

# **МИКРОСХЕМА ИНТЕГРАЛЬНАЯ 1892ВМ3Т**

**РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

## Перечень сокращений:

- **CPU** – центральный процессор на основе RISC-ядра;
- **CRAM** – двухпортовая оперативная память центрального процессора;
- **DSP** – сопроцессор цифровой обработки сигналов с фиксированной точкой (далее может называться также **ЦПОС** – цифровой процессор обработки сигналов);
- **DMA** – контроллер прямого доступа в память;
- **MPORT** – порт внешней памяти;
- **SPORT** – последовательный порт;
- **LPORТ** – линковый порт;
- **UART** – универсальный асинхронный порт;
- **ICACHE** – кэш программ центрального процессора;
- **IT** – интервальный таймер;
- **WDT** – сторожевой таймер;
- **RTT** – таймер реального времени;
- **CDB[31:0]** – шина данных CPU;
- **DDV[31:0]** – шина данных DMA;
- **A[31:0]** – шина адреса порта внешней памяти;
- **D[31:0]** – шина данных порта внешней памяти;
- **OnCD** – встроенные средства отладки программ;
- **XRAM, YRAM** – памяти данных DSP;
- **PRAM** – память программ DSP;
- **AGU** – адресный генератор;
- **EDBS** – коммутатор внешних шин;
- **IDBS** – коммутатор внутренних шин;
- **PCU** – устройство программного управления;
- **PAG** – генератор адреса программ;
- **PDC** – программный дешифратор;
- **RF** – регистровый файл;
- **ALU** – арифметическое устройство;
- **ALUctr** – управление ALU;
- **XDB0 – XDB3, GDB, PDB** – шина данных DSP;
- **XAB, YAB, PAB** – адресные шины DSP;
- **M, S, A, L** – арифметические узлы ALU DSP.

## СОДЕРЖАНИЕ

<b>1. ВВЕДЕНИЕ .....</b>	<b>9</b>
1.1 Порядок использования данного документа .....	9
1.2 Назначение .....	9
1.3 Функциональные параметры и возможности .....	10
1.4 Структурная схема .....	15
1.5 Инструментальное программное обеспечение .....	18
1.6 Операционная система для микросхемы 1892ВМ3Т .....	19
<b>2. ЦЕНТРАЛЬНЫЙ ПРОЦЕССОР .....</b>	<b>20</b>
2.1 Основные характеристики CPU .....	20
2.2 Блок-схема .....	20
2.3 Составляющие логические блоки .....	21
2.3.1 Устройство исполнения .....	21
2.3.2 Устройство умножения/деления (MDU) .....	22
2.3.3 Системный управляющий сопроцессор .....	22
2.3.4 Устройство управления памятью (MMU) .....	22
2.3.5 Контроллер кэш .....	22
2.3.6 Устройство шинного интерфейса (BIU – Bus Interface Unit) .....	22
2.3.7 OnCD контроллер .....	22
2.4 Конвейер .....	22
2.4.1 Стадии конвейера .....	23
2.4.1.1 Стадия I: выборка команды .....	23
2.4.1.2 Стадия D: дешифрация команды .....	24
2.4.1.3 Стадия E: исполнение .....	24
2.4.1.4 Стадия M: выборка из памяти .....	24
2.4.1.5 Стадия W: обратная запись .....	24
2.4.2 Операции умножения и деления .....	24
2.4.3 Задержка выполнения команд перехода (Jump, Branch) .....	24
2.4.4 Обходные пути передачи данных (Data bypass) .....	25
2.4.5 Задержка загрузки данных .....	26
2.5 Устройство управления памятью (MMU) .....	27
2.5.1 Введение .....	27
2.5.2 Режимы работы .....	28
2.5.2.1 Виртуальные сегменты памяти .....	29
2.5.2.2 Режим User .....	30
2.5.2.3 Режим Kernel .....	31
2.5.3 Буфер быстрого преобразования адреса (TLB) .....	34
2.5.4 Преобразование виртуального адреса в физический в режиме TLB .....	37
2.5.4.1 Попадания (hits), промахи (misses), и множественные попадания (multiple matches) .....	39
2.5.4.2 Размеры страниц и алгоритм замещения .....	40
2.6 Исключения .....	42
2.6.1 Условия исключений .....	42
2.6.2 Приоритеты исключений .....	42
2.6.3 Расположение векторов исключений .....	43
2.6.4 Обработка общих исключений .....	45
2.6.5 Исключения .....	46
2.6.5.1 Исключение по аппаратному сбросу (Reset Exception) .....	46

2.6.5.2	Исключение по немаскируемому прерыванию (Non Maskable Interrupt – NMI Exception) 47	47
2.6.5.3	Исключение по обновлению TLB — выборка команды или доступ к данным (TLB Refill Exception – Instruction Fetch or Data Access) .....	47
2.6.5.4	Исключение TLB Invalid — выборка команды или доступ к данным (TLB Invalid Exception – Instruction Fetch or Data Access) .....	48
2.6.5.5	Исключение по ошибке адресации — выборка команды / доступ к данным (Address Error Exception – Instruction Fetch / Data Access) .....	48
2.6.5.6	Исключение по аппаратному контролю (Mcheck – Machine Check Exception) .....	49
2.6.5.7	Исключение исполнения – системный вызов (System Call Exception) .....	49
2.6.5.8	Исключение исполнения — Breakpoint (Execution Exception – Breakpoint) .....	50
2.6.5.9	Исключение исполнения — зарезервированная команда (Execution Exception – Reserved Instruction) .....	50
2.6.5.10	Исключение исполнения — недоступен сопроцессор (Execution Exception – Coprocessor Unusable) .....	50
2.6.5.11	Исключение исполнения — целочисленное переполнение (Execution Exception – Integer Overflow) .....	51
2.6.5.12	Исключение исполнения — Trap (Execution Exception – Trap) .....	51
2.6.5.13	Исключение сохранения в запрещенной области (TLB Modified Exception) .....	51
2.6.5.14	Исключение прерывания (Interrupt Exception) .....	52
2.6.6	Алгоритмы обработки исключений .....	52
2.7	Регистры CP0 .....	56
2.7.1	Назначение .....	56
2.7.2	Обзор регистров CP0 .....	56
2.7.3	Регистры CP0 .....	57
2.7.3.1	Регистр Index (Регистр 0 CP0, Select 0) .....	59
2.7.3.2	Регистр Random (Регистр CP0 1, Select 0) .....	60
2.7.3.3	EntryLo0, EntryLo1 (Регистры 2 и 3 CP0, Select 0) .....	61
2.7.3.4	Регистр Context (Регистр 4 CP0, Select 0) .....	63
2.7.3.5	Регистр PageMask (Регистр 5 CP0, Select 0) .....	63
2.7.3.6	Регистр Wired (Регистр 6 CP0, Select 0) .....	64
2.7.3.7	Регистр BadVAddr (Регистр 8 CP0, Select 0) .....	65
2.7.3.8	Регистр Count (Регистр 9 CP0, Select 0) .....	66
2.7.3.9	Регистр EntryHi (Регистр 10 CP0, Select 0) .....	66
2.7.3.10	Регистр Compare (Регистр 11 CP0, Select 0) .....	67
2.7.3.11	Регистр Status (Регистр 12 CP0, Select 0) .....	68
2.7.3.12	Регистр Cause (Регистр 13 CP0, Select 0) .....	71
2.7.3.13	Регистр EPC (Регистр 14 CP0, Select 0) .....	72
2.7.3.14	Регистр PRId (Регистр 15 CP0, Select 0) .....	73
2.7.3.15	Регистр Config (Регистр 16 CP0, Select 0) .....	74
2.7.3.16	Регистр Config1 (Регистр 16 CP0, Select 1) .....	75
2.7.3.17	Регистр LLAddr – Load Linked Address (Регистр 17 CP0, Select 0) .....	75
2.7.3.18	Регистр ErrorEPC (Регистр 30 CP0, Select 0) .....	76
2.8	Кэш .....	77
2.8.1	Введение .....	77
2.8.2	Протокол кэш .....	77
2.8.2.1	Организация кэш .....	77
2.8.2.2	Атрибуты кэшируемости .....	77
2.9	Карта памяти CPU .....	77
<b>3.</b>	<b>ЦИФРОВОЙ СИГНАЛЬНЫЙ ПРОЦЕССОР (DSP) .....</b>	<b>85</b>
3.1	Функциональные параметры и возможности DSP .....	85
3.2	Архитектура DSP .....	86
3.2.1	Структурная схема DSP .....	86
3.2.2	Арифметико-логическое устройство (ALU) .....	87

3.2.3	Устройства генерации адреса (AGU, AGU-Y).....	88
3.2.4	Устройство программного управления (PCU).....	89
3.2.5	Коммутаторы шин данных (IDBS, EDBS) .....	89
3.2.6	Блоки памяти.....	89
3.2.7	Шины адреса и данных .....	90
3.3	Арифметико-логическое устройство (ALU).....	91
3.3.1	Архитектура ALU .....	91
3.3.1.1	Регистровый файл .....	93
3.3.1.2	Операционные устройства .....	94
3.3.1.3	Регистр PDNR.....	96
3.3.1.4	Регистр CCR .....	96
3.3.1.5	Регистры-аккумуляторы AC0, AC1 .....	99
3.3.2	Режимы работы ALU.....	99
3.3.2.1	Округление (Rounding) .....	99
3.3.2.2	Масштабирование (Scaling).....	100
3.3.2.3	Поддержка режима блочной экспоненты .....	101
3.3.2.4	Режим насыщения (Saturation).....	102
1.1.1.2	Обработка режима насыщения.....	102
3.4	Устройства генерации адресов памяти данных (AGU,AGU-Y) .....	103
3.4.1	Архитектура AGU .....	103
3.4.2	Программная модель AGU. ....	105
3.4.2.1	Адресный регистровый файл .....	105
3.4.2.2	Регистровый файл смещений.....	105
3.4.2.3	Регистровый файл модификаторов.....	105
3.4.3	Архитектура AGU-Y .....	106
3.4.4	Программная модель AGU-Y .....	107
3.4.5	Виды адресации .....	107
3.4.5.1	Прямая регистровая адресация .....	108
3.4.5.2	Виды адресации программной памяти.....	109
3.4.5.3	Косвенная адресация памяти данных.....	109
3.4.6	Типы адресной арифметики .....	110
3.4.7	Режимы адресации .....	112
3.4.7.1	Режимы адресации AGU .....	112
3.4.7.2	Режимы адресации AGU-Y .....	113
3.5	Устройство программного управления (PCU) .....	113
3.5.1	Назначение и состав PCU .....	113
3.5.2	Архитектура PCU .....	114
3.5.3	Программный конвейер .....	115
3.5.4	Программная модель PCU .....	115
3.5.4.1	Регистр-идентификатор (IDR).....	116
3.5.4.2	Регистр управления и состояния (DCSR) .....	116
3.5.4.3	Регистр программного счетчика (PC) .....	118
3.5.4.4	Регистр состояния (SR).....	118
3.5.4.5	Регистр счетчика циклов (LC).....	119
3.5.4.6	Регистр адреса цикла (LA) .....	120
3.5.4.7	Системный стек (SS).....	120
3.5.4.8	Стек цикла (CS) .....	120
3.5.4.9	Регистр указателей стека (SP).....	121
3.5.4.10	Счетчик команд (CNTR).....	123
3.6	Программная модель DSP .....	123
3.7	Состояния DSP .....	124
3.7.1	Состояние начальной установки (RESET).....	125
3.7.2	Состояние останова (STOP).....	125
3.7.3	Состояние исполнения программы (RUN).....	125

3.8	Карта памяти DSP .....	127
3.8.1	Организация обменов с памятью данных .....	128
3.8.2	Организация памяти программ PRAM .....	129
3.8.3	Адресуемые регистры .....	129
<b>4.</b>	<b>СИСТЕМНОЕ УПРАВЛЕНИЕ .....</b>	<b>132</b>
4.1	Система синхронизации .....	132
4.2	Отключение и включение тактовой частоты .....	134
4.3	Системные регистры .....	134
4.3.1	Регистр управления и состояния CSR .....	134
4.3.2	Регистр запросов прерывания QSTR .....	135
4.3.3	Регистр маски MASKR .....	137
4.4	Процедура начальной загрузки .....	137
4.5	Логика взаимодействия CPU и DSP .....	137
4.5.1	Функции CPU .....	137
4.5.2	Функции DSP .....	138
<b>5.</b>	<b>ИНТЕРВАЛЬНЫЙ ТАЙМЕР .....</b>	<b>140</b>
5.1	Назначение .....	140
5.2	Структурная схема .....	140
5.3	Регистры интервального таймера .....	141
5.4	Программирование IT .....	142
<b>6.</b>	<b>ТАЙМЕР РЕАЛЬНОГО ВРЕМЕНИ .....</b>	<b>143</b>
6.1	Назначение .....	143
6.2	Структурная схема RTT .....	143
6.3	Описание регистров таймера реального времени .....	144
6.4	Программирование RTT .....	145
<b>7.</b>	<b>СТОРОЖЕВОЙ ТАЙМЕР .....</b>	<b>146</b>
7.1	Назначение .....	146
7.2	Структурная схема .....	146
7.3	Описание регистров WDT .....	147
7.4	Программирование WDT .....	149
<b>8.</b>	<b>КОНТРОЛЛЕР ПРЯМОГО ДОСТУПА В ПАМЯТЬ (DMA) .....</b>	<b>152</b>
8.1	Общие положения .....	152
8.1.1	Типы каналов .....	152
8.1.2	Приоритет каналов DMA и CPU .....	152
8.1.3	Темп передачи .....	153
8.1.4	Регистры DMA .....	153
8.1.5	Прерывания DMA .....	154
8.2	Процедура самоинициализации .....	154
8.3	Каналы DMA последовательных портов .....	155
8.4	Каналы DMA линковых портов .....	157
8.5	Каналы обмена данными между внутренней и внешней памятью .....	158
<b>9.</b>	<b>ПОРТ ВНЕШНЕЙ ПАМЯТИ .....</b>	<b>162</b>
9.1	Введение .....	162
9.2	Регистры порта внешней памяти .....	162
9.2.1	Регистр конфигурации CSCON0 .....	163
9.2.2	Регистр конфигурации CSCON1 .....	164
9.2.3	Регистр конфигурации CSCON2 .....	164

