, *-AP3 (1)
```

Before Instruction			After Instruction		
R2	00 0000 00DC	220	R2	00 0000 00DC	220
R5	00 0000 0032	50	R5	00 0000 0032	50
R7	00 0000 0000		R7	00 0000 2710	10000
AR0	80 995A		AR0	80 995B	
AR3	80 982F		AR3	80 982F	
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	
Data memory					
80995Bh	0C8	200	80995Bh	0C8	200
80982Eh	0		80982Eh	0DC	220

MPYI3||SUBI3 Умножить и вычесть целое

Синтаксис:
```
MPYI3 srcA, srcB, dst1
|| SUBI3 srcC, srcD, dst2
```

Операция:
```
srcA × srcB → dst1
|| srcD - srcC → dst2
```

Операнды:
```
srcA |
srcB | Любые два регистры (0 ≤ Rn ≤ 7)
srcC | Любые два косвенная (смещение = 0, 1, IR0, IR1)
srcD |
```

dst1 - регистр (d1):
```
0 = R0
1 = R1
```
dst2 - регистр (d2):
```
0 = R2
1 = R3
```
src1 - регистр (Rn, 0 ≤ n ≤ 7)
src2 - регистр (Rn, 0 ≤ n ≤ 7)
src3 - косвенная (смещение = 0, 1, IR0, IR1)
src4 - косвенная (смещение = 0, 1, IR0, IR1)
P - параллельные режимы адресации (0 ≤ P ≤ 3)
Операция (Поле P)
00 - src3 × src4, src1 - src2
01 - src3 × src1, src4 - src2
11 - src3 × src1, src2 - src4

Опкод:

31 24	23				16 15	8 7	0
1 0	0 0 1 1	P	d1	d2	src1	src2	src3	src4

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 командой 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. Когда это не очень важно, ассемблер может изменить порядок в коммутативных операциях с целью упрощения обработки.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 1 при возникновении отрицательного целочисленного переполнения, в противном случае не меняется.

N 0

Z 0

V 1 при возникновении целочисленного переполнения, иначе 0.

C Не изменяется.

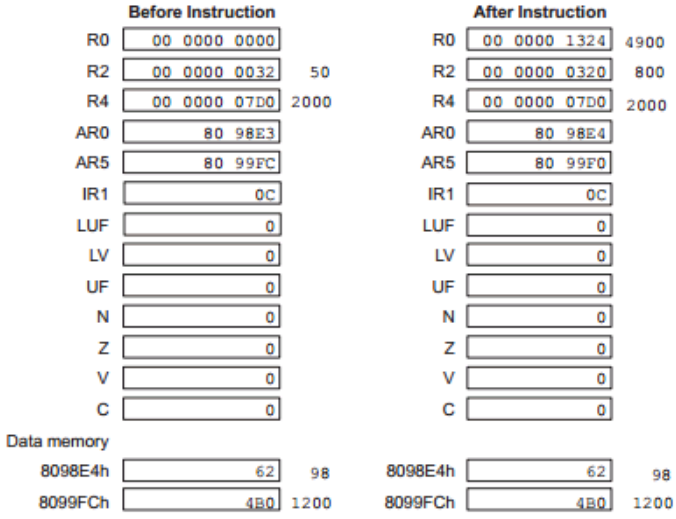
Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример:

```
MPYI3 R2, *++AR0 (1), R0
|| SUBI3 *AR5– – (IR1), R4, R2
```

или

```
MPYI3 *++AR0 (1), R2, R0
|| SUBI3 *AR5– – (IR1), R4, R2
```



NEGB Отрицание целого с заёмом

Синтаксис: NEGB src, dst

Операция: 0 - src - C → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, 0 ≤ n ≤ 27)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 1 0 1 1 0	G	dst	src

Описание: Разница от 0 операнда src и C загружаются в регистр dst. Предполагается, src и dst являются целыми со знаком.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

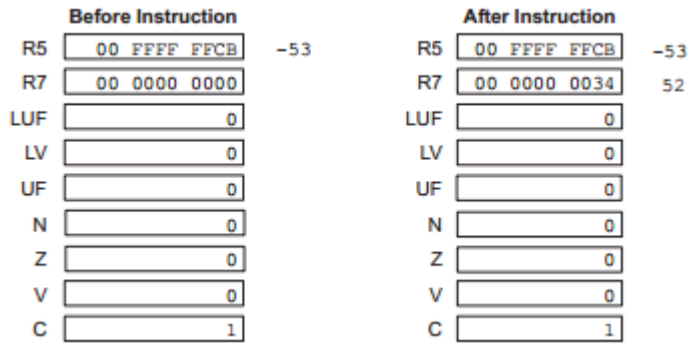
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

C 1 при генерации заёма, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример: NEGB R5, R7



NEGF Отрицание значения с плавающей запятой

Синтаксис: NEGF src, dst

Операция: 0 - src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, 0 ≤ n ≤ 7)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, 0 ≤ n ≤ 7)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 1 0 1 1 1	G	dst	src

Описание: Операнд, представляющий собой отличие от нуля операнда src загружается в регистр dst. Предполагается, src и dst являются числами в формате с ПЗ.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 1 при возникновении отрицательного переполнения, иначе 0.

N 1 при генерации отрицательного результата, иначе 0.

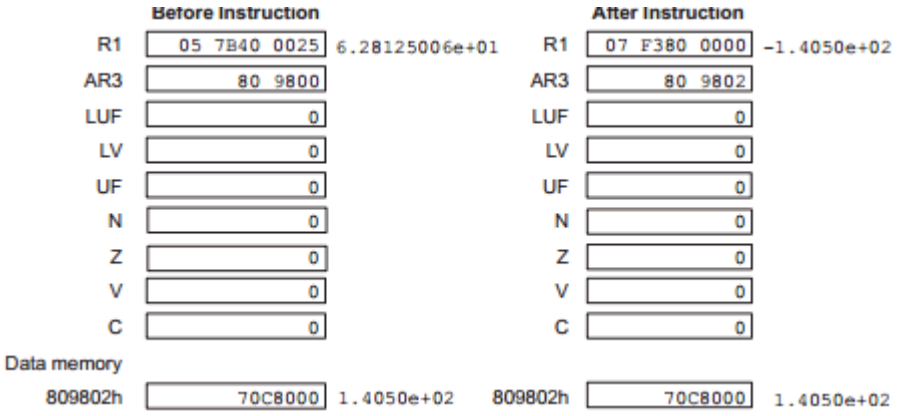
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении ПЗ-переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример: NEGF *++AR3 (2), R1



NEGF||STF Обратное значение в формате с ПЗ и сохранить значение с ПЗ

Синтаксис: NEGF src2, dst1

|| STF src3, dst2

Операция: 0 - src2 → dst1

|| src3 → dst2

Операнды: src2 - косвенная (смещение = 0, 1, IR0, IR1)

dst1 - регистр (Rn1, $0 \le n1 \le 7$)

src3 - регистр (Rn2, $0 \le n2 \le 7$)

dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 24	23 16	15 8	7 0		
1 1	1 0 0 0 1	dst1	0 0 0	src3	dst2	src2

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 командой NEGF. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 1 при возникновении отрицательного переполнения, иначе 0.

N 1 при генерации отрицательного результата, иначе 0.

Z 1 при генерации нулевого результата, иначе 0.

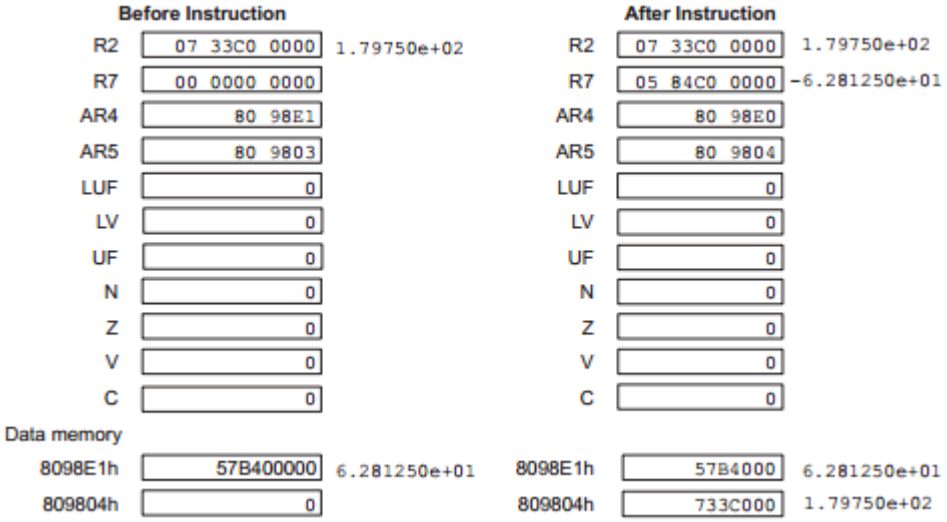
V 1 при возникновении ПЗ-переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример:

```
NEGF *AR4– – (1), R7
|| STF R2,*++AR5 (1)
```



NEGI Отрицание целого числа

Синтаксис: NEGI src, dst

Операция: 0 - src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 27$)
01 - прямая
10 - косвенная
11 - непосредственная
dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 1 1 0 0 0	G	dst	src

Описание: Операнд, представляющий собой отличие от нуля операнда src, загружается в регистр dst. Предполагается, src и dst являются целыми со знаком.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

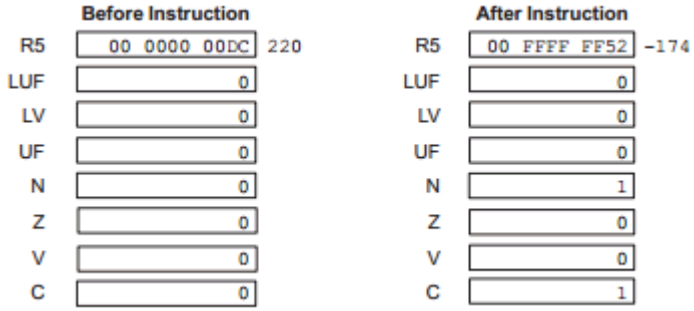
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

C 1 при возникновении заёма, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример: NEGI 174, R5 (174 = 0AEh)



NEGI||STI Обратное целое и сохранить целое

Синтаксис: NEGI src2, dst1

|| STI src3, dst2

Операция: 0 - src2 → dst1

|| src3 → dst2

Операнды: src2 - косвенная (смещение = 0, 1, IR0, IR1)

dst1 - регистр (Rn1, $0 \leq n1 \leq 7$)

src3 - регистр (Rn2, $0 \leq n2 \leq 7$)

dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 24	23 16	15 8	7 0
1 1 | 1 0 0 1 0	dst1 | 0 0 1	src3	dst2 | src2

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 командой NEGI. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

C 1 при возникновении заёма, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример:

```
NEGI *-AR3, R2
 || STI R2, *AR1++
```

	Before Instruction			After Instruction	
R2	00 0000 0019	25	R2	00 FFFF FF24	-220
AR1	80 98A5		AR1	80 98A6	
AR3	80 982F		AR3	80 982F	
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	
Data memory					
80982Eh	0DC	220	80982Eh	0DC	220
8098A5h	0		8098A5h	19	25

NOP Нет операции

Синтаксис: NOP src

Операция: Нет никаких операций АЛУ или умножителя. ARn изменяются, если src определён в режиме косвенной адресации.