9.2.4	Регистр конфигурации C3CON3 .....	165
9.2.5	Регистр конфигурации C3CON4 .....	166
9.2.6	Регистр управления работой с памятью SDRAM .....	167
9.2.7	Регистр СKE_CTR .....	168
9.3	Временные диаграммы обмена данными .....	169
9.3.1	Общие положения .....	169
9.3.2	Обмен данными с асинхронной памятью .....	170
9.3.3	Обмен данными с синхронной памятью .....	175
9.3.4	Обмен данными в режиме Flyby .....	179
9.4	Рекомендации по подключению внешней памяти .....	183
9.4.1	Память типа SDRAM .....	183
9.4.2	Память типа Flash .....	183
<b>10.</b>	<b>УНИВЕРСАЛЬНЫЙ АСИНХРОННЫЙ ПОРТ (UART) .....</b>	<b>184</b>
10.1	Общие положения .....	184
10.2	Регистры UART .....	185
10.2.1	Общие положения .....	185
10.2.2	Регистр LCR .....	186
10.2.3	Регистр FCR .....	187
10.2.4	Регистр LSR .....	187
10.2.5	Регистр IER .....	188
10.2.6	Регистр IIR .....	189
10.2.7	Регистр MCR .....	191
10.2.8	Регистр MSR .....	191
10.2.8	Программируемый генератор скорости обмена .....	192
10.3	Работа с FIFO по прерыванию .....	193
10.4	Работа с FIFO по опросу .....	194
<b>11.</b>	<b>ПОРТ ОБМЕНА ПОСЛЕДОВАТЕЛЬНЫМ КОДОМ .....</b>	<b>195</b>
11.1	Общие положения .....	195
11.2	Регистры .....	197
11.2.1	Общие положения .....	197
11.2.2	Буфер передачи STx .....	197
11.2.3	Буфер приема SRx .....	198
11.2.4	Регистр управления передачей данных STCTL .....	198
11.2.5	Регистр управления приемом данных SRCTL .....	200
11.2.6	Регистр коэффициентов деления при передаче данных TDIV .....	203
11.2.7	Регистр коэффициентов деления при приеме данных RDIV .....	204
11.2.8	Регистры выбора канала в многоканальном режиме .....	204
11.2.9	Регистры сравнения принимаемых данных в многоканальном режиме .....	205
11.3	Одноканальный режим работы .....	205
11.4	Режим петли .....	208
11.5	Многоканальный режим работы .....	208
11.6	DMA последовательного порта .....	211
11.7	Прерывания от последовательного порта .....	212
<b>12.</b>	<b>ЛИНКОВЫЙ ПОРТ .....</b>	<b>213</b>
12.1	Архитектура линкового порта .....	213
12.2	Регистры .....	214
12.2.1	Общие положения .....	214
12.2.2	Буфер передачи LTx .....	214
12.2.3	Буфер приема LRx .....	215

12.2.4	Регистр управления и состояния LCSR .....	215
12.2.5	Регистры порта ввода-вывода .....	216
12.3	DMA линковых портов .....	216
12.4	Прерывания от линковых портов .....	216
12.4.1	Прерывания при приеме и передаче данных .....	216
12.4.2	Прерывания по запросу обслуживания .....	217
1.2	Временная диаграмма работы линкового порта .....	218
<b>13.</b>	<b>ПОРТ JTAG И ВСТРОЕННЫЕ СРЕДСТВА ОТЛАДКИ ПРОГРАММ .....</b>	<b>219</b>
<b>14.</b>	<b>ЭЛЕКТРИЧЕСКИЕ И ВРЕМЕННЫЕ ПАРАМЕТРЫ.....</b>	<b>220</b>
14.1	Электропитание .....	220
14.2	Электрические параметры .....	220
14.3	Динамическая потребляемая мощность .....	221
14.4	Предельно-допустимые и предельные электрические режимы эксплуатации .....	222
1.3	Временные параметры .....	223
14.4.1	Обмен данными с внешней памятью и устройствами .....	223
14.4.2	Прием и передача данных по линковому порту .....	223
14.4.3	Прием и передача данных по последовательному порту .....	225
14.5	Зависимости основных параметров от режимов и условий эксплуатации .....	226
14.6	Рекомендации по подключению кварцевого резонатора .....	228
<b>15.</b>	<b>ОПИСАНИЕ ВНЕШНИХ ВЫВОДОВ .....</b>	<b>229</b>
<b>16.</b>	<b>ИСТОРИЯ ИЗМЕНЕНИЙ .....</b>	<b>237</b>
16.1	30 июня 2006 г. ....	237
16.2	23 октября 2007 г. ....	237
16.3	20 февраля 2008 г. ....	237
16.4	02 июня 2009 г. ....	237
16.5	13 декабря 2011 г. ....	237
16.6	27 августа 2012 г. ....	237
16.7	25 февраля 2013 г. ....	238
16.8	15 апреля 2013 г. ....	238
16.9	22 ноября 2018 г. ....	238



# 1. ВВЕДЕНИЕ

## 1.1 Порядок использования данного документа

В данном документе рассмотрены вопросы архитектуры и функционирования микросхемы 1892ВМ3Т. Приведены ее электрические параметры, а также чертеж корпуса и назначение выводов. Рассмотрены вопросы типового включения микросхемы в систему и даны рекомендации по ее программированию.

Настоящая документация охраняется действующим законодательством Российской Федерации об авторском праве и смежных правах, в частности, законом Российской Федерации «Об авторском праве и смежных правах». АО НПЦ «ЭЛВИС» является единственным правообладателем исключительных авторских прав на настоящую документацию.

## 1.2 Назначение

Микросхема интегральная 1892ВМ3Т спроектирована как однокристалльная двухпроцессорная “система на кристалле” на базе IP-ядерной (IP-intellectual property) платформы «МУЛЬТИКОР», разработанной в ОАО НПЦ «ЭЛВИС».

По общепринятой классификации СБИС, разрабатываемых на базе платформы «МУЛЬТИКОР», 1892ВМ3Т относится к сигнальным микропроцессорам мини-конфигурации с плавающей и фиксированной точкой.

В качестве двух процессоров 1892ВМ3Т содержит 32-разрядный центральный процессор (CPU – Central Processing Unit) и высокопроизводительный процессор-акселератор для цифровой обработки сигналов (DSP – Digital Signal Processing) с плавающей/фиксированной точкой, обеспечивающий обработку информации с переменными форматами данных от битовых форматов до стандартных форматов данных с плавающей точкой в формате IEEE754.

Сигнальный микропроцессор 1892ВМ3Т реализован на основе ядер из библиотеки платформы «МУЛЬТИКОР»: процессорного RISC - ядра **RISCore32** с архитектурой MIPS32 (CPU) и программируемого ядра с SISD (Single Instructions Single Data) архитектурой цифрового сигнального процессора (DSP) с плавающей/фиксированной точкой **ELcore-14™** (ELcore = Elvees’s core).

Микросхема 1892ВМ3Т сочетает в себе лучшие качества двух классов приборов: микроконтроллеров и цифровых процессоров обработки сигналов, что особенно важно для микроминиатюрных встраиваемых применений, когда приходится решать в рамках ограниченных габаритов одновременно обе задачи: управления и высокоточной обработки информации, включая сигналы и изображение.

Для разработчика системы обеспечивается уникальная возможность применения новых алгоритмов принятия решений в CPU на основе параллельно выполняемых процедур

адаптивного анализа и обработки сигналов в DSP, что реализуется в пределах одной и той же микросхемы, и что особенно важно для сверхминиатюрных применений. Для этих целей разработаны методы применения RLS/LNS-алгоритмов на базе микросхем серий «МУЛЬТИКОР», в частности для адаптивных антенных решеток.

Микропроцессор 1892ВМ3Т обеспечивает работу под операционной системой Linux, а также под другими операционными системами для встраиваемых применений.

Микропроцессор 1892ВМ3Т предназначен для применения в следующих приложениях:

- Управление объектами с использованием высокоточных адаптивных методов;
- Высокоточная обработка данных для малогабаритных мобильных и встраиваемых систем;
- Системы промышленного контроля;
- Графические ускорители;
- Мультимедийная обработка звука (MPEG-1 Audio Layer3 [MP3], AMR, WMA, AAC и другие звуковые кодеки);
- Фильтрация, корреляция, быстрая свертка;

### 1.3 Функциональные параметры и возможности

Сигнальный микропроцессор 1892ВМ3Т имеет следующие функциональные параметры и возможности:

#### Центральный процессор (CPU):

- Архитектура – MIPS32;
- 32-х битные шины передачи адреса и данных;
- Кэш команд объемом 16 Кбайт;
- Архитектура привилегированных ресурсов в стиле ядра R4000:
  - Регистры Count/Compare для прерываний реального времени;
  - Отдельный вектор обработки исключений по прерываниям;
- Программируемое устройство управления памятью:
  - Два режима работы – с TLB (Translation Lookaside Buffer) и FM (Fixed Mapped);
  - 16 строк в режиме TLB.
- Устройство умножения и деления;
- JTAG IEEE 1149.1, встроенные средства отладки программ
- Производительность – 80 млн. оп/сек (здесь и далее параметры производительности приведены при тактовой частоте 80 МГц);
- Оперативная память центрального процессора (CRAM) объемом 64 Кбайт;
- 5 внешних запросов прерывания, в том числе немаскируемое прерывание (NMI).

#### Цифровой сигнальный процессор (DSP):

- “Гарвардская” RISC–подобная архитектура с оригинальной системой команд и преимущественно одноктактным исполнением инструкций;
- **SISD (Single Instructions Single Data)** организация потоков команд и данных;
- Набор инструкций, совмещающий процедуры обработки и пересылки;
- 3-ступенчатый конвейер по выполнению 32- и 64-разрядных инструкций;
- Расширенные возможности по динамическому диапазону обрабатываемых данных, позволяющие обрабатывать данные в 8/16/32–разрядных форматах с фиксированной точкой, либо в одном из форматов с плавающей точкой – 24E8 (стандарт IEEE 754) или 32E16 (расширенный формат). Обеспечение при этом компромиссного выбора между точностью и производительностью. Аппаратные меры повышения точности и динамического диапазона (блочная плавающая точка; режим насыщения; инструкции преобразования форматов);
- Аппаратная поддержка программных циклов;
- Память программ PRAM объемом 16 Кбайт;
- Двухпортовые памяти данных XRAM и YRAM объемом 96 и 48 Кбайт соответственно;
- Пиковая производительность DSP:
  - 240 млн. оп/с 32-битных операций с плавающей точкой (IEEE 754);
  - 1440 млн. оп/с 8-битных операций с фиксированной точкой;
  - 640 млн. оп/с 16-битных операций с фиксированной точкой;
  - 320 млн. оп/с 32-битных операций с фиксированной точкой.

### **Порт внешней памяти (MPORT):**

- Шина данных – 32 разряда, шина адреса – 32 разряда;
- Встроенный контроллер управления статической памятью типа SRAM, FLASH, ROM, а также синхронной памятью типа SDRAM;
- Программное конфигурирование типа блока памяти и его объема;
- Программное задание циклов ожидания;
- Формирование сигналов выборки 4 блоков внешней памяти;
- Обеспечение обслуживания 4 внешних прерываний.

### **Периферийные устройства:**

- 12-канальный контроллер прямого доступа в память (DMA). 4 внешних запроса прямого доступа. Специальные режимы синхронизации. Поддержка 2-мерной и разрядно-инверсной адресации. Режим передачи Flyby, подобный реализованному в ADSP-TS201: внешнее устройство ↔ внешняя память;
- два порта обмена последовательным кодом (SPORT), совместимые с ADSP21160 (разработка фирмы ADI);
- четыре линковых порта (LPORT), совместимые с ADSP21160. Имеется режим работы в качестве портов ввода-вывода общего назначения (GPIO);
- универсальный асинхронный порт (UART) типа 16550;
- 32-разрядный интервальный таймер (IT);

- 32-разрядный таймер реального времени (RTT);
- 32-разрядный сторожевой таймер (WDT).

**Дополнительные возможности и особенности:**

- Узел фазовой автоподстройки частоты (PLL) с множителем/делителем входной частоты;
- Встроенные средства отладки программ (OnCD);
- Порт JTAG в соответствии со стандартом IEEE 1149.1;
- Режимы энергосбережения;
- Поддержка операционной системы Linux;

В Таблице 1.1 приведены основные параметры быстродействия микросхемы 1892BM3T.

**Таблица 1.1 Основные параметры быстродействия микросхемы 1892ВМ3Т**

Характеристика	Значение параметра
<b>Пиковая производительность (в количестве арифметических операций за 1 такт) для:</b> <ul style="list-style-type: none"> <li>• 1b целочисленного формата данных</li> <li>• 8b целочисленного формата данных</li> <li>• 16b целочисленного формата данных</li> <li>• 32b целочисленного формата данных</li> <li>• 32b формата данных с плавающей точкой (IEEE754)</li> </ul>	<p>64</p> <p>18</p> <p>8</p> <p>5</p> <p>3</p>
<b>Количество МАС - операций (умножение с накоплением) за 1 такт для:</b> <ul style="list-style-type: none"> <li>• МАС 1*1+32, целочисленный 1b формат данных</li> <li>• МАС <math>(8+j8)*(8+j8)+(32+j32)</math>, комплексный целочисленный 8b формат данных</li> <li>• МАС 16*16+32, целочисленный 16b формат данных</li> <li>• МАС 32*32+64, целочисленный 32b формат данных</li> <li>• МАС 32*32+32, формат 32b данных с плавающей точкой (IEEE754)</li> </ul>	<p>32</p> <p>2</p> <p>2</p> <p>1</p> <p>1</p>
<b>Время выполнения операции сложения с плавающей точкой расширенного формата 32E16, в тактах:</b> <ul style="list-style-type: none"> <li>• с нормализацией результата</li> <li>• без нормализации результата</li> </ul>	<p>5</p> <p>3</p>
<b>Время выполнения операции вычитания с плавающей точкой расширенного формата 32E16, в тактах:</b> <ul style="list-style-type: none"> <li>• с округлением</li> <li>• без округления</li> <li>• без нормализации результата</li> <li>• без округления и нормализации</li> </ul>	<p>6</p> <p>5</p> <p>4</p> <p>3</p>
<b>Время выполнения операции сложения и вычитания одной пары операндов с плавающей точкой расширенного формата 32E16, в тактах:</b> <ul style="list-style-type: none"> <li>• с округлением</li> <li>• без округления</li> <li>• без нормализации результата</li> <li>• без округления и нормализации</li> </ul>	<p>9</p> <p>8</p> <p>5</p> <p>4</p>
<b>Время выполнения операции умножения с плавающей точкой расширенного формата 32E16, в тактах:</b> <ul style="list-style-type: none"> <li>• с нормализацией результата</li> <li>• без нормализации результата</li> </ul>	<p>4</p> <p>2</p>
<b>Нерекурсивная фильтрация, целочисленный формат 16*16+32:</b> <ul style="list-style-type: none"> <li>• Производительность, число тактов на отвод</li> <li>• Скалярная задержка</li> </ul>	<p>0.5</p> <p>1</p>
<b>Нерекурсивная фильтрация, целочисленный формат 32*32+64:</b> <ul style="list-style-type: none"> <li>• Производительность, число тактов на отвод</li> <li>• Скалярная задержка</li> </ul>	<p>1</p> <p>1</p>
<b>Нерекурсивная фильтрация, целочисленный комплексный формат <math>(8+j8)*(8+j8)+(32+j32)</math>:</b>	

Характеристика	Значение параметра
<ul style="list-style-type: none"> <li>Производительность, число тактов на отвод</li> <li>Скалярная задержка</li> </ul>	0.5 1
<b>Нерекурсивная фильтрация, целочисленный комплексный формат <math>(16+j16)*(16+j16)+(32+j32)</math>:</b> <ul style="list-style-type: none"> <li>Производительность, число тактов на отвод</li> <li>Скалярная задержка</li> </ul>	2 2
<b>Нерекурсивная фильтрация, целочисленный комплексный формат <math>(32+j32)*(32+j32)+(64+j64)</math>:</b> <ul style="list-style-type: none"> <li>Производительность, число тактов на отвод</li> <li>Скалярная задержка</li> </ul>	4 4
<b>Нерекурсивная фильтрация, комплексный формат плавающей точки <math>(32+j32)*(32+j32)+(32+j32)</math>:</b> <ul style="list-style-type: none"> <li>Производительность, число тактов на отвод</li> <li>Скалярная задержка</li> </ul>	4 4
<b>БПФ- 1024, комплексный формат данных и коэффициентов <math>(16+j16)</math>, блочная плавающая точка</b>	11600
<b>БПФ - 1024, комплексный формат плавающей точки, стандарт IEEE 754</b>	21000
<b>БПФ- 256, комплексный формат данных и коэффициентов <math>(16+j16)</math>, блочная плавающая точка</b>	2400
<b>БПФ - 256, комплексный формат плавающей точки, стандарт IEEE 754</b>	4300
<b>Декодер Витерби, на одну метрику пути, 16-битный формат</b>	1
<b>БП Уолша – Адамара - 256, комплексное, формат <math>(16+j16)</math>, блочная плавающая точка</b>	1200
<b>Деление <math>(y/x)</math>, формат плавающей точки, стандарт IEEE 754</b>	10
<b>Обратная величина квадратному корню, формат плавающей точки, стандарт IEEE 754</b>	12

## 1.4 Структурная схема

Структурная схема микросхемы 1892ВМ3Т приведена на Рисунок 1.1.

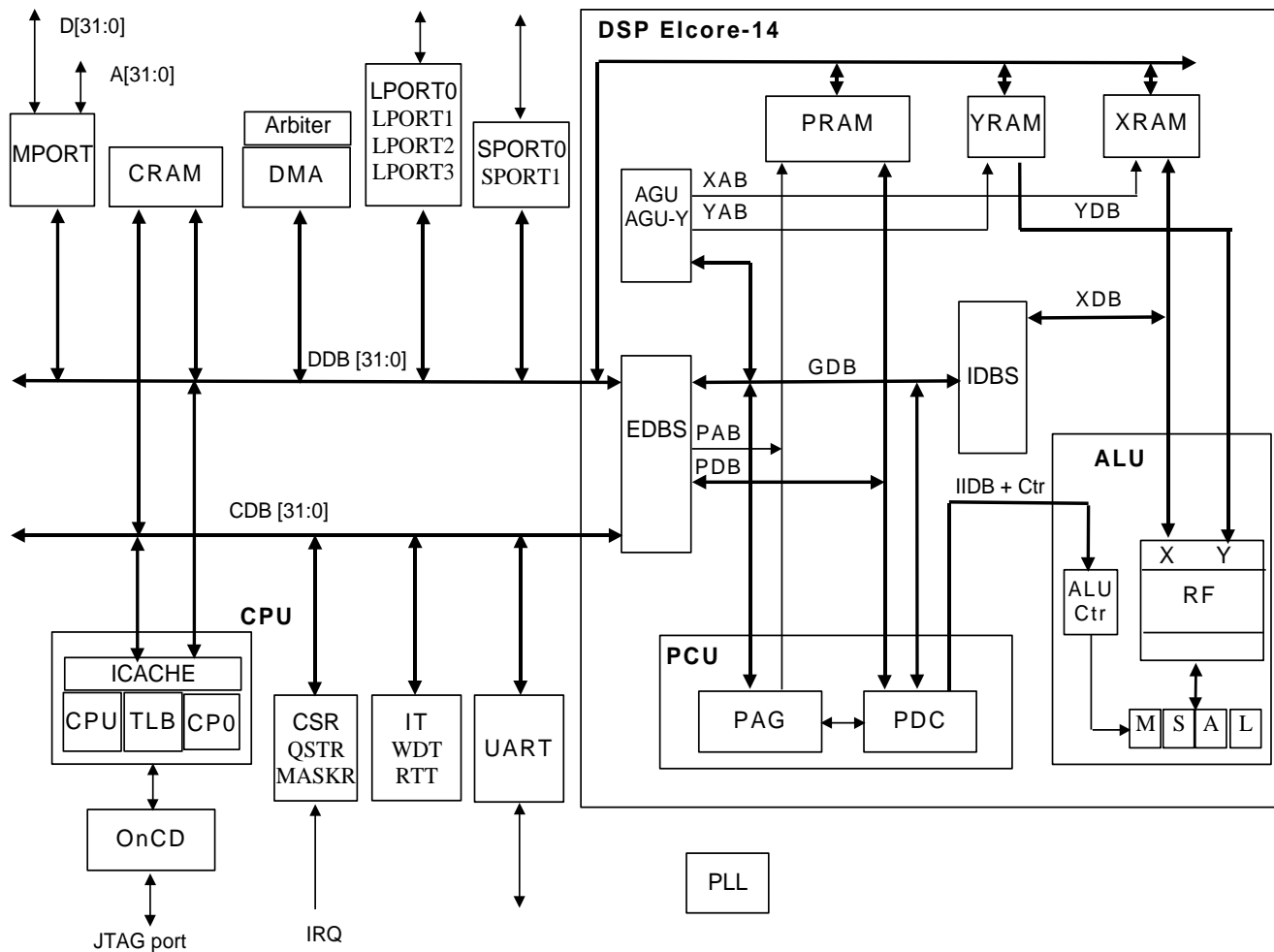


Рисунок 1.1. Структурная схема микросхемы 1892ВМ3Т

В состав микросхемы 1892BM3T входят следующие основные узлы и компоненты:

- **CPU** – центральный процессор на основе RISC-ядра;
- **CRAM** – двухпортовая оперативная память центрального процессора;
- **DSP** – сопроцессор цифровой обработки сигналов с фиксированной точкой (далее может называться также **ЦПОС** – цифровой процессор обработки сигналов);
- **DMA** – контроллер прямого доступа в память;
- **MPORT** – порт внешней памяти;
- **SPORT** – последовательный порт;
- **LPORT** – линковый порт;
- **UART** – универсальный асинхронный порт;
- **ICACHE** – кэш программ центрального процессора;
- **IT** – интервальный таймер;
- **WDT** – сторожевой таймер;
- **RTT** – таймер реального времени;
- **CDB[31:0]** – шина данных CPU;
- **DDB[31:0]** – шина данных DMA;
- **A[31:0]** – шина адреса порта внешней памяти;
- **D[31:0]** – шина данных порта внешней памяти;
- **OnCD** – встроенные средства отладки программ;
- **XRAM, YRAM** – памяти данных DSP;
- **PRAM** – память программ DSP;
- **AGU** – адресный генератор;
- **EDBS** – коммутатор внешних шин;
- **IDBS** – коммутатор внутренних шин;
- **PCU** – устройство программного управления;
- **PAG** – генератор адреса программ;
- **PDC** – программный дешифратор;
- **RF** – регистровый файл;
- **ALU** – арифметическое устройство;
- **ALUctr** – управление ALU;
- **XDB0 – XDB3, GDB, PDB** – шина данных DSP;
- **XAB, YAB, PAB** – адресные шины DSP;
- **M, S, A, L** – арифметические узлы ALU DSP.



На Рисунке 1.2 приведена типовая схема применения микросхемы 1892ВМ3Т.

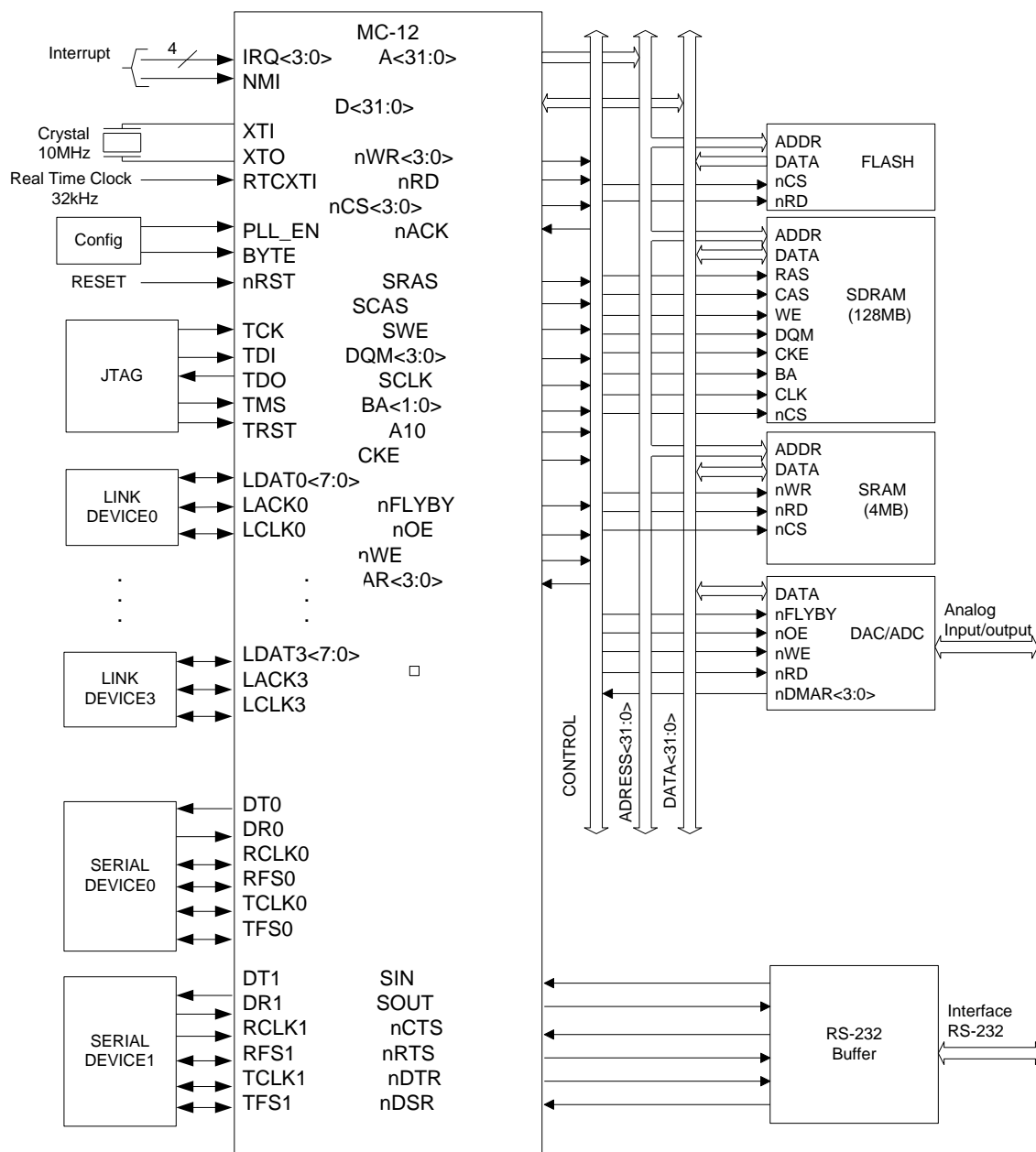


Рисунок 1.2. Типовая схема применения микросхемы 1892ВМ3Т

На Рисунке 1.2 использованы следующие обозначения:

- **FLASH** – постоянное запоминающее устройство типа FLASH;
- **SDRAM** – синхронное динамическое оперативное запоминающее устройство (ОЗУ);
- **SRAM** – статическое ОЗУ;
- **Interrupt** – запросы прерывания;
- **DAC/ADC** – цифро-аналоговые и аналого-цифровые преобразователи;
- **LINK DEVICE**– устройства, подключаемые к линковым портам;
- **SERIAL DEVICE** – устройства, подключаемые к последовательным портам;
- **RS-232 Buffer** – приемо-передатчики RS-232;
- **Config** – схема задания конфигурации;
- **RESET** – узел формирования сигнала установки исходного состояния.

## 1.5 Инструментальное программное обеспечение

Для данной микросхемы разработана интегрированная среда проектирования программного обеспечения MCStudio, которая обеспечивает полный цикл разработки и отладки программ. Эта среда является кросс - системой и функционирует на инструментальной машине IBM PC в среде Windows 9x, XP.

Интегрированная среда проектирования включает в себя:

- среду разработки программ для CPU- и DSP-ядер;
- среду отладки программ в исходных текстах, исполняемых на программном симуляторе, и отладчик для работы с платой отладочного модуля для данной микросхемы или целевым устройством. Целевое устройство подключается к персональному компьютеру через эмулятор JTAG, предназначенный для работы с данным микропроцессором;
- средства программного моделирования;
- возможность доступа пользователю ко всем инструментам через один интерфейс.

Среда разработки программ для CPU-ядра включает в себя:

- компилятор с языка Си с препроцессором;
- ассемблер с препроцессором;
- дисассемблер;
- линковщик;
- библиотекарь;
- утилиты подготовки исполняемого кода.

Среда разработки программ для DSP-ядра включает в себя:

- ассемблер с препроцессором;
- дисассемблер;
- линковщик;

- библиотекарь;
- утилиты подготовки исполняемого кода.

Инструментальное программное обеспечение 1892ВМ3Т базируется на архитектуре MIPS32. Вследствие этого, оно поддерживает большой объем свободно распространяемого программного обеспечения для этой архитектуры.

Библиотека прикладных программ для микросхемы 1892ВМ3Т включает:

- программы БПФ комплексных и действительных чисел;
- программы быстрой свертки и корреляции посредством БПФ (перекрытие с накоплением);
- рекурсивные и не рекурсивные программы фильтрации данных;
- элементарные математические функции;
- арифметические операции над матрицами;
- обработка изображений.

## 1.6 Операционная система для микросхемы 1892ВМ3Т

Linux - свободно распространяемое ядро Unix-подобной операционной системы. Linux обладает всеми свойствами современной Unix-системы, включая полноценную многозадачность, развитую подсистему управления памятью и сетевую подсистему.

Ядро Linux, поставляемое вместе со свободно распространяемыми прикладными и системными программами образует полнофункциональную универсальную операционную систему. Большую часть базовых системных компонент Linux унаследовал от проекта GNU, целью которого является создание свободной микроядерной операционной системы с лицом Unix.

На CPU-ядро микросхемы 1892ВМ3Т портировано ядро Linux, предоставляемое по отдельному запросу.

## 2. ЦЕНТРАЛЬНЫЙ ПРОЦЕССОР

### 2.1 Основные характеристики CPU

- Архитектура – MIPS32;
- 32-х битные пути передачи адреса и данных;
- Кэш команд объемом 16 Кбайт;
- Архитектура привилегированных ресурсов в стиле ядра R4000:
  - Регистры Count/Compare для прерываний реального времени;
  - Отдельный вектор обработки исключений по прерываниям;
- Программируемое устройство управления памятью:
  - Два режима работы – с TLB (Translation Look aside Buffer) и FM (Fixed Mapped);
  - 16 строк в режиме TLB;
- Устройство умножения и деления;
- Поддержка отладки JTAG.

### 2.2 Блок-схема

Блок схема процессорного ядра RISCore32 приведена на Рисунок 2.1.

Ядро содержит следующие узлы:

- Устройство исполнения (Execution Core);
- Устройство умножения и деления (MDU);
- Системный управляющий сопроцессор (CP0);
- Устройство управления памятью (MMU – Memory Management Unit);
- Контроллер кэш (Cache Controller);
- Устройство шинного интерфейса (BIU);
- Кэш команд (I\$);
- Средства отладки программ (OnCD – On Chip Debugger) с JTAG портом.