Операнды: основные режимы адресации src (G):

00 - регистровая (нет операции)

10 - косвенная (изменяет ARn, $0 \le n \le 7$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 1 1 0 0 1	G	0 0 0 0 0	src

Описание: Если операнд src определён в косвенном режиме адресации, выполняется определённая операция адресации и имеет место пустое чтение памяти. Если операнд src опущен, не выполняются никакие операции.

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 1: NOP

Пример 2: NOP *AR3– – (1)

NORM Нормирование значений с плавающей запятой

Синтаксис: NORM src, dst

Операция: norm(src) → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, 0 ≤ n ≤ 7)

01 - прямая

10 - косвенная

11 - непосредственная

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 1 1 0 1 0	G	dst	src

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. 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. См. подраздел A.4.6.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.

LV Не изменяется.

UF 1 при возникновении отрицательного переполнения, иначе 0.

N 1 при генерации отрицательного результата, иначе 0.

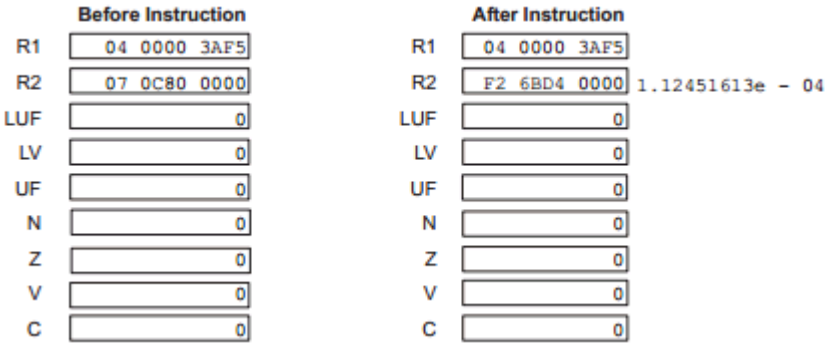
Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: NORM R1, R2



NOT Поразрядное логическое дополнение

Синтаксис: NOT src, dst

Операция: ~src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 1 1 0 1 1	G	dst	src

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.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N Старший разряд выходного значения.

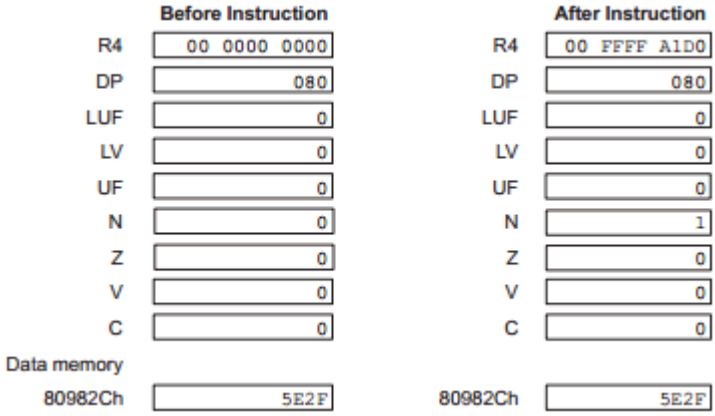
Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример: NOT @982Ch, R4



NOT||STI Логическое дополнение (поразрядная инверсия) значения и сохранить целое

Синтаксис: NOT src2, dst1
|| STI src3, dst2

Операция: ~src2 → dst1
|| src3 → dst2

Операнды: src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn1, 0 ≤ n1 ≤ 7)
src3 - регистр (Rn2, 0 ≤ n2 ≤ 7)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 24	23 16	15 8	7 0
1 1 | 1 0 0 1 1	dst1 | 0 0 0	src3	dst2 | src2

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 командой NOT. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

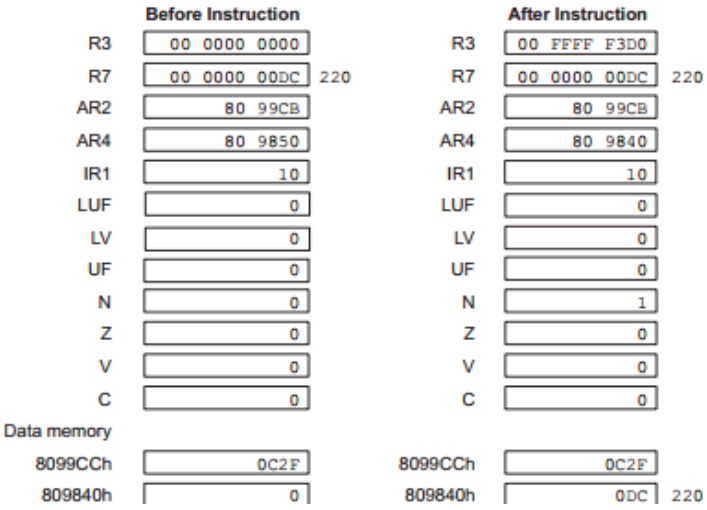
Циклы: 1

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 результата, иначе 0.
V 0
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: NOT *AR2, R3

|| STI R7, *– –AR4 (IR1)



OR Поразрядное логическое OR (ИЛИ)

Синтаксис: OR src, dst

Операция: dst OR src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 1 0 0 0 0 0 0	G	dst	src

Описание: Поразрядное логическое ИЛИ (OR) между операндами src и dst загружается в регистр dst. Операнды src и dst являются беззнаковыми целыми.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N Старший разряд выходного значения.

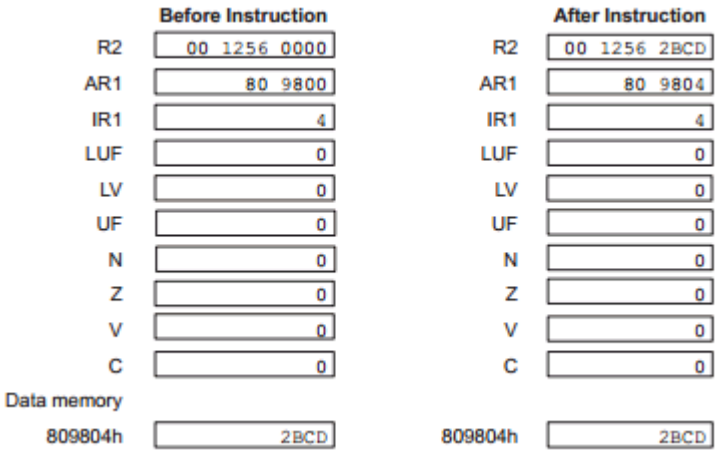
Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: OR *++AR1 (IR1), R2



OR3 Поразрядное логическое – OR (ИЛИ) (3 операнда)

Синтаксис: OR3 src2, src1, dst

Операция: src1 OR src2 → dst

Операнды: Трёхоперандные режимы адресации src1 (T):

00 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Трёхоперандные режимы адресации src2 (T):

00 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

01 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

10 - косвенная (смещение = 0, 1, IR0, IR1)

11 - косвенная (смещение = 0, 1, IR0, IR1)

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 24	23 16	15 8	7 0
0 0 1 0 0 1 0 1 1	T	dst	src1	src2

Описание: Поразрядное логическое ИЛИ(OR) между операндами src1 и src2 загружается в регистр dst. Операнды src1, src2 и dst являются беззнаковыми целыми.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N Старший разряд выходного значения.

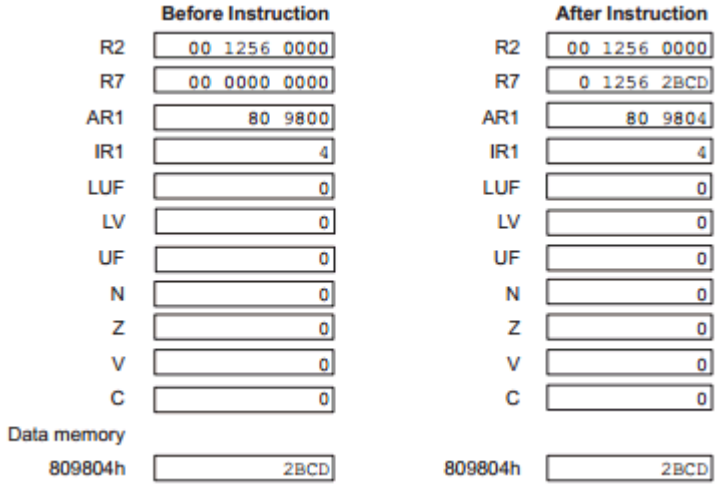
Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: OR3 *++AR1(IR1), R2, R7



OR3||STI Поразрядное логическое ИЛИ и сохранить целое

Синтаксис: OR3 src2, src1, dst1
|| STI src3, dst2

Операция: src1 OR src2 → dst1
|| src3 → dst2

Операнды: src1 - регистр (Rn1, 0 ≤ n1 ≤ 7)
src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn2, 0 ≤ n2 ≤ 7)
src3 - регистр (Rn3, 0 ≤ n3 ≤ 7)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 24	23 16	15 8	7 0
0 0 | 1 0 1 0 0 | dst1	src1 | src3	dst2	src2

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 командой OR3. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

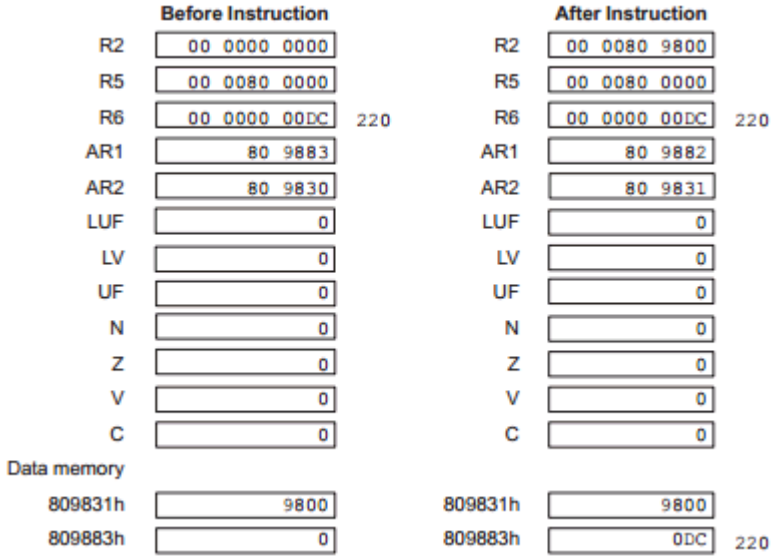
Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, когда регистр назначения – один из R7 - R0.