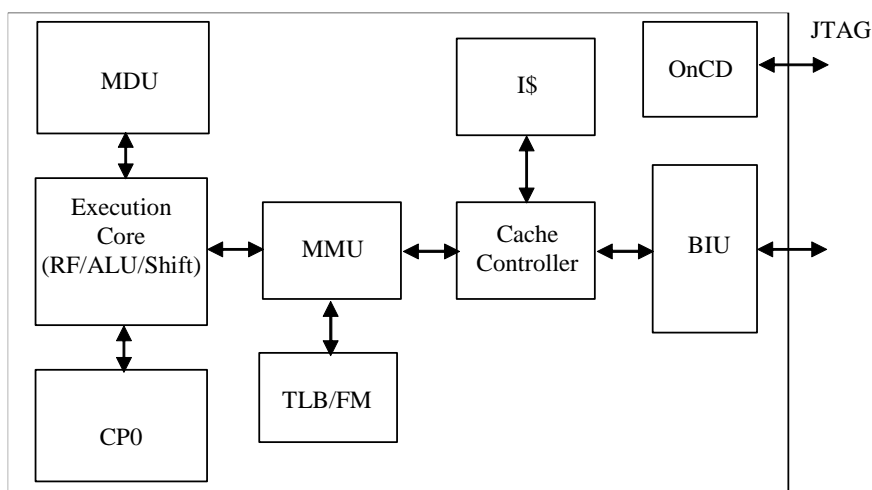


Рисунок 2.1. Блок-схема процессорного ядра RISC Core32

## 2.3 Составляющие логические блоки

В следующих подразделах описываются устройства, входящие в состав процессорного ядра.

### 2.3.1 Устройство исполнения

Входящее в ядро устройство исполнения реализует архитектуру load-store (загрузка-сохранение) с одноктактными операциями арифметического логического устройства (АЛУ) (логические операции, операции сдвига, сложение и вычитание). В ядре имеется тридцать два 32-х битных регистра общего назначения, используемых для скалярных целочисленных операций и вычисления адреса. В регистровом файле есть два порта чтения и один порт записи. Также используются обходные пути передачи данных для минимизации количества остановок конвейера.

В состав устройства исполнения входят:

- 32-х битный сумматор, используемый для вычисления адреса данных;
- Адресное устройство для вычисления адреса следующей команды;
- Логика определения перехода и вычисления адреса перехода;
- Блок выравнивания при загрузке данных;
- Мультиплексоры обходных путей передачи данных для исключения остановок конвейера в тех случаях, когда команды, производящие данные и команды, использующие эти данные, расположены в программе достаточно близко;
- Блок обнаружения Нуля/Единицы для реализации команд CLZ и CLO;
- АЛУ для выполнения побитных операций;
- Сдвигающее устройство и устройство выравнивания при сохранении данных.

### **2.3.2 Устройство умножения/деления (MDU)**

Устройство умножения/деления выполняет соответствующие операции. MDU выполняет операции умножения за 17 тактов, операции умножения с накоплением за 18 тактов, операции деления за 33 такта и операции деления с накоплением за 34 такта. Попытка активизировать следующую команду умножения/деления до завершения выполнения предыдущей, так же как и использование результата этой операции до того, как она закончена, вызывает остановку конвейера. В MDU имеется вывод, определяющий формат операции – знаковый или беззнаковый.

### **2.3.3 Системный управляющий сопроцессор**

Сопроцессор отвечает за преобразование виртуального адреса в физический, протоколы кэш, систему управления исключениями, выбор режима функционирования (Kernel/User) и за разрешение/запрещение прерываний. Конфигурационная информация доступна посредством чтения регистров CP0 (см. раздел 2.7 “Регистры CP0”).

### **2.3.4 Устройство управления памятью (MMU)**

Процессорное ядро содержит устройство управления памятью (MMU), реализующее интерфейс между исполнительным блоком и контроллером кэш. Ядро может работать как в режиме TLB – с 16-строчной, полностью ассоциативной матрицей TLB, так и в режиме FM (Fixed Mapped), когда используются простые преобразования виртуального адреса в физический адрес. Полностью устройство MMU описано в п. 2.5.

### **2.3.5 Контроллер кэш**

В данной версии процессора реализован кэш команд, виртуально индексируемый и контролируемый по физическому тэгу типа direct mapped, что позволяет осуществлять доступ к кэш параллельно с преобразованием виртуального адреса в физический адрес. Объем кэш памяти составляет 16 Кбайт.

### **2.3.6 Устройство шинного интерфейса (BIU – Bus Interface Unit)**

Устройство шинного интерфейса управляет внешними интерфейсными сигналами в соответствии со спецификацией шины АНВ (Advanced High-performance Bus) архитектуры АМВА (Advanced Microcontroller Bus Architecture).

### **2.3.7 OnCD контроллер**

В ядре имеется устройство для отладки программ OnCD с портом JTAG.

## **2.4 Конвейер**

В RISC-ядре процессора реализован конвейер, состоящий из пяти стадий и аналогичный конвейеру ядра R3000. Конвейер дает возможность процессору работать

на высокой частоте, при этом минимизируется сложность устройства, а также уменьшается стоимость и потребление энергии.

В этой главе содержатся следующие разделы:

- Раздел 2.1, “Стадии работы конвейера”
- Раздел 2.2, “Операции умножения и деления”
- Раздел 2.3, “Задержка выполнения команд перехода”
- Раздел 2.4, “Обходные пути передачи данных (Data bypass)”
- Раздел 2.5, “Задержка загрузки данных”
- Раздел 2.6, “Особые случаи при выполнении команд (Instruction Hazards)”

### 2.4.1 Стадии конвейера

Конвейер содержит пять стадий:

- Выборка команды (стадия I- Instruction)
- Дешифрация команды (стадия D - Data)
- Исполнение команды (стадия E - Execution)
- Выборка из памяти (стадия M - Memory)
- Обратная запись (стадия W – Write Back)

На Рисунок 2.2 показаны операции, выполняемые CPU-ядром на каждом этапе конвейера.

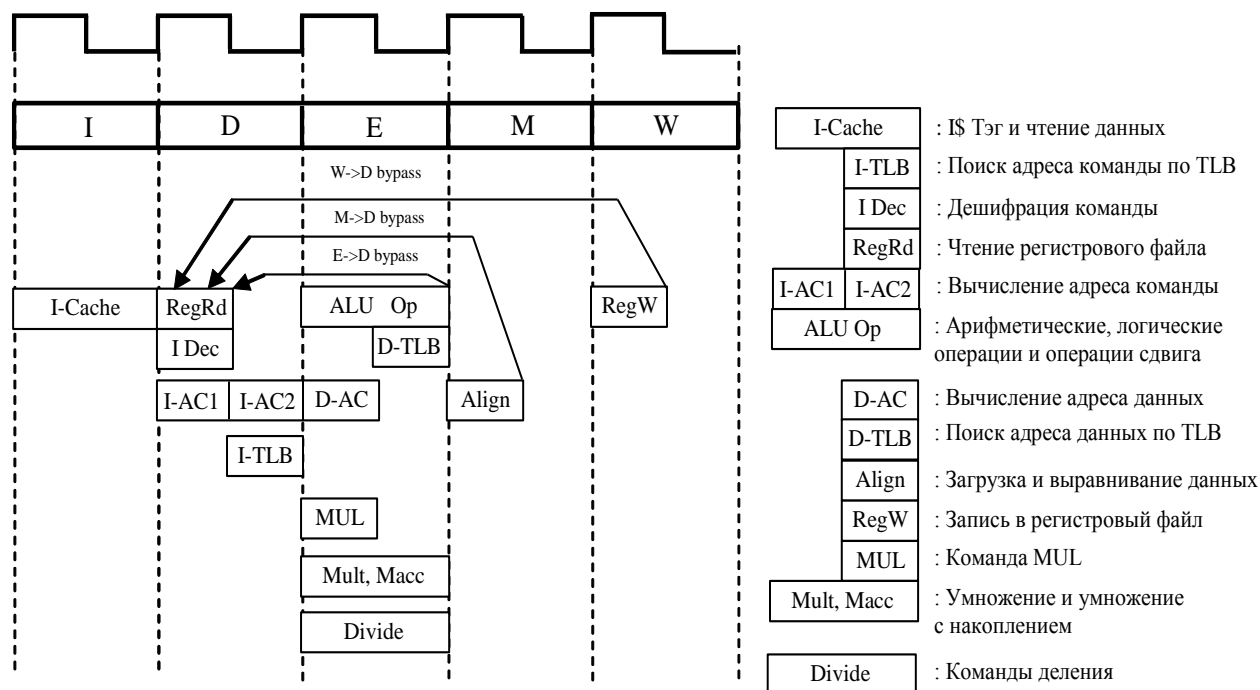


Рисунок 2.2

#### 2.4.1.1 Стадия I: выборка команды

На этой стадии команда выбирается из командного кэш.

### 2.4.1.2 Стадия D: дешифрация команды

На этой стадии:

- Операнды выбираются из регистрового файла.
- Операнды передаются на эту стадию со стадий E, M и W.
- ALU определяет, выполняется ли условие перехода и вычисляет виртуальный адрес перехода для команд перехода.
- Осуществляется преобразование виртуального адреса в физический адрес.
- Производится поиск адреса команды по TLB и вырабатывается признак hit/miss.
- Командная логика выбирает адрес команды.

### 2.4.1.3 Стадия E: исполнение

На этой стадии:

- ALU выполняет арифметические или логические операции для команд типа регистр-регистр.
- Производится преобразование виртуального адреса в физический адрес для данных, используемых командами загрузки и сохранения.
- Производится поиск данных по TLB и вырабатывается признак hit/miss.
- Все операции умножения и деления выполняются на этой стадии.

### 2.4.1.4 Стадия M: выборка из памяти

На этой стадии осуществляется загрузка и выравнивание загруженных данных в границах слова.

### 2.4.1.5 Стадия W: обратная запись

На этой стадии для команд типа регистр-регистр или для команд загрузки результат записывается обратно в регистровый файл.

## 2.4.2 Операции умножения и деления

Время выполнения этих операций соответствует 17 тактам для команд умножения и 18 тактам для команд умножения с накоплением, а также 33 тактам для команд деления и 34 тактам для команд деления с накоплением.

## 2.4.3 Задержка выполнения команд перехода (Jump, Branch)

Конвейер осуществляет выполнение команд перехода с задержкой в один такт. Однотактная задержка является результатом функционирования логики, ответственной за принятие решения о переходе на стадии D конвейера. Эта задержка позволяет использовать адрес перехода, вычисленный на предыдущей стадии, для доступа к команде на следующей D-стадии. Слот задержки перехода (branch delay slot) позволяет отказаться от остановок конвейера при переходе. Вычисление адреса и проверка условия перехода выполняются одновременно на стадии D. Итоговое значение PC



(счетчика команд) используется для выборки очередной команды на стадии I, которая является второй командой после перехода. На Рисунок 2.3 показан слот задержки перехода.

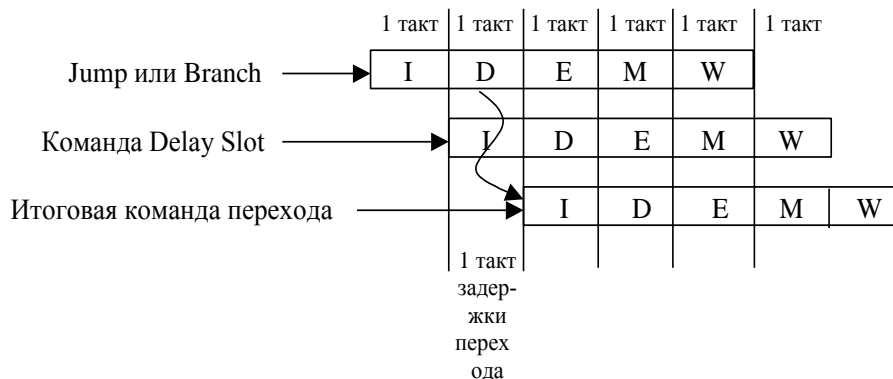


Рисунок 2.3. Слот задержки перехода

#### 2.4.4 Обходные пути передачи данных (Data bypass)

Для большинства команд MIPS32 исходными операндами являются значения, хранящиеся в регистрах общего назначения. Эти операнды выбираются из регистрового файла в первой половине D-стадии. После исполнения на ALU результат, в принципе, готов для использования другими командами. Но запись результата в регистровый файл осуществляется только на стадии W. Это лишает следующую команду возможности использовать результат в течение 3-х циклов, если ее операндом является результат выполнения последней операции, сохраненный в регистровом файле. Для преодоления этой проблемы используются обходные пути передачи данных.

Мультиплексоры обходных путей передачи данных для обоих операндов располагаются между регистровым файлом и ALU (Рисунок 2.4). Они позволяют передавать данные с выхода стадий E, M и W конвейера прямо на стадию D, если один из регистров источника (source) декодируемой команды совпадает с регистром назначения (target) одной из предшествующих команд. Входы мультиплексоров подключены к обходным путям M→D и E→D, а также W→D.

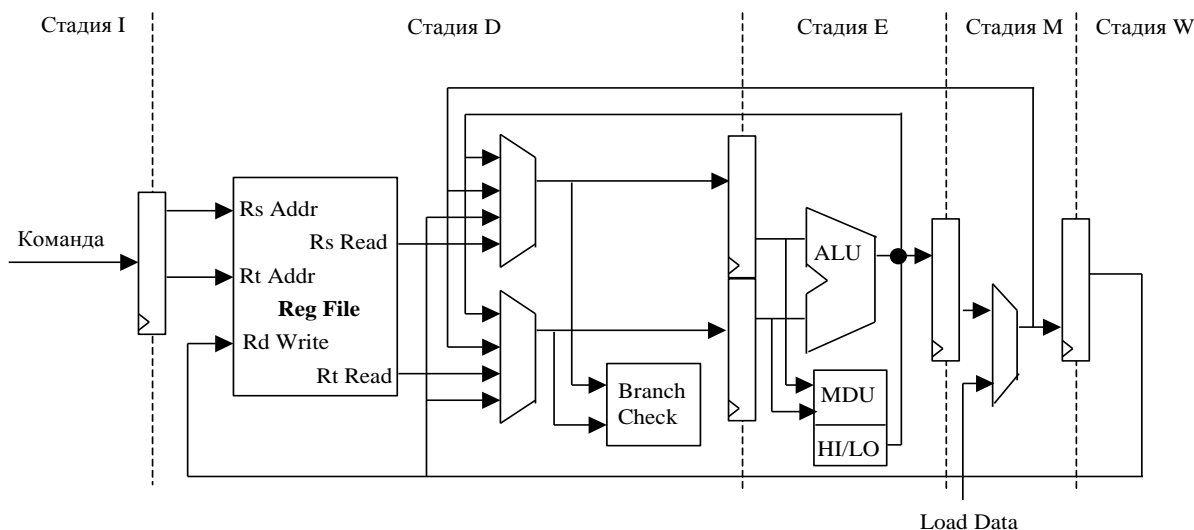


Рисунок 2.4

На Рисунок 2.5 показаны обходные пути передачи данных для команды Add<sub>1</sub>, за которой следует команда Sub<sub>2</sub> и затем снова Add<sub>3</sub>. Поскольку команда Sub<sub>2</sub> в качестве одного из операндов использует результат операции Add<sub>1</sub>, используется обходной путь E→D. Следующая команда Add<sub>3</sub> использует результаты обеих предшествующих операций: Add<sub>1</sub> и Sub<sub>2</sub>. Так как данные команды Add<sub>1</sub> в это время находятся на стадии M, используется обходной путь M→D. Кроме того, вновь используется обходной путь E→D для передачи результата операции Sub<sub>2</sub> команде Add<sub>3</sub>.

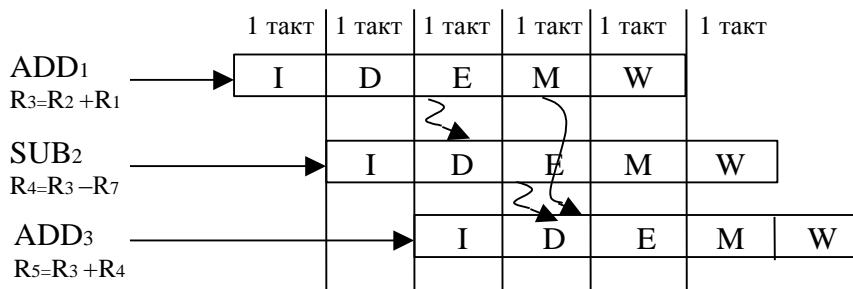


Рисунок 2.5

## 2.4.5 Задержка загрузки данных

Данные, выбираемые командами загрузки (Load), становятся доступными на конвейере только после выравнивания на стадии M. При этом данные, являющиеся исходными операндами, должны предоставляться командам для обработки уже на стадии D. Поэтому, если сразу за командой загрузки следует команда, для которой один из регистров исходных операндов совпадает с регистром, в который производится загрузка данных, это вызывает приостановку в работе конвейера на стадии D. Эта приостановка осуществляется аппаратной вставкой команды NOP. Во время этой задержки часть конвейера, которая находится дальше стадии D, продолжает продвигаться. Если же команда, использующая загружаемые данные, следует за командой загрузки не сразу, а через одну или через две, то для обеспечения бесперебойной работы конвейера используется один из обходных путей передачи данных: M→D или W→D (Рисунок 2.6).

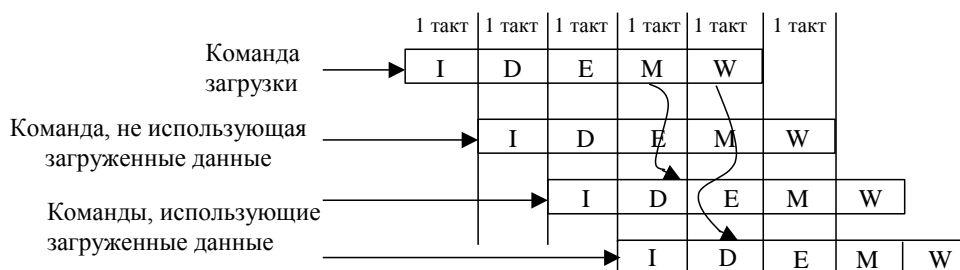


Рисунок 2.6

## 2.5 Устройство управления памятью (MMU)

### 2.5.1 Введение

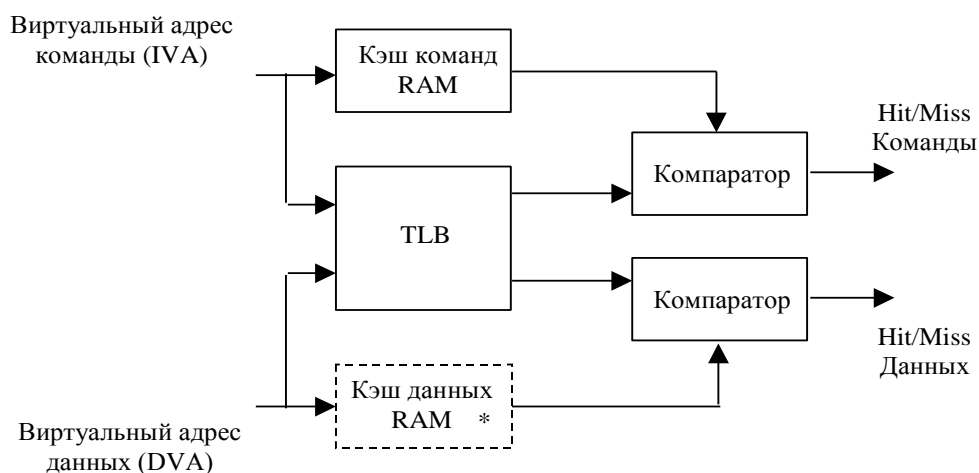
Процессорное ядро содержит устройство управления памятью (MMU), реализующее интерфейс между устройством исполнения и контроллером кэш. MMU преобразует виртуальный адрес в физический прежде, чем посылает запрос контроллеру кэш для сравнения тэга или блоку шинного интерфейса для доступа к внешнему запоминающему устройству. Это преобразование является очень полезным свойством функционирования операционных систем при управлении физической памятью таким образом, чтобы в ней размещались несколько процессов, активных в одной и той же области памяти, и может быть даже на одном виртуальном адресе, но обязательно в различных областях физической памяти. Другие свойства MMU - защита зон памяти и определение протокола кэш.

MMU может выполнять преобразование адресов в двух режимах: в режиме TLB и в режиме FM. Режим преобразования определяется битом FM регистра CSR.

В режиме TLB используется полностью ассоциативная таблица преобразования адресов (TLB), имеющая 16 парных строк (entries). Во время преобразования осуществляется поиск соответствия по TLB. Если искомая строка отсутствует, генерируется прерывание.

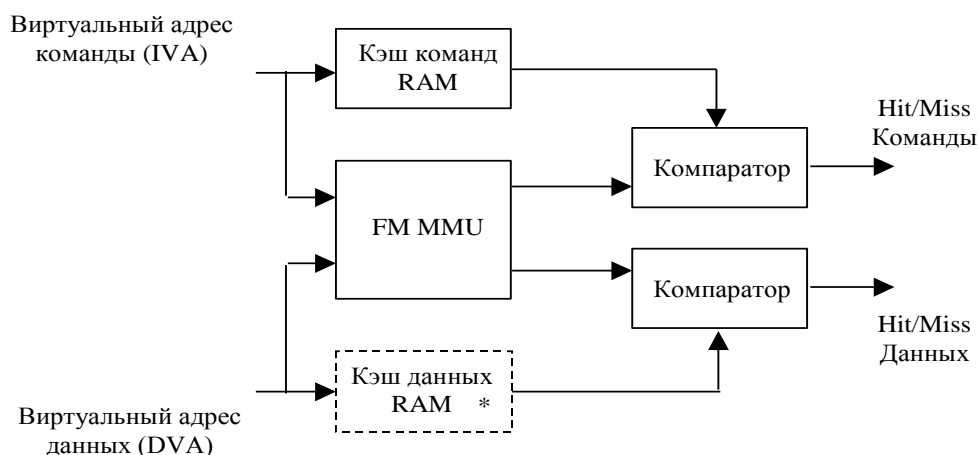
В режиме FM (Fixed Mapped) работа MMU основана на простом алгоритме, обеспечивающем преобразование виртуального адреса в физический посредством механизма фиксированного отображения. Правила преобразования отличаются для различных областей виртуального адресного пространства (useg/kuseg, kseg0, kseg1, kseg2, kseg3).

На Рисунок 2.7 показано, взаимодействие MMU с процедурой доступа к кэш в режиме TLB, а на Рисунок 2.8 – в режиме FM.



\* - Кэш данных в данной реализации отсутствует

**Рисунок 2.7**



\* - Кэш данных в данной реализации отсутствует

**Рисунок 2.8**

## 2.5.2 Режимы работы

Процессорное ядро поддерживает два режима работы:

- Режим User (непривилегированный режим)
- Режим Kernel (привилегированный режим)

Режим User в основном используется для прикладных программ. Режим Kernel обычно используется для обработки исключительных ситуаций и привилегированных функций операционной системы, включая управление сопроцессором CP0 и доступ к устройствам ввода-вывода.

Преобразования, выполняемые MMU, зависят от режима работы процессора.

### 2.5.2.1 Виртуальные сегменты памяти

Виртуальные сегменты памяти, на которые делится адресное пространство, различаются в зависимости от режима работы процессора. На Рисунок 2.9 показана сегментация для 4 Гбайт ( $2^{32}$  байт) виртуального адресного пространства, адресуемого 32-разрядным виртуальным адресом для обоих режимов работы.

Ядро входит в режим Kernel после аппаратного сброса или когда происходит исключение. В режиме Kernel программное обеспечение имеет доступ к полному адресному пространству и ко всем регистрам CP0. В режиме User доступ ограничен подмножеством виртуального адресного пространства (0x0000\_0000 - 0x7FFF\_FFFF) и запрещен доступ к функциям CP0. В режиме User недоступны виртуальные адреса 0x8000\_0000 - 0xFFFF\_FFFF и обращение к ним вызывает исключение.

0xFFFF_FFFF			kseg3
0xE000_0000			
0xDFFF_FFFF			kseg2
0xC000_0000			
0xBFFF_FFFF			kseg1
0xA000_0000			
0x9FFF_FFFF			kseg0
0x8000_0000			
0x7FFF_FFFF			
	useg		kuseg
0x0000_0000			

**Рисунок 2.9. Карта виртуальной памяти для режимов User и Kernel**

Каждый из сегментов, показанных на Рисунок 2.9, является либо отображаемым (mapped), либо неотображаемым (unmapped). Различие объясняется в следующих двух разделах.

#### 2.5.2.1.1 Неотображаемые сегменты

В неотображаемом сегменте механизмы TLB или FM для преобразования виртуального адреса в физический адрес не используются. Особенно важно иметь неотображаемые сегменты памяти после аппаратного сброса, потому что TLB еще не запрограммировано и не может осуществлять преобразования.

Для неотображаемых сегментов преобразование виртуального адреса в физический является фиксированным.

Все неотображаемые сегменты, за исключением kseg0, никогда не кэшируемы. Кэшируемость kseg0 определяется полем K0 регистра Config CP0.

### 2.5.2.1.2 Отображаемые сегменты

В отображаемом сегменте для преобразования виртуального адреса в физический адрес используются TLB или FM.

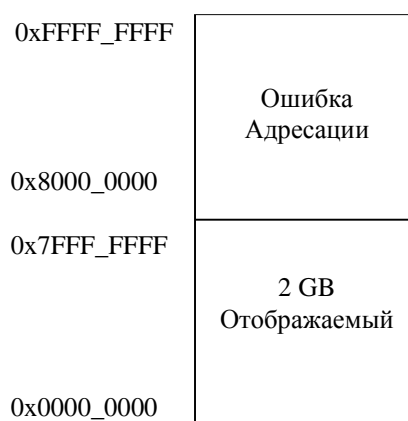
В режиме TLB преобразование отображаемых сегментов имеет постраничную основу. При преобразовании выявляется информация о кэшируемости страницы, а также атрибуты защиты, относящиеся к странице.

Для режима FM отображаемые сегменты имеют закрепленное преобразование виртуального адреса в физический. Кэшируемость сегмента определяется значениями полей K23 и KU регистра Config CP0. При FM-преобразовании невозможна защита сегментов от записи.

### 2.5.2.2 Режим User

В режиме User доступно однородное виртуальное адресное пространство размером 2 Гбайт ( $2^{31}$  байт), называемое сегментом пользователя.

На Рисунок 2.10 показано размещение виртуального адресного пространства режима User.



**Рисунок 2.10**

Сегмент потребителя начинается с адреса 0x0000\_0000 и заканчивается адресом 0x7FFF\_FFFF. Обращения по всем остальным адресам вызывают прерывания по ошибке адресации.

Процессор находится в режиме User, если в регистре Status CP0 установлены следующие значения разрядов:

- UM = 1
- EXL = 0
- ERL = 0

В Таблица 2.1 приводятся характеристики сегмента useg режима User.

Таблица 2.1

Адрес	Регистр Состояния			Имя сегмента	Диапазон адресов	Размер сегмента
	EXL	ERL	UM			
A(31)=0	0	0	1	useg	0x0000_0000 → 0x7FFF_FFFF	2GB (2 <sup>31</sup> байт)

Для всех допустимых виртуальных адресов режима User старший значащий бит адреса равен нулю, поскольку в режиме User допустимо обращение только к нижней половине карты виртуальной памяти. Любая попытка обращения по адресу со старшим битом, равным 1, в режиме User вызывает прерывание по ошибке адресации.

В режиме TLB виртуальный адрес перед преобразованием расширяется содержимым 8-разрядного поля ASID, образуя уникальный виртуальный адрес. Кэшируемость ссылки для страницы в этом режиме определяется установкой определенных бит строки TLB.

В режиме FM, область виртуальных адресов 0x0000\_0000-0x7FFF\_FFFF преобразуется в область физических адресов 0x4000\_0000-0xBFFF\_FFFF. Кэшируемость задается полем KU регистра Config CP0.

### 2.5.2.3 Режим Kernel

Процессор находится в режиме Kernel, когда регистр Status CP0 содержит хотя бы одно из следующих значений:

- UM = 0
- ERL = 1
- EXL = 1

Когда обнаруживается исключение, биты EXL или ERL устанавливаются, и процессор входит в режим Kernel. При завершении процедуры обработки исключения обычно выполняется команда возвращения из исключения (ERET). Команда ERET осуществляет переход по PC исключения, очищает ERL и EXL (если ERL=0). В результате возможен возврат процессора в режим User.

Виртуальное адресное пространство режима Kernel разделено на области в соответствии со значением старших битов виртуального адреса, как показано на Рисунок 2.11. Кроме того, в Таблица 2.2 содержатся характеристики сегментов режима Kernel.

0xFFFF_FFFF	Kernel virtual address space Mapped , 512 MB	kseg3
0xE000_0000		
0xDFFF_FFFF	Kernel virtual address space Mapped, 512 MB	kseg2
0xC000_0000		
0xBFFF_FFFF	Kernel virtual address space Unmapped, Uncached, 512 MB	kseg1
0xA000_0000		
0x9FFF_FFFF	Kernel virtual address space Unmapped, 512 MB	kseg0
0x8000_0000		
0x7FFF_FFFF	Mapped, 2048 MB	kuseg
0x0000_0000		

Рисунок 2.11

Таблица 2.2

Адрес	Регистр Состояния			Имя сегмента	Диапазон адресов	Размер сегмента
	EX L	ER L	UM			
A(31)=0	UM = 0			kuseg	0x0000_0000 → 0x7FFF_FFFF	2 GB (2 <sup>31</sup> )
A(31:29)=100 <sub>2</sub>	или			kseg0	0x8000_0000 → 0x9FFF_FFFF	512 MB (2 <sup>29</sup> )
A(31:29)=101 <sub>2</sub>	EXL=1			kseg1	0xA000_0000 → 0xBFFF_FFFF	512 MB (2 <sup>29</sup> )
A(31:29)=110 <sub>2</sub>	или			kseg2	0xC000_0000 → 0xDFFF_FFFF	512 MB (2 <sup>29</sup> )
A(31:29)=111 <sub>2</sub>	ERL=1			kseg3	0xE000_0000 → 0xFFFF_FFFF	512 MB (2 <sup>29</sup> )



### Режим Kernel, Пространство пользователя (kuseg)

Если старший значащий бит виртуального адреса  $A[31]=0$ , то выбирается виртуальное адресное пространство kuseg объемом 2 Гбайт, отображенное на адреса  $0x0000\_0000 - 0x7FFF\_FFFF$ .