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.

Пример: OR3 *++AR2, R5, R2

|| STI R6, *AR1– –



POP Выталкивание целых значений из стека

Синтаксис: POP dst

Операция: *SP– – → dst

Операнды: dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 | 0 1 1 1 0 0 | 0 1	dst	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	

Описание: Вершина системного стека выталкивается и загружается в регистр dst (в 32 младших разряда). Вершина стека – целое со знаком. Команда POP выполняется с постдекрементом указателя стека. Разряды экспоненты в регистрах расширенной точности (R7-R0) слева не изменяются.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

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: POP R3

	Before Instruction		After Instruction	
R3	00 0000 12DA	4,826	00 FFFF 0DA4	-62,044
SP	809856		809855	
LUF	0		0	
LV	0		0	
UF	0		0	
N	0		1	
Z	0		0	
V	0		0	
C	0		0	
Data memory				
809856h	FFFF0DA4	-62,044	FFFF0DA4	-62,044

POPF Выталкивание значений с плавающей запятой из стека

Синтаксис: POPF dst

Операция: *SP– – → dst1

Операнды: dst - регистр (Rn, 0 ≤ n ≤ 7)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 0 1 1 1 0 0	0 1	dst	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Описание: Вершина системного стека выталкивается и загружается в регистр dst (в 32 старших разряда). Вершина стека – число в формате с ПЗ. Команда POPF выполняется с постдекрементом указателя стека. 8 младших разрядов в регистрах расширенной точности (R7-R0) заполняются нулями.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

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

	Before Instruction		After Instruction	
R4	02 5D2E 0123	6.91186578e+00	5F 2C13 0200	5.32544007e+28
SP	80984A		809849	
LUF	0		0	
LV	0		0	
UF	0		0	
N	0		0	
Z	0		0	
V	0		0	
C	0		0	
Data memory				
80984Ah	5F2C1302	5.32544007e+28	5F2C1302	5.32544007e+28

PUSH Загрузка в стек целого

Синтаксис: PUSH src

Операция: src → *++SP

Операнды: src - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 | 0 1 1 1 1 0 | 0 1	dst	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	

Описание: Содержимое регистра src (32 младших значащих разряда) загружается в текущий системный стек. Предполагается, что src является целым со знаком. Команда PUSH выполняется с прединкрементом указателя стека. Целая часть или мантисса регистров с расширенной точностью (R7-R0) сохраняется при этой команды.

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: PUSH R6

	Before Instruction			After Instruction	
R6	02 5C12 8081	633,415,688	R6	02 5C12 8081	633,415,688
SP	8098AE		SP	8098AF	
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	
Data memory					
8098AFh	0	-62,044	8098AFh	5C128081	1,544,716,417

PUSHF Загрузка значений с плавающей запятой в стек

Синтаксис: PUSHF src

Операция: src → *++SP

Операнды: src - регистр (Rn, 0 ≤ n ≤ 7)

Опкод:

31 … 24	23	22 … 16	15 … 8	7 … 0
0 0 0	0 1 1 1 1 0	0 1	dst	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

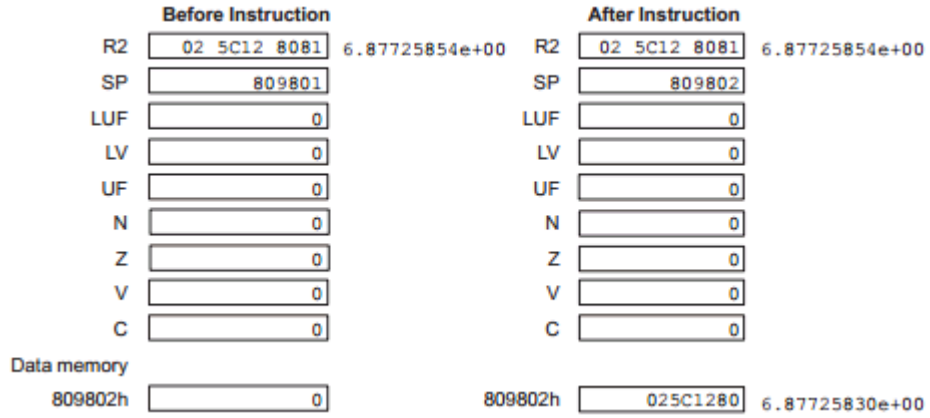
Описание: Содержимое регистра src (32 старших значащих разряда) загружается в текущий системный стек. Предполагается, что src является числом в формате с ПЗ. Команда PUSHF выполняется с прединкрементом указателя стека. 8 младших значащих разрядов мантиссы не сохраняются (обратите внимание на отличие значений в R2 и в стеке в приведённом ниже примере).

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: PUSHF R2



RETIcond Условный выход из прерывания

Синтаксис: RETIcond

Операция: Если cond – "истина":
*SP– – → PC
1 → ST (GIE).
Иначе, продолжение.

Операнды: Нет

Опкод:

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

Описание: Выполняется условный возврат. Если условие истинно, вершина стека выталкивается в PC, и 1 записывается в разряд GIE (глобальное разре-

шение прерывания) регистра состояния. Это имеет эффектом разрешение всех прерываний, для которых соответствующий разряд разрешения прерывания установлен в 1. В ПЦОС имеются 20 кодов условий, которые могут быть использованы с этой команды (в подразделе A.10.2, где приведён список мнемоник условий, коды и флаги). Флаги условий устанавливаются предыдущей командой, только когда регистром назначения является один из регистров повышенной точности (R7 - R0), либо когда выполняется одна из команд сравнения (CMPF, CMPF3, CMPI, CMPI3, TSTB или TSTB3).

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: RETINZ

	Before Instruction		After Instruction
PC	0456	PC	0123
SP	809830	SP	80982F
ST	0	ST	2000
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
Data memory			
809830h	123	809830h	123

RETScond Условный выход из подпрограммы

Синтаксис: RETScond

Операция: Если cond - "истина":

*SP– – → PC

Иначе, продолжение.

Операнды: Нет

Опкод:

31	24	23	16	15	8	7	0
0 1 1 1 1	0 0 0 1	0 0	cond	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0			

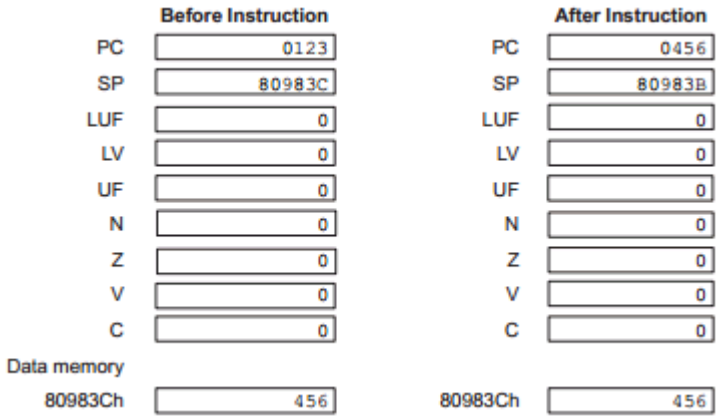
Описание: Выполняется условный возврат. Если условие истинно, вершина стека выталкивается в PC. В ПЦОС имеются 20 кодов условий, которые могут быть использованы с этой команды (в подразделе A.10.2 приведён список мнемоник условий, коды и флаги). Флаги условий устанавливаются предыдущей командой, только когда регистром назначения является один из регистров повышенной точности (R7 - R0), либо когда выполняется одна из команд сравнения (CMPF, CMPF3, CMPI, CMPI3, TSTB или TSTB3).

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: RETSGE



RND Округление значения с плавающей запятой

Синтаксис: RND src, dst

Операция: rnd (src) → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 7$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 7$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 | 1 0 0 0 1 0	G	dst	src

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.

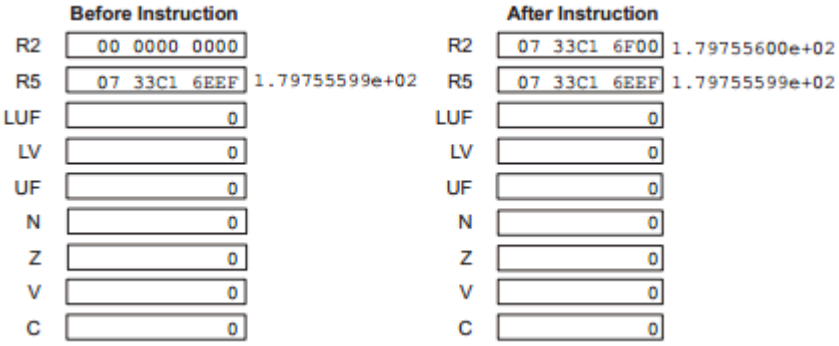
Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.
LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.
LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.
UF 1 при возникновении отрицательного переполнения, 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.

Пример: RND R5, R2



ROL Циклический сдвиг влево

Синтаксис: ROL dst

Операция: dst циклически сдвинутый влево на 1 разряд → dst

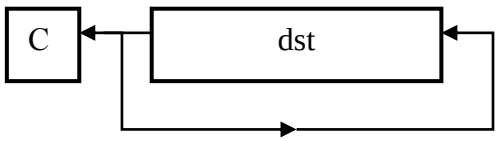
Операнды: dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 | 1 0 0 0 1 1 | 1 1	dst	src	

Описание: Содержимое операнда dst циклически сдвигается влево на один разряд и загружается в регистр dst. Это циклический сдвиг с пересылкой старшего значащего разряда в младший.

Циклический левый сдвиг:



Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

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. Не изменяется, если dst не является одним из R7-R0.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: ROL R3

Before Instruction	
R3	00 8002 5CD4
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

After Instruction	
R3	00 0004 B9A9
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	1

ROLC Циклический сдвиг влево через перенос

Синтаксис: ROLC dst

Операция: dst циклически сдвинутый влево на 1 разряд через разряд переноса →dst

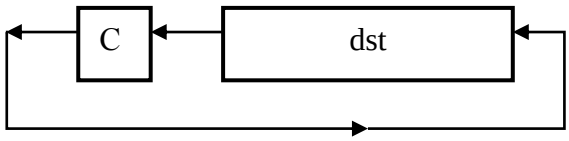
Операнды: dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31	24 23		16	15 8 7 0
0 0 0	1 0 0 1 0 0	1 1	dst	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

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:



Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

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.

Если dst не является одним из R7-R0, C сдвигается в dst, но не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример 1: ROLC R3

Before Instruction	
R3	00 0000 0420
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	1

After Instruction	
R3	00 0000 0841
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

Пример 2: ROLC R3

Before Instruction	
R3	00 8000 4281
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

After Instruction	
R3	00 0000 8502
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	1

ROR Циклический сдвиг вправо

Синтаксис: ROR dst

Операция: dst циклически сдвинутый вправо на 1 разряд через разряд переноса → dst

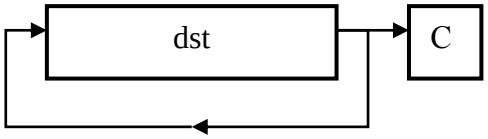
Операнды: dst - регистр (Rn, $0 \leq n \leq 27$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 | 1 0 0 1 0 1 | 1 1	dst	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	

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:



Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

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. Не изменяется, если dst не является одним из R7-R0.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: ROR R7

Before Instruction	
R7	00 0000 0421
LUF	0
LV	0
UF	0
N	0
Z	0
V	0
C	0

After Instruction	
R7	00 8000 0210
LUF	0
LV	0
UF	0
N	1
Z	0
V	0
C	1

RORC Циклический сдвиг вправо

Синтаксис: RORC dst

Операция: dst циклически сдвинутый вправо на 1 разряд через разряд переноса →dst

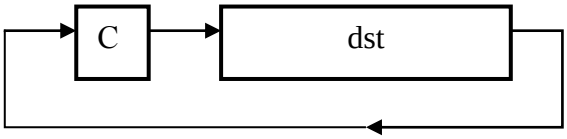
Операнды: dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

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	

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:



Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

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. Если dst не является одним из R7-R0, C сдвигается в dst, но не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: RORC R4

Before Instruction		After Instruction	
R4	00 8000 0081	R4	00 4000 0040
LUF	0	LUF	0
LV	0	LV	0
UF	0	UF	0
N	1	N	0
Z	0	Z	0
V	0	V	0
C	0	C	1

RPTB Повторение блока команд

Синтаксис: RPTB src

Операция: src → RE

1 → ST (RM)

Следующий PC → RS

Операнды: src длинный непосредственный режим адресации

Опкод:

31 24	23 16	15 8	7 0
0 1 1 0 0 1 0 0	src(смещение)		

Описание: RPTB позволяет обеспечить повторение блока команд несколько раз без потерь на зацикливание. Эта команда активизирует режим повторения при изменении PC. Операнд src – 24-разрядное непосредственное значение, которое загружается в регистр адреса конца повторений (RE). В разряд режима повторения регистра состояния ST (RM) записывается 1, указывая, что PC будет изменяться в режиме повторения. Адрес следующей команды загружается в регистр адреса начала повторения (RS).

Циклы: 4

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: RPTB 127h

	Before Instruction		After Instruction
PC	0123	PC	0124
RE	0	RE	127
RS	0	RS	124
ST	0	ST	100
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

RPTS Повторение одной инструкции

Синтаксис: RPTS src

Операция: src → RC
1 → ST (RM)
1 → S
Следующий PC → RS
Следующий PC → RE

Операнды: основные режимы адресации src (G):
00 - регистровая
01 - прямая
10 - косвенная
11 - непосредственная

Опкод:

31 24	23 16	15 8	7 0
0 0 0 | 1 0 0 1 1 1	G | 1 1 0 1 1	src	

Описание: RPTB позволяет обеспечить повторение отдельной команды несколько раз без потерь на зацикливание. Выборка из регистра команд (IR) также может производиться, что позволяет предотвращать доступ к памяти повторений. Src загружается в счётчик повторений (RC). В разряд режима повторения регистра состояния ST (RM). 1 также записывается в разряд одиночного повторения (S). Это указывает на то, что выборка программы будет выполняться только из регистра команд. Следующий PC загружается в регистр адреса конца повторения (RE) и в регистр адреса начала повторения (RS). Для непосредственного режима операнд src – беззнаковое целое без расширения знака.

Циклы: 4

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: RPTS AR5

Before Instruction		After Instruction	
AR5	00 00FF	AR5	00 00FF
PC	0123	PC	0124
RC	0	RC	0FF
RE	0	RE	124
RS	0	RS	124
ST	0	ST	100
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

SIGI Сигнал блокировки сигнализирует об операции блокировки

Синтаксис: SIGI

Операция: Операция сигнализации блокировки. Ожидание подтверждения блокировки. Очистка блокировки.

Операнды: Нет

Опкод:

31 24	23 16	15 8	7 0
0 0 0 | 1 0 1 1 0 0	G	dst	src

Описание: Операция межпроцессорного взаимодействия (блокировки) сигнализируется через XF0 и XF1. После подтверждения операции блокировки, операция блокировки завершается. SIGI игнорирует внешние сигналы готовности. См. подраздел A.6.4 для получения более детальной информации.

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 зависит от значения разряда OVM.

STF Сохранение значения с плавающей запятой

Синтаксис: STF src, dst

Операция: src → dst

Операнды: src - регистр (Rn, 0 ≤ n ≤ 7)
основные режимы адресации dst (G):
01 - прямая
10 - косвенная

Опкод:

31 24	23 16	15 8	7 0
0 0 0 | 1 0 1 0 0 0	G	src	dst

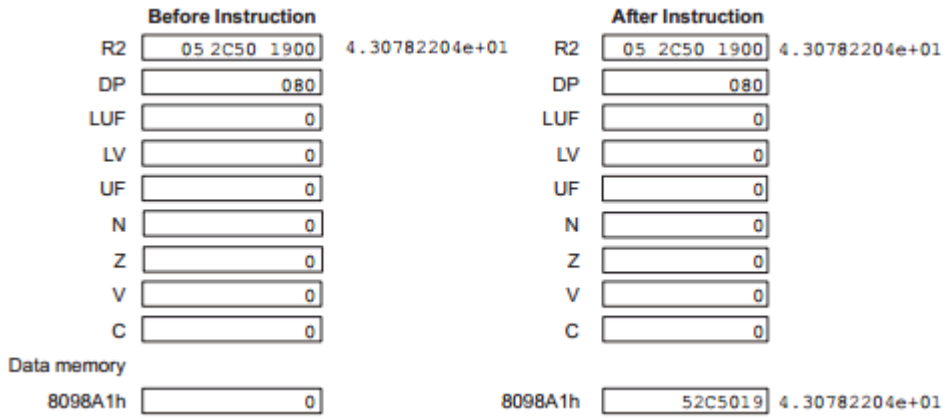
Описание: Операнд src загружается в память по адресу dst. Операнды src и dst являются числами в формате с ПЗ.

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: STF R2,@98A1h



STFI Сохранение значения с плавающей запятой, блокировка

Синтаксис: STFI src, dst

Операция: src → dst

Сигнализирует об окончании операции блокировки

Операнды: src - регистр (Rn, 0 ≤ n ≤ 7)

основные режимы адресации dst (G):

01 - прямая

10 - косвенная

Опкод:

31 24	23 16	15 8	7 0
0 0 0 | 1 0 1 0 0 1 | G	src	dst	

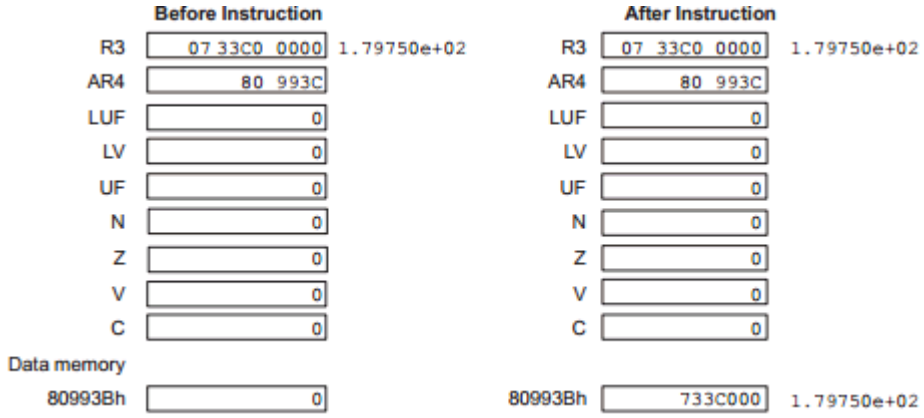
Описание: Операнд src загружается в память по адресу dst. Операция блокировки сигнализируется через выводы XF0 и XF1. Операнды src и dst являются числами в формате с ПЗ.

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: STFI R3,*-AR4



STF||STF Сохранить значения в формате с ПЗ

Синтаксис: STF src2, dst2
|| STF src1, dst1

Операция: src2 → dst2
|| 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)

Опкод:

31	24	23	16	15	8	7	0
1 1	0 0 0 0 0	src2	0 0 0	src1	dst1		dst2

Описание: Две команды STF выполняются параллельно. Оба операнда src1 и src2 являются числами в формате с ПЗ.

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:
```
STF R4,*AR3– –
|| STF R3,*++AR5
```

Before Instruction			After Instruction		
R3	07 33C0 0000	1.79750e+02	R3	07 33C0 0000	1.79750e+02
R4	07 0C80 0000	1.4050e+02	R4	07 0C80 0000	1.4050e+02
AR3	80 9835		AR3	80 9834	
AR5	80 99D2		AR5	80 99D3	
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	
Data memory					
809835h	0		809835h	070C8000	1.4050e+02
8099D3h	0		8099D3h	0733C000	1.79750e+02

STI Сохранение целого

Синтаксис: STI src, dst

Операция: src → dst

Операнды: src - регистр (Rn, 0 ≤ n ≤ 27)
основные режимы адресации dst (G):
01 - прямая
10 - косвенная

Опкод:

31	24	23	16	15	8	7	0
0 0 0	1 0 1 0 1 0	G	src	dst			

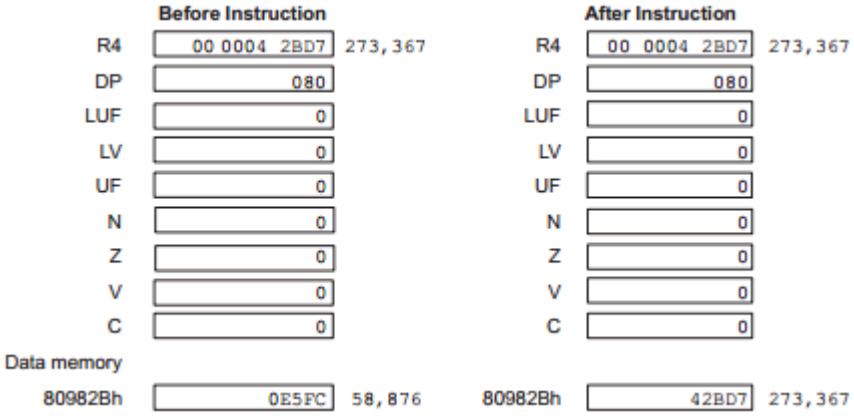
Описание: Операнд src загружается в память по адресу dst. Операнды src и dst являются целыми со знаком.