При  $ERL=0$  в режиме TLB виртуальный адрес расширяется 8-битным значением поля ASID для образования уникального виртуального адреса. Кэшируемость определяется полем C строки TLB.

При  $ERL=0$  в режиме FM, область виртуальных адресов  $0x0000\_0000-0x7FFF\_FFFF$  преобразуется в область физических адресов  $0x4000\_0000-0xBFFF\_FFFF$ . Кэшируемость задается полем KU регистра Config CP0.

При  $ERL = 1$  в режимах TLB и FM, область адресов пользователя становится неотображаемым и некэшируемым адресным пространством. Виртуальный адрес kuseg соответствует тому же физическому адресу и не включает поле ASID. То есть, область виртуальных адресов kuseg соответствует области физических адресов  $0x0000\_0000-0x7FFF\_FFFF$ .

### Режим Kernel, пространство 0 режима Kernel (kseg0).

Если в режиме Kernel три старших бита виртуального адреса равны  $100_2$ , выбирается виртуальное адресное пространство kseg0. Это область размером  $2^{29}$  байт (512 MB), которая расположена внутри границ, определяемых адресами  $0x8000\_0000$  и  $0x9FFF\_FFFF$ .

Вне зависимости от состояния бита ERL и режима работы ссылки к kseg0 не отображаются, а физический адрес получается вычитанием  $0x8000\_0000$  из виртуального адреса. Кэшируемость сегмента kseg0 определяется значением поля K0 регистра Config CP0.

### Режим Kernel, пространство 1 режима Kernel (kseg1)

Если в режиме Kernel три старших бита виртуального адреса равны  $101_2$ , выбирается виртуальное адресное пространство kseg1. Это область размером  $2^{29}$  байт (512 MB), которая расположена внутри границ, определяемых адресами  $0xA000\_0000$  и  $0xBFFF\_FFFF$ .

Вне зависимости от состояния бита ERL и режима работы ссылки к kseg1 не отображаются, а физический адрес получается вычитанием  $0xA000\_0000$  из виртуального адреса.

### Режим Kernel, пространство 2 режима Kernel (kseg2)

Если в режиме Kernel три старших бита виртуального адреса равны  $110_2$ , выбирается виртуальное адресное пространство kseg2.

В режиме TLB вне зависимости от состояния бита ERL это виртуальное пространство отображается через TLB и его кэшируемость определяется полем C строки TLB.

В режиме FM вне зависимости от состояния бита ERL это виртуальное пространство зафиксировано в физических адресах 0xC000\_0000 - 0xDFFF\_FFFF и его кэшируемость определяется полем K23 Регистра Config CP0.

Режим Kernel, пространство 3 режима Kernel (kseg3)

Если в режиме Kernel три старших бита виртуального адреса равны  $111_2$ , выбирается 32-разрядное виртуальное адресное пространство kseg3.

В режиме TLB вне зависимости от состояния бита ERL это пространство отображается через TLB и его кэшируемость определяется полем C строки TLB.

В режиме FM вне зависимости от состояния бита ERL это виртуальное пространство зафиксировано в физических адресах 0xE000\_0000 - 0xFFFF\_FFFF и его кэшируемость определяется полем K23 регистра Config.

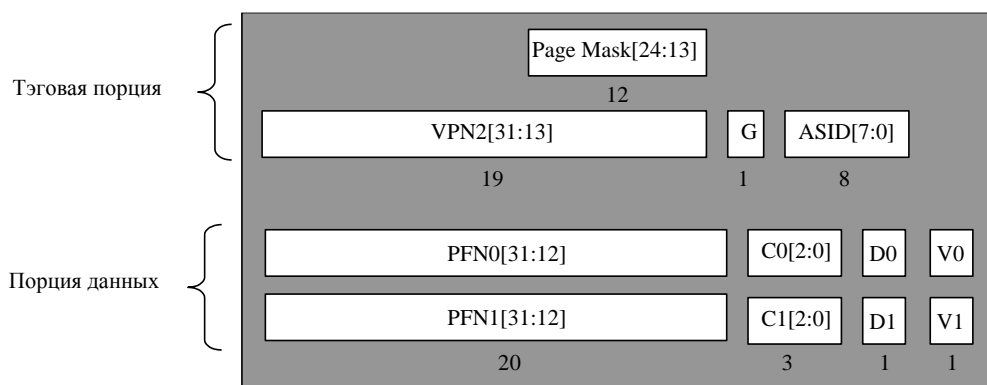
### 2.5.3 Буфер быстрого преобразования адреса (TLB)

В этой главе описывается управление памятью с помощью буфера быстрого преобразования адреса (TLB), которое осуществляется в режиме TLB.

В режиме TLB реализуется полностью ассоциативный буфер быстрого преобразования адреса (TLB), содержащий 16 двойных строк, позволяющих отображать 32 виртуальных страницы в соответствующие физические адреса. TLB организовано в виде 16 парных строк – четных и нечетных, содержащих адреса страниц размером от 4 Кбайт до 16 Мбайт, которые хранятся в 4 Гбайтном физическом адресном пространстве. Задача TLB состоит в преобразовании виртуальных адресов и их соответствующего идентификатора адресного пространства (ASID) в физический адрес памяти. Преобразование выполняется путем сравнения старших разрядов виртуального адреса (вместе с битами поля ASID) с каждой из строк тэговой порции TLB и иначе называется поиском соответствия по TLB (поиском соответствия тэга одной из строк виртуальному адресу на входе TLB).

Буфер TLB организован в виде страничных пар для минимизации общего количества хранящейся информации. Каждая строка тэговой порции соответствует двум физическим строкам данных – строке четных страниц и строке нечетных страниц. Самый старший разряд виртуального адреса, не участвующий в сравнении тэгов, определяет какая строка из двух строк данных используется. Поскольку размер страницы может варьироваться для каждой пары страниц, определение адресных разрядов, участвующих в сравнении и разряда, задающего четность страницы, должно осуществляться динамически при поиске по TLB.

На Рисунок 2.12 показано содержание одной из 16 двойных строк TLB.



**Рисунок 2.12**

Описание полей строки TLB приведены в таблице 2.3.

Таблица 2.3

Название поля	Описание																								
Page Mask[24:13]	<p>Значение маски размера страницы. Определяет размер страницы маскировкой соответствующих разрядов VPN2, и тем самым исключением их из рассмотрения. Также используется для задания адресного разряда, определяющего четность страницы (PFN0-PFN1). См. следующую таблицу:</p> <table border="1"> <thead> <tr> <th>Page Mask[11:0]</th> <th>Размер страницы</th> <th>Бит определения четности</th> </tr> </thead> <tbody> <tr> <td>0000 0000 0000</td> <td>4 КБ</td> <td>VAddr[12]</td> </tr> <tr> <td>0000 0000 0011</td> <td>16 КБ</td> <td>VAddr[14]</td> </tr> <tr> <td>0000 0000 1111</td> <td>64 КБ</td> <td>VAddr[16]</td> </tr> <tr> <td>0000 0011 1111</td> <td>256 КБ</td> <td>VAddr[18]</td> </tr> <tr> <td>0000 1111 1111</td> <td>1 МБ</td> <td>VAddr[20]</td> </tr> <tr> <td>0011 1111 1111</td> <td>4 МБ</td> <td>VAddr[22]</td> </tr> <tr> <td>1111 1111 1111</td> <td>16 МБ</td> <td>VAddr[24]</td> </tr> </tbody> </table> <p>В столбце Page Mask приведены все возможные значения Page Mask. Поскольку каждая пара битов этого поля всегда имеет одинаковое значение, физическая строка в TLB содержит сокращенную версию Page Mask, содержащую только 6 бит. Однако для программы это значение всегда преобразуется в 12-битное.</p> <p>Следует иметь в виду, что при кэшируемых ссылках, страницы размером 4 Кбайт использовать нельзя.</p>	Page Mask[11:0]	Размер страницы	Бит определения четности	0000 0000 0000	4 КБ	VAddr[12]	0000 0000 0011	16 КБ	VAddr[14]	0000 0000 1111	64 КБ	VAddr[16]	0000 0011 1111	256 КБ	VAddr[18]	0000 1111 1111	1 МБ	VAddr[20]	0011 1111 1111	4 МБ	VAddr[22]	1111 1111 1111	16 МБ	VAddr[24]
	Page Mask[11:0]	Размер страницы	Бит определения четности																						
	0000 0000 0000	4 КБ	VAddr[12]																						
	0000 0000 0011	16 КБ	VAddr[14]																						
	0000 0000 1111	64 КБ	VAddr[16]																						
	0000 0011 1111	256 КБ	VAddr[18]																						
	0000 1111 1111	1 МБ	VAddr[20]																						
	0011 1111 1111	4 МБ	VAddr[22]																						
1111 1111 1111	16 МБ	VAddr[24]																							
VPN2[31:13]	<p>Виртуальный номер страницы без младшего разряда. Данное поле содержит старшие разряды виртуального номера страницы. Виртуальный номер соответствует двум страницам TLB. Конкретная страница TLB выбирается младшим разрядом виртуального адреса страницы. Разряды 31:25 всегда участвуют в сравнении. Участие в сравнении разрядов 24:13 зависит от размера страницы, задаваемого полем Page Mask.</p>																								
G	<p>Бит глобальности. Если он установлен, данная строка является глобальной для всех процессов и подпроцессов, и таким образом, поле ASID исключается из рассмотрения.</p>																								
ASID[7:0]	<p>Идентификатор адресного пространства. Определяет процесс или подпроцесс, с которым ассоциируется данная строка TLB.</p>																								
PFN0[31:12], PFN1[31:12]	<p>Физический номер кадра. Задаёт старшие разряды физического адреса. Для страниц размером более 4 Кбайт используется подмножество этого поля.</p>																								
C0[2:0], C1[2:0]	<p>Кэшируемость. Содержит закодированное значение атрибута кэшируемости и определяет должна ли страница помещаться в кэш или нет. Поле кодируется следующим образом:</p> <table border="1"> <thead> <tr> <th>C[2:0]</th> <th>Атрибуты когерентности</th> </tr> </thead> <tbody> <tr> <td>000</td> <td>При записи преобразуется в код 011</td> </tr> <tr> <td>001</td> <td>При записи преобразуется в код 011</td> </tr> <tr> <td>010</td> <td>Некэшируемая страница</td> </tr> <tr> <td>011</td> <td>Кэшируемая страница</td> </tr> <tr> <td>100</td> <td>При записи преобразуется в код</td> </tr> </tbody> </table>	C[2:0]	Атрибуты когерентности	000	При записи преобразуется в код 011	001	При записи преобразуется в код 011	010	Некэшируемая страница	011	Кэшируемая страница	100	При записи преобразуется в код												
	C[2:0]	Атрибуты когерентности																							
	000	При записи преобразуется в код 011																							
	001	При записи преобразуется в код 011																							
	010	Некэшируемая страница																							
	011	Кэшируемая страница																							
100	При записи преобразуется в код																								

		011	
	101	При записи преобразуется в код 011	
	110	При записи преобразуется в код 011	
	111	При записи преобразуется в код 010	
D0, D1	“Dirty” (Грязная страница) – бит разрешения записи. Показывает, что в страницу была сделана запись и/или разрешена запись в данную страницу. Если этот бит установлен, разрешены операции сохранения в данной странице. Если не установлен, сохранения в данной странице будут вызывать исключения модификации.		
V0, V1	Бит валидности. Показывает, что данная строка TLB и, соответственно, отображение виртуальной страницы, действительны. Если этот бит установлен, то обращения к данной странице разрешены. Если не установлен, то обращения к странице будут вызывать исключения TLB (TLB invalid).		

Для заполнения строки TLB используются команды TLBWI и TLBWR (см. документ “Процессорное ядро RISCore32. Система команд”). Перед запуском этих команд нужно обновить некоторые регистры CP0, записав в них значения, которые будут затем помещены в строку TLB.

- Значение Page Mask задается в регистре Page Mask CP0.
- Значения VPN2 и ASID задаются в регистре EntryHi CP0.
- Значения PFN0, C0, D0, V0 и G задаются в регистре EntryLo0 CP0.
- Значения PFN1, C1, D1, V1 и G задаются в регистре EntryLo1 CP0.

Биты глобальности G входят в оба регистра EntryLo0 и EntryLo1. Бит G строки TLB является результатом логической операции "И", проведенной над битами глобальности из EntryLo0 и EntryLo1. Более подробно эти регистры описаны в разделе 2.7 “Регистры CP0”.

Наличие идентификатора адресного пространства (ASID) дает возможность уменьшить частоту попаданий при поисках по TLB на контекстной основе. Это определяет возможность одновременного существования нескольких процессов как в TLB, так и в кэш команд. Значение ASID хранится в регистре EntryHi и сравнивается со значением ASID каждой строки.

#### 2.5.4 Преобразование виртуального адреса в физический в режиме TLB.

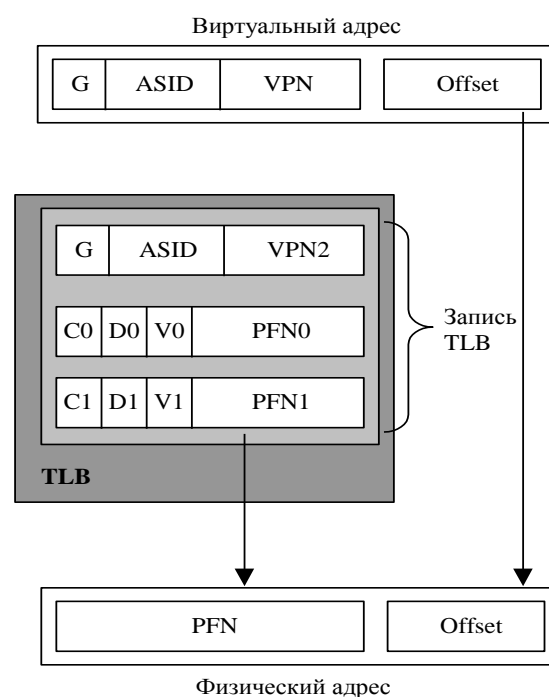
Преобразование виртуального адреса в физический начинается со сравнения полученного виртуального адреса с виртуальными адресами, хранящимися в TLB. Соответствие имеет место, если виртуальный номер страницы (VPN) адреса совпадает с полем VPN строки TLB с учетом маски, хранящейся в этой строке, а также выполняется одно из двух условий:

- Установлен бит глобальности (G) для четных и нечетных страниц в строке TLB;
- Поле ASID виртуального адреса совпадает с полем ASID строки TLB.

Это соответствие называется попаданием TLB. Если не имеется ни одного соответствия, возникает исключение промаха TLB и программному обеспечению дается возможность пополнить TLB из расположенной в памяти таблицы страниц виртуальных /физических адресов. На Рисунок 2.13 показана логика преобразования виртуального адреса в физический.