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: STI R4,@982Bh



STII Сохранение целых, блокировка

Синтаксис: STII src, dst

Операция: src → dst сигнализирует об окончании операции блокировки

Операнды: src - регистр (Rn, 0 ≤ n ≤ 27)
основные режимы адресации dst (G):
01 - прямая
10 - косвенная

Опкод:

31 24	23 16	15 8	7 0
0 0 0 1 0 1 1 1	G	src	dst

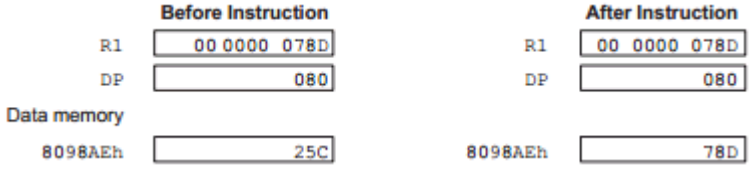
Описание: Операнд src загружается в память по адресу dst. Операция блокировки сигнализируется через выводы XF0 и XF1. Операнды src и dst являются целыми со знаком.

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: STII R1,@98AEh



STI||STI Сохранить целые

Синтаксис: STI src2, dst2
|| STI src1, dst1

Операция: src2 → dst2
|| 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)

Опкод:

31 24	23 16	15 8	7 0
1 1 | 0 0 0 0 1 | src2	0 0 0 | src1	dst1	dst2

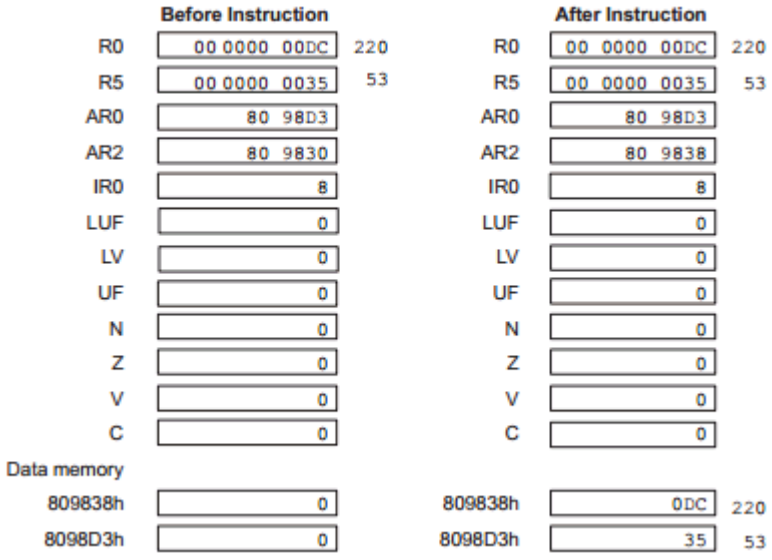
Описание: Сохранение двух целых производится параллельно. Если обе операции сохранения выполняются по одному адресу, значение записывается как при выполнении операции STI src2, src2.

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: STI R0, *++AR2 (IR0)
|| STI R5, *AR0



SUBB Вычитание целых с заёмом

Синтаксис: SUBB src, dst

Операция: dst-src-C → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, 0 ≤ n ≤ 27)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 | 1 0 1 1 0 1 | G	dst	src	

Описание: Разность между операторами dst, src и C загружается в регистр dst. Предполагается, что операнды dst являются целыми со знаком.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

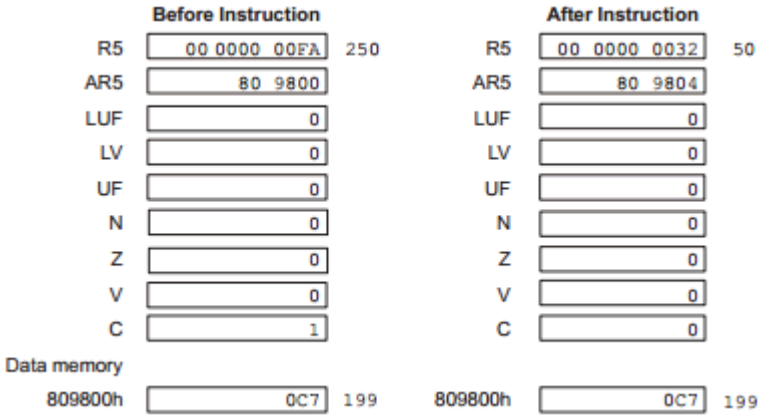
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

C 1 при возникновении заёма, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример: SUBB *AR5++ (4), R5



SUBB3 Вычитание целых с заёмом (3 операнда)

Синтаксис: SUBB3 src2, src1, dst

Операция: src1-src2-C → dst

Операнды: Трёхоперандные режимы адресации src1 (T):

00 - регистровая (Rn1, $0 \le n1 \le 27$)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn1, $0 \le n1 \le 27$)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Трёхоперандные режимы адресации src2 (T):

00 - регистровая (Rn2, $0 \le n2 \le 27$)

01 - регистровая (Rn2, $0 \le n2 \le 27$)

10 - косвенная (смещение = 0, 1, IR0, IR1)

11 - косвенная (смещение = 0, 1, IR0, IR1)

dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 24	23 16	15 8	7 0
0 0 1 0 0 1 1 0 0	T	dst	src1	src2

Описание: Разность между операндами src1 и src2, и флагом C (перенос) загружается в регистр dst. Операнды src1, src2 и dst являются целыми со знаком.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

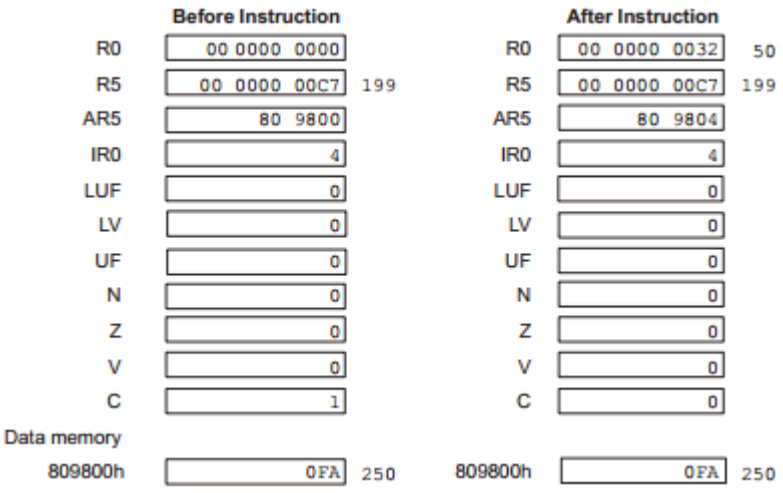
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

C 1 при возникновении переноса, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример: SUBB3 R5, *AR5++ (IR0), R0



SUBC Условное вычитание целых

Синтаксис: SUBC src, dst

Операция: Если (dst - src ≥ 0):

(dst - src << 1) или 1 → dst.

Иначе:

dst << 1 → dst.

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, 0 ≤ n ≤ 27)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 24	23 16	15 8	7 0
0 0 1 | 1 0 1 1 1 0	G	dst	src

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.

Циклы: 1

Разряды состояния: LUF Не изменяется.

LV Не изменяется.

UF Не изменяется.

N Не изменяется.

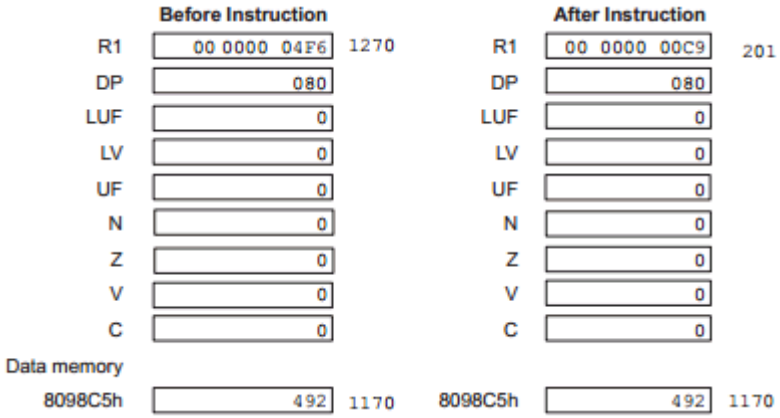
Z Не изменяется.

V Не изменяется.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример 1: SUBC @98C5h, R1



Пример 2: SUBC 3000, R0 (3000 = 0BB8h)

	Before Instruction		After Instruction	
R0	00 0000 07D0	2000	00 0000 0FA0	4000
LUF	0		0	
LV	0		0	
UF	0		0	
N	0		0	
Z	0		0	
V	0		0	
C	0		0	

SUBF Условное вычитание значений с плавающей запятой

Синтаксис: SUBF src, dst

Операция: dst - src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 7$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 7$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 | 1 0 1 1 1 1	G	dst	src

Описание: Разность операнда dst минус операнд src загружается в регистр dst. Операнды dst и src являются числами в формате с ПЗ.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 1 при возникновении отрицательного переполнения, 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.

Пример: SUBF *AR0– – (IR0), R5

	Before Instruction		After Instruction	
R5	07 33C0 0000	1.79750000e+02	05 1D00 0000	3.9250e+01
AR0	80 9888		80 9808	
IR0	80		80	
LUF	0		0	
LV	0		0	
UF	0		0	
N	0		0	
Z	0		0	
V	0		0	
C	0		0	
Data memory				
809888h	70C8000	1.4050e+02	809888h 70C8000	1.4050e+02

SUBF3 Вычитание значений с плавающей запятой (3 операнда)

Синтаксис: SUBF3 src2, src1, dst

Операция: src1 - src2 → dst

Операнды: Трёхоперандные режимы адресации src1 (T):

00 - регистровая (Rn1, 0 ≤ n1 ≤ 7)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn1, 0 ≤ n1 ≤ 7)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Трёхоперандные режимы адресации src2 (T):

00 - регистровая (Rn2, 0 ≤ n2 ≤ 7)

01 - регистровая (Rn2, 0 ≤ n2 ≤ 7)

10 - косвенная (смещение = 0, 1, IR0, IR1)

11 - косвенная (смещение = 0, 1, IR0, IR1)

dst - регистр (Rn, 0 ≤ n ≤ 7)

Опкод:

31 24	23 16	15 8	7 0
0 0 1 | 0 0 1 1 0 1 | T	dst	src1	src2

Описание: Разность между операндами src1 и src2 загружается в регистр dst. Операнды dst, src1 и src2 являются числами в формате с ПЗ.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 1 при возникновении отрицательного переполнения, 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.

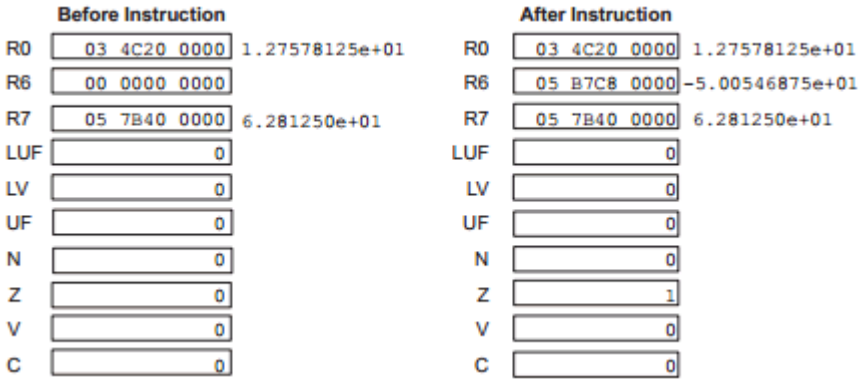
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример 1: SUBF3 *AR0– – (IR0), *AR1, R4

Before Instruction			After Instruction		
R4	00 0000 0000		R4	05 1D00 0000	3.9250e+01
AR0	80 9888		AR0	80 9808	
AR1	80 9851		AR1	80 9851	
IR0	80		IR0	80	
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	
Data memory					
809888h	70C8000	1.4050e+02	809888h	70C8000	1.4050e+02
809851h	733C000	1.79750e+02	809851h	733C000	1.79750e+02

Пример 2: SUBF3 R7, R0, R6



SUBF3||STF Вычесть значение в формате с ПЗ и сохранить значение в формате с ПЗ

Синтаксис: SUBF3 src1, src2, dst1
|| STF src3, dst2

Операция: src2 - src1 → dst1
|| src3 → dst2

Операнды: src1 - регистр (Rn1, 0 ≤ n1 ≤ 7)
src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn2, 0 ≤ n2 ≤ 7)
src3 - регистр (Rn3, 0 ≤ n3 ≤ 7)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 24	23 16	15 8	7 0
1 1 | 1 | 0 1 0 1 | dst1	src1 | src3	dst2	src2

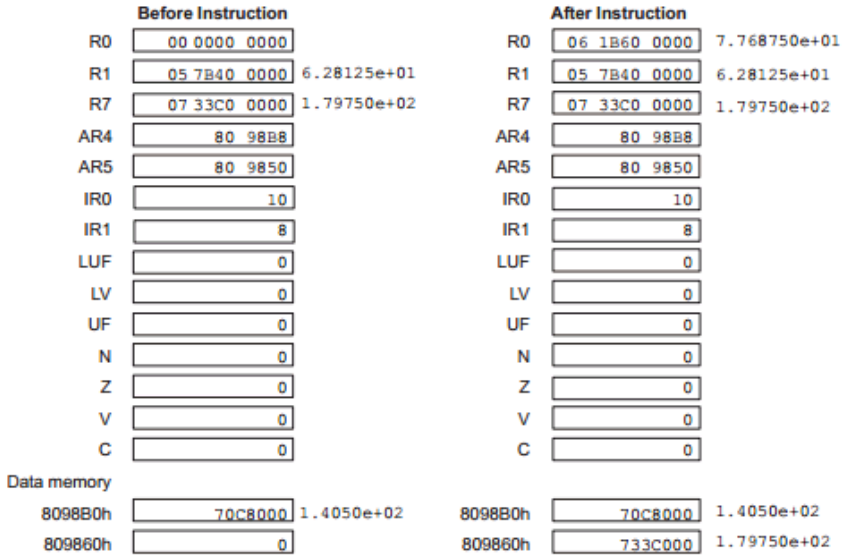
Описание: Вычитание в формате с ПЗ и сохранение числа в формате с ПЗ выполняется параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STF) считывает из регистра, и операция, которая производится параллельно (SUBF3), записывает в тот же регистр, STF имеет на входе содержимое регистра до того, как оно было изменено командой SUBF3. Если src3 и dst1 указывают на один и тот же адрес, src3 считывается до записи в dst1.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7 - R0.
LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.
LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.
UF 1 при возникновении отрицательного переполнения, 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.

Пример: SUBF3 R1, *-AR4 (IR1), R0
|| STF R7, *+AR5 (IR0)



SUBI Вычитание целых

Синтаксис: SUBI src, dst

Операция: dst-src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 | 1 1 0 0 0 0	G	dst	src

Описание: Разность операнда dst минус операнд src загружается в регистр dst. Операнды src и dst являются целыми со знаком.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

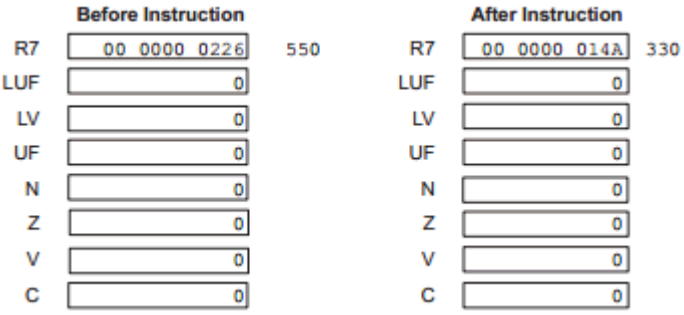
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

C 1 при возникновении заёма, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример: SUBI 220, R7



SUBI3 Вычитание целых (3 операнда)

Синтаксис: SUBI3 src2, src1, dst

Операция: src1 - src2 → dst

Операнды: Трёхоперандные режимы адресации src1 (T):

00 - регистровая (Rn1, $0 \le n1 \le 27$)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn1, $0 \le n1 \le 27$)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Трёхоперандные режимы адресации src2 (T):

00 - регистровая (Rn2, $0 \le n2 \le 27$)

01 - регистровая (Rn2, $0 \le n2 \le 27$)

10 - косвенная (смещение = 0, 1, IR0, IR1)

11 - косвенная (смещение = 0, 1, IR0, IR1)

dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 24	23 16	15 8	7 0
0 0 1 | 0 0 1 1 1 0 | T	dst	src	src

Описание: Разность операнда src1 минус операнд src2 загружается в регистр dst. Операнды src1, src2 и dst являются целыми со знаком.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.

UF 0

N 1 при генерации отрицательного результата, иначе 0.

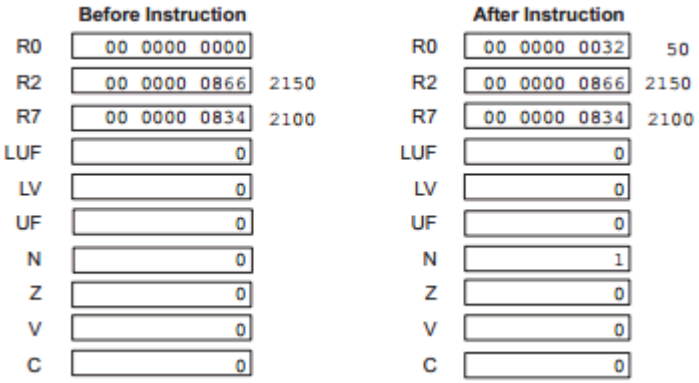
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении целочисленного переполнения, иначе 0.

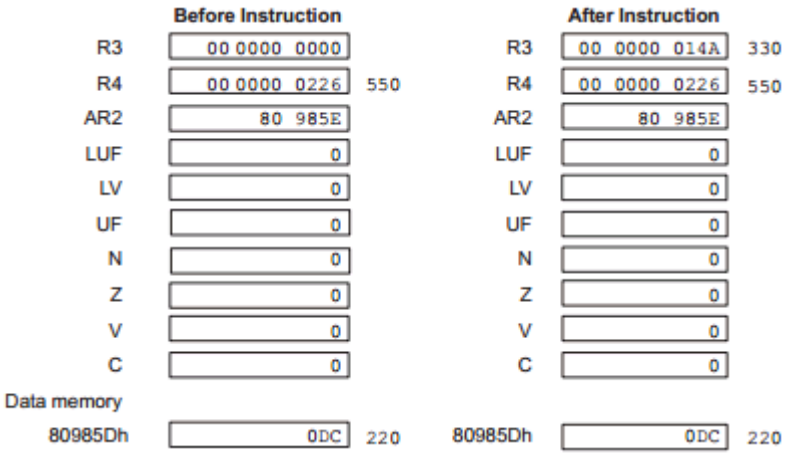
C 1 при возникновении замена, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример 1: SUBI3 R7, R2, R0



Пример 2: SUBI3 *-AR2 (1), R4, R3



SUBI3||STI Вычесть целое и сохранить целое

Синтаксис: SUBI3 src1, src2, dst1
|| STI src3, dst2

Операция: src2 - src1 → dst1
|| src3 → dst2

Операнды: src1 - регистр (Rn1, 0 ≤ n1 ≤ 7)
src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn2, 0 ≤ n2 ≤ 7)
src3 - регистр (Rn3, 0 ≤ n3 ≤ 7)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 24	23 16	15 8	7 0
1 1 1 1 0 1 1 0	dst1 src1	src3 dst2	src2

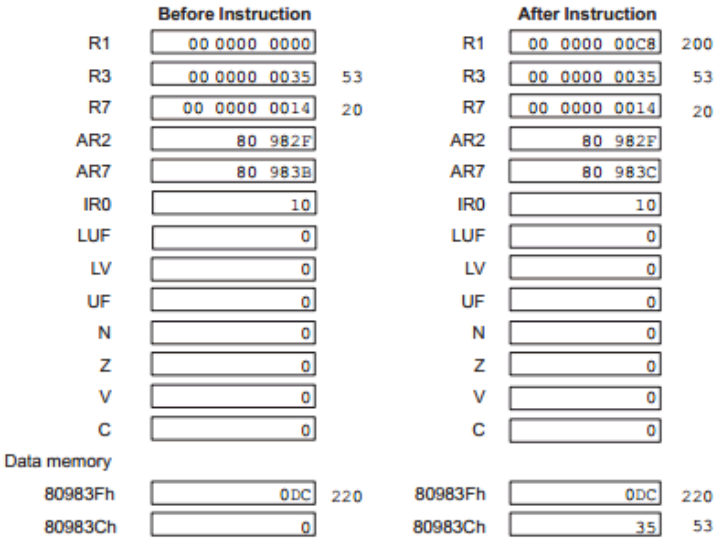
Описание: Целочисленное вычитание и сохранение целого производится параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STI) считывает из регистра, и операция, которая производится параллельно (SUBI3), записывает в тот же регистр, STI имеет на входе содержимое регистра до того, как оно было изменено командой SUBI3. Если src3 и dst1 указывают на один и тот же адрес, src3 считывается до записи в dst1.

Циклы: 1

Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7 - R0.
LUF Не изменяется.
LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF 0
N 1 при генерации отрицательного результата, иначе 0.
Z 1 при генерации нулевого результата, иначе 0.
V 1 при возникновении целочисленного переполнения, иначе 0.