На этом рисунке виртуальный адрес расширяется 8-разрядным идентификатором адресного пространства (ASID), который уменьшает частоту попаданий при просмотрах TLB на контекстной основе. Это 8-разрядное поле ASID содержит номер, присвоенный процессу, и хранится в регистре EntryHi CP0.

1. Виртуальный адрес (VA), представленный виртуальным номером страницы (VPN), сравнивается с тэгом из строки TLB (VPN2) с учетом маски (PageMask).
2. Если имеется соответствие, номер страничного кадра (PFN0 или PFN1, в зависимости от значения бита четности – самого старшего бита, не участвующего в сравнении) извлекается и помещается в старшие разряды физического адреса (PA)
3. В младшие разряды физического адреса помещается смещение (Offset), не участвующее в сравнении.



**Рисунок 2.13**

Когда происходит совпадение виртуальных адресов при поиске по TLB, физический номер кадра (PFN) извлекается из соответствующей физической порции строки TLB и дополняется смещением, взятым из виртуального адреса, формируя, таким образом, физический адрес. Смещение представляет собой адрес в пределах пространства страничного кадра. Как показано на рисунке, смещение не пропускается через TLB.

На Рисунок 2.14 показана блок-схема процесса преобразования адреса. В верхней части рисунка показан виртуальный адрес для страницы размером 4 Кбайт. Ширина поля смещения определяется размером страницы.

В нижней части рисунка показан виртуальный адрес для страницы размером 16 Мбайт.

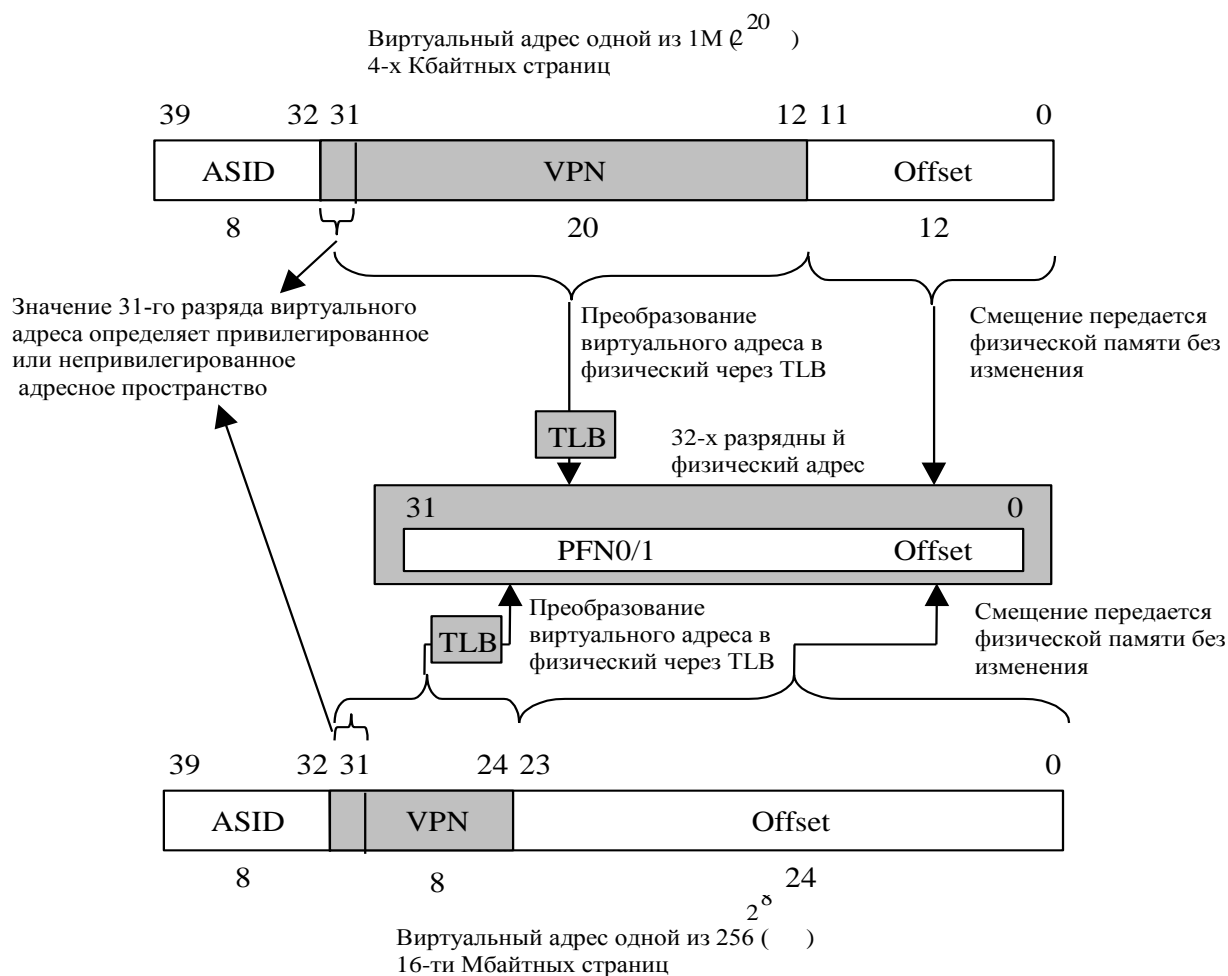


Рисунок 2.14

### 2.5.4.1 Попадания (hits), промахи (misses), и множественные попадания (multiple matches)

Каждая строка TLB содержит тэг и два поля данных. Если найдено соответствие, старшие разряды виртуального адреса заменяются физическим номером кадра (PFN), хранящимся в соответствующей строке массива данных TLB. Способ разбиения памяти при отображении определяется в терминах TLB-страниц. TLB поддерживает страницы различных размеров в пределах от 4 КБ до 16 МБ с шагом по степеням 4. Если соответствие найдено, но строка является запрещенной (т.е., бит V в поле данных равен 0), вырабатывается исключение TLB Invalid.

Если соответствие не найдено, возникает исключение TLB Refill, и программное обеспечение пополняет TLB из таблицы страниц, находящейся в памяти. На Рисунок 2.15 показан алгоритм преобразования и условия возникновения исключений TLB.

Программное обеспечение может делать записи в конкретные строки TLB или использовать аппаратный механизм записи в случайно выбранные строки. Регистр Random определяет, в какую строку будет сделана запись командой TLBWR. Этот регистр декрементируется на каждом такте продвижения конвейера, возвращаясь к максимальному значению после достижения величины, равной значению регистра Wired. Таким образом, строки TLB, чей номер меньше значения регистра Wired, не

затрагиваются командой TLBWR, что позволяет зарезервировать TLB-отображения первостепенной важности.

В режиме TLB также реализован механизм сравнения при записи с целью предотвращения возникновения нескольких соответствий (множественных попаданий). Работает он следующим образом. При выполнении операции записи в TLB, поле VPN2 сравнивается с одноименными полями всех строк TLB. Если будет найдено соответствие, возникнет аппаратно обрабатываемое исключение, которое установит бит TS регистра Status CP0 и прервет эту операцию. Подробно исключения описаны в п. 2.6. В каждой строке TLB имеется скрытый бит, обнуляемый при аппаратном сбросе. Устанавливается этот бит при записи в данную строку, разрешая просмотр этой строки при поисках соответствий. Поэтому непроинициализированные строки не вызывают неадекватные преобразования адресов.

Замечание: этот скрытый бит инициализации приводит все строки TLB к запрещенному состоянию после аппаратного сброса, что делает ненужной процедуру очистки (flush) TLB. Но для совместимости с другими MIPS – процессорами рекомендуется заполнять значения тэгов уникальными величинами и обнулять бит валидности (V).

Очистить строку TLB (вывести ее из рассмотрения при поиске) можно, записав в нее значение с неотображаемым через TLB адресом.

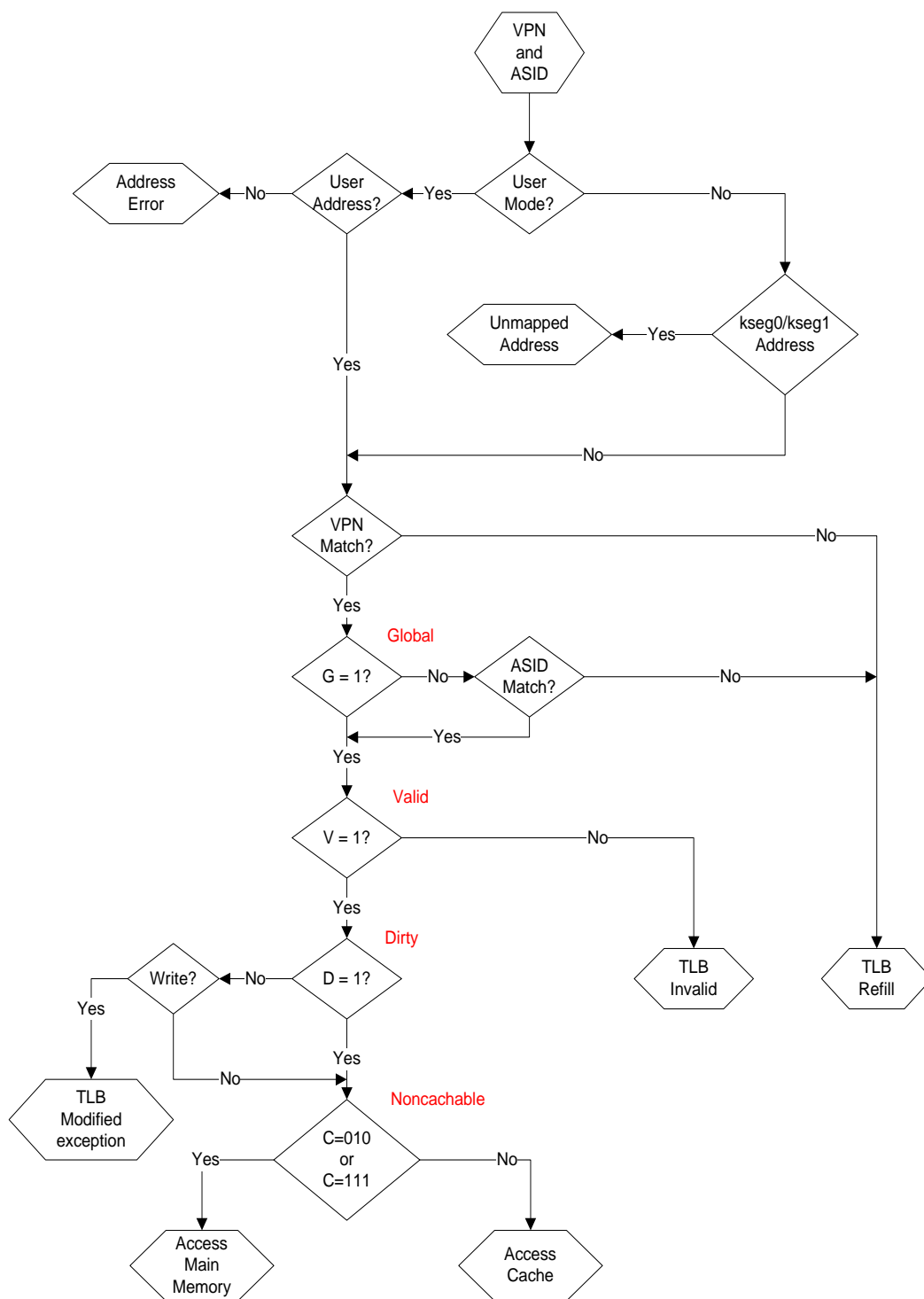
Смена размера маски или других переменных строки TLB не приводит к исключению, если она не вводит в противоречие данной строки с другими строками. Например, увеличение размера страницы расширением маски в одной строке TLB может привести к перекрытию данной страницы с другими страницами TLB.

#### **2.5.4.2 Размеры страниц и алгоритм замещения**

Для управления общим количеством отображаемого адресного пространства и характеристиками замещения в различных областях памяти ядро обеспечивает два механизма. Первый заключается в том, что размер страницы может быть задан относительно каждой строки TLB, что позволяет отображать страницы размером от 4 Кбайт до 16 Мбайт (по степеням 4). В регистр Page Mask CP0 загружается требуемый размер страницы, который при выполнении операции записи попадает в очередную строку TLB. Таким образом, операционная система может задавать отображения особых назначений. Например, характерный кадровый буфер (frame buffer) может быть отображен на память всего одной строкой TLB.

Второй механизм управляет замещением, когда возникает промах при просмотре TLB. Для выбора строки TLB, в которую будет записано новое отображение, в процессорном ядре предусмотрен алгоритм случайного замещения. Но существует также способ программно предотвратить случайное замещение зарезервированных отображений, количество которых определяется значением регистра Wired CP0. (см. также п. 2.7.3.6).





**Рисунок 2.15. Алгоритм преобразования адреса через TLB**

## 2.6 Исключения

Процессорное ядро способно принимать исключения от ряда источников, в том числе промах буфера преобразования адресов (TLB), арифметические переполнение, прерывание ввода-вывода, и системные вызовы. Обнаружив одно из этих исключений, CPU приостанавливает нормальную последовательность исполнения команд и процессор входит в режим Kernel.

В режиме Kernel ядро отключает прерывания и вынуждает процессор запустить программу обработчика исключений, расположенную в фиксированных адресах памяти. Обработчик сохраняет контекст процессора – содержимое счетчика команд, текущий режим процессора и статус разрешения прерываний. Таким образом, контекст может быть восстановлен по завершению обработки исключения.

При возникновении исключения в регистр Exception Program Counter (EPC) загружается адрес, начиная с которого исполнение команд может возобновиться после завершения обработки исключения. В регистр EPC помещается адрес команды, вызвавшей исключение или, если команда находилась в слоте задержки перехода, адрес команды перехода, предшествующей слоту задержки. Чтобы различить эти ситуации, программное обеспечение должно проанализировать бит BD (branch delay) в регистре Cause CP0.

### 2.6.1 Условия исключений

Исключения обрабатываются на стадии M конвейера. Когда исключительная ситуация обнаруживается, команда, находящаяся на стадии M, и все команды, следующие за ней на конвейере, отменяются. Соответственно, все условия остановки конвейера, относящиеся к этой команде, а также условия последующих исключений, которые также могут относиться к ней, игнорируются, поскольку обслуживание приостановок для отмененной команды не приносит выигрыша.

Когда условие исключения обнаруживается на стадии M, процессор заполняет необходимые регистры CP0 значениями, относящимися к состоянию исключения, изменяет счетчик команд (PC) на адрес соответствующего вектора обработки исключения и очищает признаки исключения, относящиеся к более ранним стадиям конвейера.

Такая реализация позволяет завершить исполнение команды, находящейся на стадии W, и запретить завершение последующих команд. Таким образом, значения, сохраненного в регистре EPC (в случае ошибок – в Epcg PC), достаточно для возобновления исполнения. Это также обеспечивает поступление исключений в соответствии с порядком исполнения команд – команда, вызывающая исключение, может быть уничтожена командой с более поздней стадии конвейера, также вызвавшей исключение.

### 2.6.2 Приоритеты исключений

В Таблица 2.3. перечислены все возможные исключения со своими относительными приоритетами от высшего к низшему. Некоторые из этих исключений могут случаться одновременно, в этом случае вызывается исключение с наивысшим приоритетом.