C 1 при возникновении заёма, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример:
```
SUBI3 R7, *+AR2 (IR0), R1
|| STI R3, *++AR7
```



SUBRB Вычесть целое и сохранить целое

Синтаксис: SUBRB src, dst

Операция: src - dst - C → dst

Операнды: основные режимы адресации src (G):
00 - регистровая (Rn, 0 ≤ n ≤ 27)
01 - прямая
10 - косвенная
11 - непосредственная
dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 | 1 1 0 0 0 1	G	dst	src

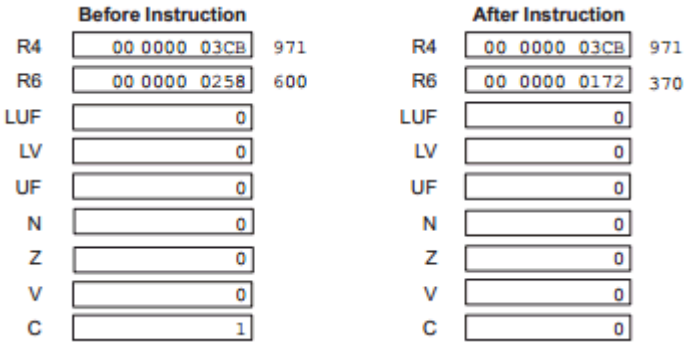
Описание: Разность операндов src, dst и C загружается в регистр dst. Операнды dst и src являются целыми со знаком.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.
LUF Не изменяется.
LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF 0
N 1 при генерации отрицательного результата, иначе 0.
Z 1 при генерации нулевого результата, иначе 0.
V 1 при возникновении целочисленного переполнения, иначе 0.
C 1 при возникновении заёма, иначе 0.

Разряд режима: OVM - операция зависит от значения разряда OVM.

Пример: SUBRB R4, R6



SUBRF Вычитание значений с плавающей запятой в обратном порядке

Синтаксис: SUBRF src, dst

Операция: src - dst → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 7$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 7$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 1 1 0 0 1 0 G	dst	src	

Описание: Разность операнда src минус операнд dst загружается в регистр dst. Операнды dst и src являются числами в формате с ПЗ.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF 1 при возникновении отрицательного переполнения, иначе не изменяется.

LV 1 при возникновении ПЗ-переполнения, в противном случае не меняется.

UF 1 при возникновении отрицательного переполнения, иначе 0.

N 1 при генерации отрицательного результата, иначе 0.

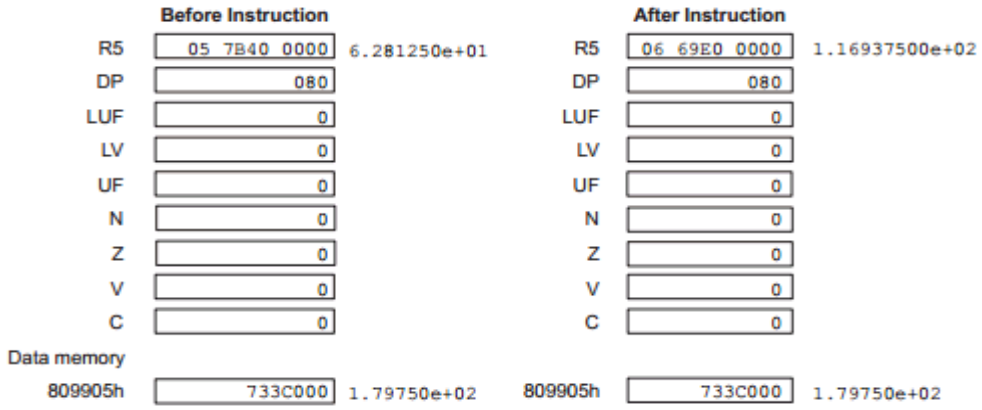
Z 1 при генерации нулевого результата, иначе 0.

V 1 при возникновении ПЗ-переполнения, иначе 0.

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: SUBRF @9905h, R5



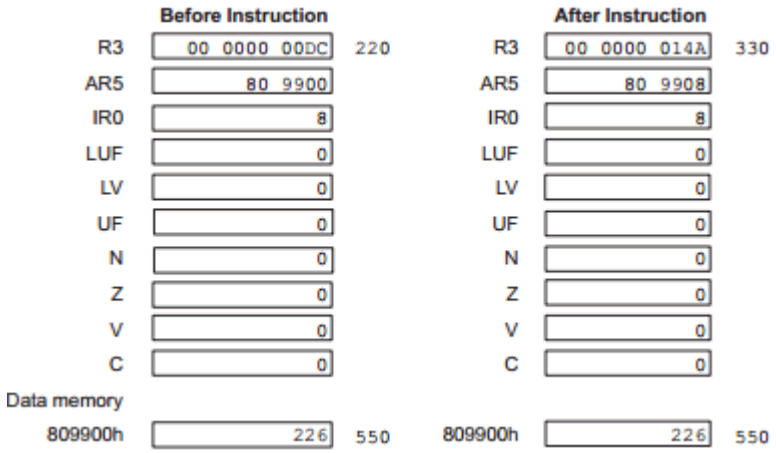
SUBRI Вычитание целых в обратном порядке

Синтаксис: SUBRI src, dst
Операция: src - dst → dst
Операнды: основные режимы адресации src (G):
00 - регистровая (Rn, 0 ≤ n ≤ 27)
01 - прямая
10 - косвенная
11 - непосредственная
dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 1 1 0 0 1 1	G	dst	src

Описание: Разность операнда src минус операнд dst загружается в регистр dst. Операнды dst и src являются целыми со знаком.
Циклы: 1
Разряды состояния: Флаги состояния изменяются только в том случае, если регистр назначения – один из R7 - R0.
LUF Не изменяется.
LV 1 при возникновении целочисленного переполнения, в противном случае не меняется.
UF 0
N 1 при генерации отрицательного результата, иначе 0.
Z 1 при генерации нулевого результата, иначе 0.
V 1 при возникновении целочисленного переполнения, иначе 0.
C 1 при возникновении заёма, иначе 0.
Разряд режима: OVM - операция зависит от значения разряда OVM.
Пример: SUBRI *+AR5++ (IR0), R3



SWI Программное прерывание

Синтаксис: SWI

Операция: Выполняет прерывание эмуляции

Операнды: Нет

Опкод:

31 24	23 16	15 8	7 0
0 1 1 0 0 1 1 0 0	0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	

Описание: Команда SWI выполняет прерывание эмуляции. Эта команда является резервной и не может быть использована в обычном программировании.

Циклы: 4

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.

TRAPcond Программное прерывание

Синтаксис: TRAPcond N

Операция: 0 → ST (GIE)
Если условие(cond) "истина":
Следующий PC → *++SP,
Вектор системного прерывания N → PC.
Иначе:
Установить GIE в начальное состояние.
Продолжение.

Операнды: N ($0 \leq N \leq 31$)

Опкод:

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 0 0 0 0	N

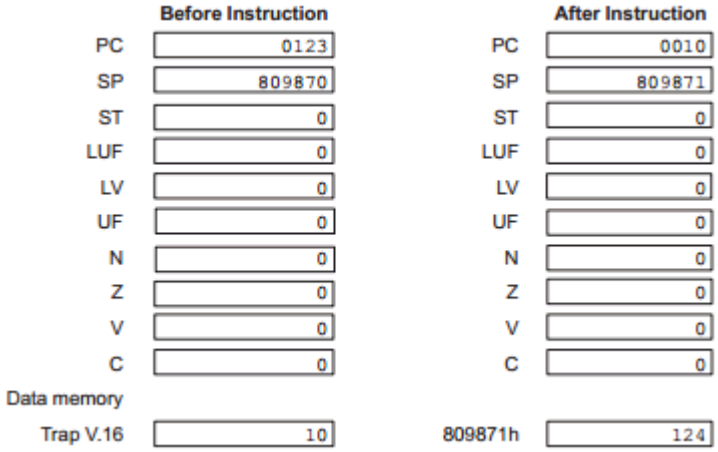
Описание: Все прерывания запрещаются, когда 0 записывается в ST(GIE). Если условие истинно, содержимое счётчика команд (PC) загружается в системный стек, а в PC загружается содержимое указанного вектора системного прерывания (N). Если условие не "истина", ST(GIE) устанавливается по значению, которое он имел до того, как команда TRAPcond его изменила. В ПЦОС имеются 20 кодов условий, которые могут быть использованы с этой команды (в подразделе A.10.2 приведён список мнемоник условий, коды и флаги). Флаги условий устанавливаются предыдущей командой, только когда регистром назначения является один из регистров повышенной точности (R7 - R0), либо когда выполняется одна из команд сравнения (CMPF, CMPF3, CMPI, CMPI3, TSTB или TSTB3).

Циклы: 5

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: TRAPZ 16



TSTB Проверка битовых полей

Синтаксис: TSTB src, dst

Операция: dst AND src (dst И src)

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 1 1 0 1 0 0 G	dst	src	

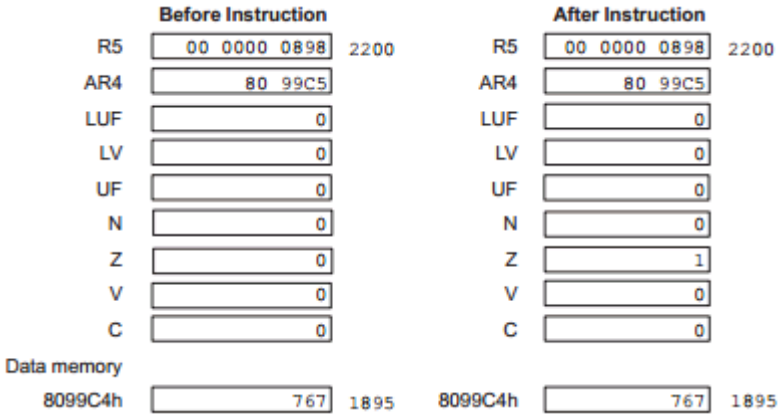
Описание: Поразрядное логическое И операндов dst и src формируется, но результат не загружается ни в один регистр. Таким образом, обеспечивается неразрушающее сравнение. Предполагается, что операнды являются беззнаковыми целыми.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.
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.

Пример: TSTB *-AR4 (1), R5



TSTB3 Проверка битовых полей (3 операнда)

Синтаксис: TSTB3 src2, src1

Операция: src1 AND src2

Операнды: Трёхоперандные режимы адресации src1 (T):

00 - регистровая (Rn1, $0 \le n1 \le 27$)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn1, $0 \le n1 \le 27$)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Трёхоперандные режимы адресации src2 (T):

00 - регистровая (Rn2, $0 \le n2 \le 27$)

01 - регистровая (Rn2, $0 \le n2 \le 27$)

10 - косвенная (смещение = 0, 1, IR0, IR1)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 24	23 16	15 8	7 0
0 0 1 | 0 0 1 1 1 1 | T	0 0 0 0 0	src1	src1

Описание: Поразрядное логическое И операндов src1 и src1 формируется, но результат не загружается ни в один регистр. Таким образом, обеспечивается сравнение без потери информации. Предполагается, что операнды являются беззнаковыми целыми. Хотя данная команда имеет только два операнда, она обозначается как трёхоперандная, т. к. операнды определены в трёхоперандном формате.

Циклы: 1

Разряды состояния: LUF Не изменяется.

LV Не изменяется.

UF 0

N Старший значащий разряд выходного значения.

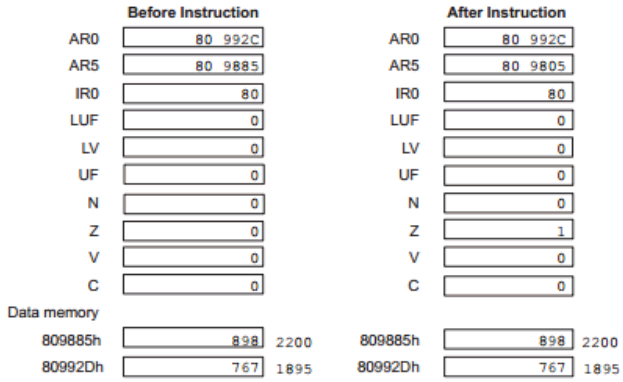
Z 1 при генерации нулевого результата, иначе 0.

V 0

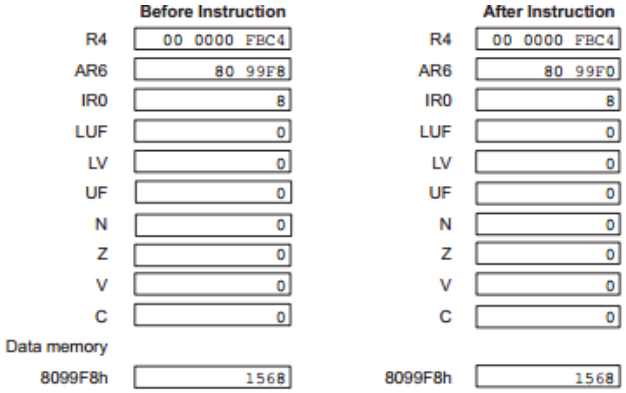
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример 1: TSTB3 *AR5– – (IR0), *+AR0 (1)



Пример 2: TSB3 R4, *AR6– – (IR0)



XOR Поразрядное исключающее ИЛИ

Синтаксис: XOR src, dst

Операция: dst XOR src → dst

Операнды: основные режимы адресации src (G):

00 - регистровая (Rn, $0 \le n \le 27$)

01 - прямая

10 - косвенная

11 - непосредственная

dst - регистр (Rn, $0 \le n \le 27$)

Опкод:

31 24	23 16	15 8	7 0
0 0 0 | 1 1 0 1 0 1	G	dst	src

Описание: Поразрядное исключающее-ИЛИ операндов src и dst загружается в регистр dst. Операнды dst и src являются беззнаковыми целыми.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N Старший значащий разряд выходного значения.

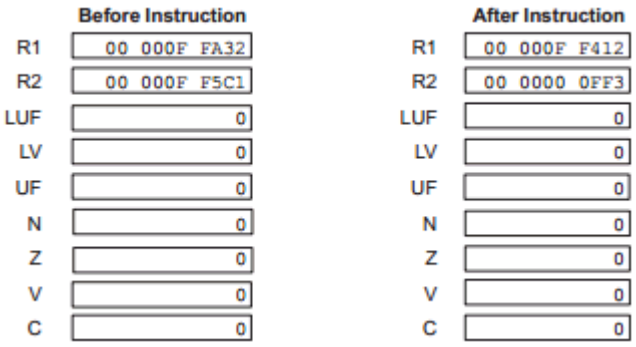
Z 1 при генерации нулевого результата, иначе 0.

V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример: XOR R1, R2



XOR3 Поразрядное исключающее ИЛИ (3 операнда)

Синтаксис: XOR3 src2, src1, dst

Операция: src1 XOR src2 → dst

Операнды: Трёхоперандные режимы адресации src1 (T):

00 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

01 - косвенная (смещение = 0, 1, IR0, IR1)

10 - регистровая (Rn1, 0 ≤ n1 ≤ 27)

11 - косвенная (смещение = 0, 1, IR0, IR1)

Трёхоперандные режимы адресации src2 (T):

00 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

01 - регистровая (Rn2, 0 ≤ n2 ≤ 27)

10 - косвенная (смещение = 0, 1, IR0, IR1)

11 - косвенная (смещение = 0, 1, IR0, IR1)

dst - регистр (Rn, 0 ≤ n ≤ 27)

Опкод:

31 24	23 16	15 8	7 0
0 0 1 | 0 1 0 0 0 0 | T	dst	src1	src2

Описание: Поразрядное исключающее ИЛИ между операндами src1 и src2 загружается в регистр dst. Операнды src1, src2 и dst являются беззнаковыми целыми.

Циклы: 1

Разряды состояния: LUF Не изменяется.

LV Не изменяется.

UF 0

N Старший значащий разряд выходного значения.

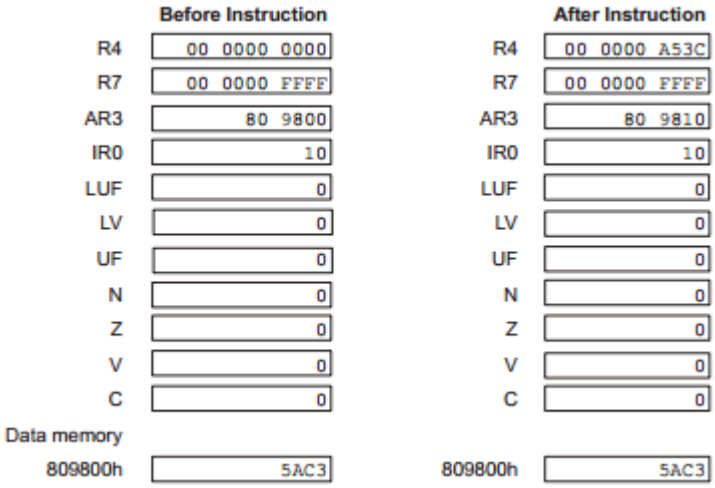
Z 1 при генерации нулевого результата, иначе 0.

V 0

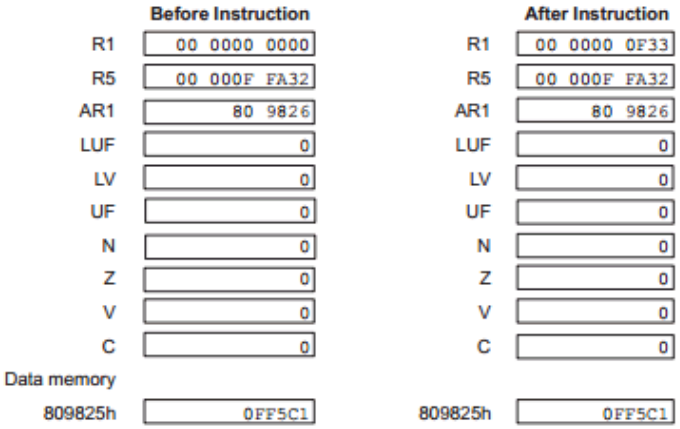
C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример 1: XOR3 *AR3++ (IR0), R7, R4



Пример 2: XOR3 R5, *-AR1 (1), R1



Примечание – См. А.9.5.2 для выяснения влияния следования операций на счётчик циклов.

XOR3||STI Поразрядное ИСКЛЮЧАЮЩЕЕ ИЛИ значений и сохранить целое

Синтаксис: XOR3 src2, src1, dst1
|| STI src3, dst2

Операция: src1 XOR src2 → dst1
|| src3 → dst2

Операнды: src1 - регистр (Rn1, 0 ≤ n1 ≤ 7)
src2 - косвенная (смещение = 0, 1, IR0, IR1)
dst1 - регистр (Rn2, 0 ≤ n2 ≤ 7)
src3 - регистр (Rn3, 0 ≤ n3 ≤ 7)
dst2 - косвенная (смещение = 0, 1, IR0, IR1)

Опкод:

31 24	23 16	15 8	7 0
1 1 1 0 1 1 1 1 dst1	src1	src3	dst2	src2

Описание: Поразрядное ИСКЛЮЧАЮЩЕЕ ИЛИ (XOR3) и сохранение целого (STI) выполняется параллельно. Все регистры считываются в начале и загружаются в конце цикла исполнения. Это означает, что если одна из параллельных операций (STI) считывает из регистра, и операция, которая производится параллельно (XOR3), записывает в тот же регистр, STI

имеет на входе содержимое регистра до того, как оно было изменено командой XOR3. Если src2 и dst2 указывают на один и тот же адрес, src2 считывается до записи в dst2.

Циклы: 1

Разряды состояния: Флаги состояния изменяются тогда, когда регистр назначения – один из R7 - R0.

LUF Не изменяется.

LV Не изменяется.

UF 0

N Старший значащий разряд выходного значения.

Z 1 при генерации нулевого результата, иначе 0.

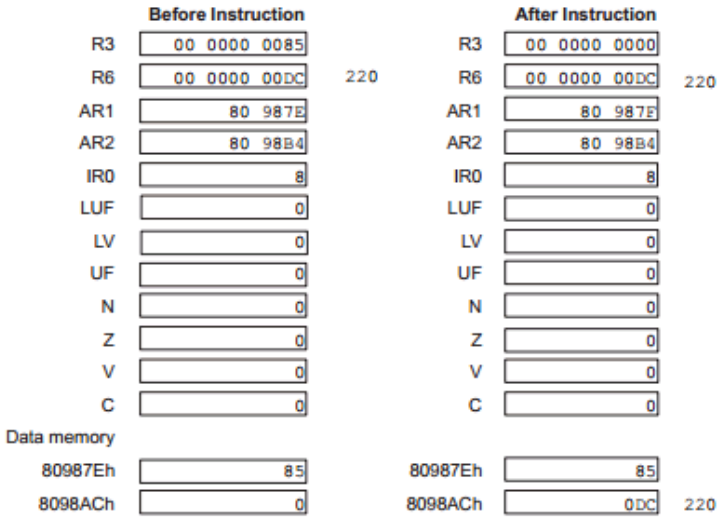
V 0

C Не изменяется.

Разряд режима: OVM - операция не зависит от значения разряда OVM.

Пример:

```
XOR3        *AR1++, R3, R3
|| STI R6,   *-AR2 (IR0)
```



Лист регистрации изменений								
Изм.	Номера листов (страниц)				Всего листов (страниц) в документе	Номер документа	Подпись	Дата
	измененных	замененных	новых	аннулированных				
-	-	-	Все	-	468	10195		25.03.20
1	1	-	-	-	-	10196		27.03.20