Таблица 2.3

Исключение	Описание
Reset	Аппаратный сброс
NMI	Внешнее немаскируемое прерывание и прерывание от таймера WDT (см. табл. 7.2).
TLB_Ri, TLB_Ii	Промах TLB при выборке команды, Попадание в запрещенную страницу TLB (V=0) при выборке команды
AdELi	Ошибка выравнивания адреса при выборке команды; Ссылка на адрес режима Kernel при работе в режиме User при выборке команды
MCheck Sys Bp CpU RI Ov Tr AdELd AdES	Запись в TLB, создающая конфликт с существующей строкой TLB Выполнение команды SYSCALL Выполнение команды BREAK Выполнение команды сопроцессора в режиме User Выполнение зарезервированной команды Переполнение в арифметической команде Выполнение trap (когда условие trap истинно) Ошибка выравнивания адреса при загрузке данных; Ссылка на адрес режима Kernel при работе в режиме User при загрузке данных Ошибка выравнивания адреса при сохранении данных; Попытка сохранения по адресу Kernel в режиме User
TLB_Rd, TLB_Id	Промах TLB при загрузке данных; Попадание в запрещенную страницу TLB (V=0) при загрузке данных
TLB_M	Сохранение в TLB-странице с D=0
Interrupt	Установка немаскируемых HW или SW - прерываний

### 2.6.3 Расположение векторов исключений

Векторы исключений аппаратного сброса и NMI всегда находятся по адресу 0xBFC\_0000. Адреса всех других исключений являются комбинациями векторных смещений и базового адреса. В Таблица 2.4 приведены базовые адреса как функции исключения и состояния бита BEV Регистра Status. В Таблица 2.5. приведены смещения от базового адреса как функции исключения. В Таблица 2.6 эти две таблицы сведены в одну таблицу, содержащую все возможные адреса векторов исключений как функции состояний, влияющих на выбор этих векторов.

Таблица 2.4.

Исключение	Status <sub>BEV</sub>	
	0	1
Reset, NMI	0xBFC0_0000	
Остальные исключения	0x8000_0000	0xBFC0_0200

**Таблица 2.5. Базовые адреса векторов исключений**

<b>Исключение</b>	<b>Смещение вектора</b>
TLB Refill, EXL = 0	0x000
Reset, NMI	0x000
Исключения общего характера (General Exceptions)	0x180
Interrupt, Cause <sub>IV</sub> = 1	0x200

Таблица 2.6. Векторы исключений

Исключение	BEV	EXL	IV	Вектор
Reset, NMI	–	–	–	0xBFC0_0000
TLB Refill	0	0	–	0x8000_0000
TLB Refill	0	1	–	0x8000_0180
TLB Refill	1	0	–	0xBFC0_0200
TLB Refill	1	1	–	0xBFC0_0380
Interrupt	0	0	0	0x8000_0180
Interrupt	0	0	1	0x8000_0200
Interrupt	1	0	0	0xBFC0_0380
Interrupt	1	0	1	0xBFC0_0400
Остальные	0	–	–	0x8000_0180
Остальные	1	–	–	0xBFC0_0380

### 2.6.4 Обработка общих исключений

Кроме исключений аппаратного сброса и NMI, которые обслуживаются особым образом, обработка всех остальных исключений происходит в соответствии со следующим основным маршрутом:

- Если бит EXL Регистра Состояния (Status) очищен, в регистр EPC загружается значение PC, по которому выполнение программы будет перезапущено, и при необходимости устанавливается бит BD в Регистре Причины (Cause). Если команда не находится в слоте задержки перехода, бит BD в Регистре Причины будет очищен, а в регистр EPC загружается значение, соответствующее текущему PC. Если же команда находится в слоте задержки перехода, бит BD в Регистре Причины устанавливается в “1”, и в EPC загружается значение, равное PC - 4. Если бит EXL в Регистре Состояния установлен, в регистр EPC ничего не загружается, и бит BD в Регистре Причины не модифицируется.
- В поля CE и ExcCode Регистра Причины загружаются значения, соответствующие исключению.
- Устанавливается бит EXL в Регистре Состояния (Status).
- Процессор стартует с вектора исключения.

Значение, загруженное в EPC, представляет собой адрес возврата из исключения и в обычной ситуации программе обработки исключения не требуется его модифицировать. Программе также не нужно просматривать бит BD в Регистре Причины, если не возникает потребность определить действительный адрес команды, вызвавшей исключение.

#### Operation:

```

if StatusEXL == 0 then
  if InstructionInBranchDelaySlot then
    EPC <= PC - 4
    CauseBD <= 1
  else
    EPC <= PC
    CauseBD <= 0
  endif
endif

```

```

if (ExceptionType == TLBRefill) then
vectorOffset <= 0x000
elseif (ExceptionType == Interrupt) and
(CauseIV == 1) then
vectorOffset <= 0x200
else
vectorOffset <= 0x180
endif
else
vectorOffset <= 0x180
endif
CauseCE <= FaultingCoprocesorNumber
CauseExcCode <= ExceptionType
StatusEXL <= 1
if (StatusBEV == 1) then
PC <= 0xBFC0_0200 + vectorOffset
else
PC <= 0x8000_0000 + vectorOffset
endif

```

## 2.6.5 Исключения

В следующих разделах описаны все исключения в порядке, соответствующем табл.2.4.

### 2.6.5.1 Исключение по аппаратному сбросу (Reset Exception)

Это немаскируемое исключение, которое происходит при установке сигнала аппаратного сброса. Когда возникает исключение аппаратного сброса, процессор выполняет полную начальную инициализацию, то есть приводит автоматы к начальному состоянию и переводит процессор в состояние, из которого он может начать запуск команд, находящихся в неэкзизируемой и неотображаемой области. После возникновения исключения аппаратного сброса состояние процессора не определено, за исключением следующего:

- Регистр Random устанавливается в значение, равное количеству строк TLB - 1.
- Регистр Wired устанавливается в 0.
- Регистр Config устанавливается в свое начальное состояние (boot state).
- Поля BEV, TS, NMI и ERL Регистра Status устанавливаются в заданные значения.
- В PC загружается значение 0xBFC0\_0000 (виртуальный адрес).

#### Вектор исключения:

Reset (0xBFC0\_0000)

#### Operation:

```

Random <= TLBEntries - 1
Wired <= 0
Config <= ConfigurationState
StatusBEV <= 1
StatusTS <= 0
StatusNMI <= 0

```

```
StatusERL <= 1
PC <= 0xBFC0_0000
```

### 2.6.5.2 Исключение по немаскируемому прерыванию (Non Maskable Interrupt – NMI Exception)

Немаскируемое прерывание возникает по положительному фронту входного сигнала NMI или при срабатывании сторожевого таймера WDT. Исключение NMI происходит только в пределах границ команды, поэтому оно не вызывает сброса или другую переинициализацию аппаратных средств. Состояние кэш, памяти, а также другие состояния процессора остаются неизменными. Значения регистров также сохраняются за исключением следующего:

- Поля BEV, TS, NMI и ERL регистра Status принимают заданные значения.
- В регистр ErrorEPC загружается значение PC - 4, если прерывание произошло на фоне команды в слоте задержки перехода. В противном случае в регистр ErrorEPC загружается значение PC.
- В PC загружается значение 0xBFC0\_0000.

#### Вектор исключения:

Reset (0xBFC0\_0000)

#### Operation:

```
StatusBEV <= 1
StatusTS <= 0
StatusNMI <= 1

StatusERL <= 1
if InstructionInBranchDelaySlot then
  ErrorEPC <= PC - 4
else
  ErrorEPC <= PC
endif
PC <= 0xBFC0_0000
```

### 2.6.5.3 Исключение по обновлению TLB — выборка команды или доступ к данным (TLB Refill Exception – Instruction Fetch or Data Access)

Исключение TLB Refill происходит во время выборки команды или доступа к данным, если в TLB нет ни одной строки, соответствующей ссылке к отображенному адресному пространству, и бит EXL в регистре Status равен 0.

#### Значение поля ExcCode регистра Cause:

TLBL: Произошла ссылка по загрузке данных или выборке команды

TLBS: Произошла ссылка по сохранению данных

#### Дополнительно сохраняемые состояния:

Таблица 2.7

Состояние регистра	Значение
--------------------	----------

BadVAddr	ошибочный адрес
Context	поле BadVPN2 содержит VA <sub>31:13</sub> ошибочного адреса
EntryHi	поле VPN2 содержит VA <sub>31:13</sub> ошибочного адреса; поле ASID содержит ASID отсутствующей ссылки

#### Вектор исключения:

Вектор TLB Refill (смещение 0x000)

#### 2.6.5.4 Исключение TLB Invalid — выборка команды или доступ к данным (TLB Invalid Exception – Instruction Fetch or Data Access)

Исключение TLB Invalid происходит во время выборки команды или доступа к данным в одном из следующих случаев:

- В TLB нет ни одной строки, соответствующей ссылке к отображенному адресному пространству, и бит EXL в регистре Status равен 1.
- Строка TLB соответствует ссылке к отображенному адресу, но ее бит валидности выключен.

#### Значение поля ExcCode регистра Cause:

TLBL: Произошла ссылка по загрузке данных или выборке команды

TLBS: Произошла ссылка по сохранению данных

#### Дополнительно сохраняемые состояния:

Таблица 2.8

Состояние регистра	Значение
BadVAddr	ошибочный адрес
Context	поле BadVPN2 содержит VA <sub>31:13</sub> ошибочного адреса
EntryHi	поле VPN2 содержит VA <sub>31:13</sub> ошибочного адреса; поле ASID содержит ASID отсутствующей ссылки

#### Вектор исключения:

Общий Вектор исключения (смещение 0x180)

#### 2.6.5.5 Исключение по ошибке адресации — выборка команды / доступ к данным (Address Error Exception – Instruction Fetch / Data Access)

Исключение по ошибке адресации во время доступа к команде или данным возникает при попытке выполнить одно из следующих действий:

- Выбрать команду, загрузить или сохранить слово данных, если они не выровнены в границах слова
- Загрузить или сохранить половину слова, если оно не выровнено в границах половины слова
- Обратиться по адресу пространства Kernel при работе в режиме User



**Значение поля ExcCode регистра Cause:**

ADEL: Произошла ссылка по загрузке данных или выборке команды

ADES: Произошла ссылка по сохранению данных

**Дополнительно сохраняемые состояния:****Таблица 2.9**

Состояние регистра	Значение
BadVAddr	ошибочный адрес

**Вектор исключения:**

Общий Вектор исключения (смещение 0x180)

**2.6.5.6 Исключение по аппаратному контролю (Mcheck – Machine Check Exception)**

Данное исключение возникает, если при выполнении команды записи в TLB (TLBWI или TLBWR) обнаруживается, что поле виртуального адреса записываемой строки соответствует такому же полю одной из строк, уже хранящихся в TLB.

При возникновении данной ситуации запись в TLB не выполняется и устанавливается бит TS в регистре Status. Этот бит является статусным и не влияет на функционирование процессорного ядра. Сбрасывается он программно после разрешения данной ситуации, осуществляемого очисткой конфликтных строк в TLB.

**Значение поля ExcCode регистра Cause:**

Mcheck

**Дополнительно сохраняемые состояния:**

Нет

**Вектор исключения:**

Общий Вектор исключения (смещение 0x180)

**2.6.5.7 Исключение исполнения – системный вызов (System Call Exception)**

Исключение System Call является одним из шести исключений исполнения. Все такие исключения имеют одинаковый приоритет. Исключение System Call возникает при исполнении команды SYSCALL.

**Значение поля ExcCode регистра Cause:**

Sys

**Дополнительно сохраняемые состояния:**

Нет

**Вектор исключения:**

Общий Вектор исключения (смещение 0x180)

### **2.6.5.8 Исключение исполнения — Breakpoint (Execution Exception – Breakpoint)**

Исключение Breakpoint является одним из шести исключений исполнения. Все такие исключения имеют одинаковый приоритет. Исключение Breakpoint возникает при исполнении команды BREAK.

**Значение поля ExhCode регистра Cause:**

Вр

**Дополнительно сохраняемые состояния:**

Нет

**Вектор исключения:**

Общий Вектор исключения (смещение 0x180)

### **2.6.5.9 Исключение исполнения — зарезервированная команда (Execution Exception – Reserved Instruction)**

Исключение зарезервированной команды является одним из шести исключений исполнения. Все такие исключения имеют одинаковый приоритет. Исключение зарезервированной команды вызывается при исполнении команды с неопределенным кодом операции или полем функции.

**Значение поля ExhCode регистра Cause:**

RI

**Дополнительно сохраняемые состояния:**

Нет

**Вектор исключения:**

Общий Вектор исключения (смещение 0x180)

### **2.6.5.10 Исключение исполнения — недоступен сопроцессор (Execution Exception – Coprocessor Unusable)**

Исключение недоступности сопроцессора является одним из шести исключений исполнения. Все такие исключения имеют одинаковый приоритет. Исключение недоступности сопроцессора вызывается при попытке исполнения команды сопроцессора CP0 в режиме User.

**Значение поля ExhCode регистра Cause:**

SpU

**Дополнительно сохраняемые состояния:**

Нет

**Вектор исключения:**

Общий Вектор исключения (смещение 0x180)

### 2.6.5.11 Исключение исполнения — целочисленное переполнение (Execution Exception – Integer Overflow)

Исключение целочисленного переполнения является одним из шести исключений исполнения. Все такие исключения имеют одинаковый приоритет. Исключение целочисленного переполнения вызывается, когда выбранные целочисленные команды приводят к переполнению в двоичном коде.

**Значение поля ExhCode регистра Cause:**

0v

**Дополнительно сохраняемые состояния:**

Нет

**Вектор исключения:**

Общий Вектор исключения (смещение 0x180)

### 2.6.5.12 Исключение исполнения — Trap (Execution Exception – Trap)

Исключение Trap является одним из шести исключений исполнения. Все такие исключения имеют одинаковый приоритет. Исключение Trap вызывается, если условие команды trap истинно (TRUE).

**Значение поля ExhCode регистра Cause:**

Tr

**Дополнительно сохраняемые состояния:**

Нет

**Вектор исключения:**

Общий Вектор исключения (смещение 0x180)

### 2.6.5.13 Исключение сохранения в запрещенной области (TLB Modified Exception)

Это исключение возникает при обращении по записи данных к отображенному адресу, если выполняется следующее условие:

- Найденная строка TLB действительна, но страница запрещена для записи.

**Значение поля ExhCode регистра Cause:**

Mod

**Дополнительно сохраняемые состояния:****Таблица 2.10**

Состояние регистра	Значение
--------------------	----------

BadVAddr	Ошибочный адрес
Context	Поля BadVPN2 содержат VA <sub>31:13</sub> ошибочного адреса
EntryHi	Поле VPN2 содержит VA <sub>31:13</sub> ошибочного адреса; поле ASID содержит ASID отсутствующей ссылки

**Вектор исключения:**

Общий Вектор исключения (смещение 0x180)

**2.6.5.14 Исключение прерывания (Interrupt Exception)**

Исключение прерывания возникает, когда сигнал одного или более разрешенных регистром Status прерываний устанавливается на входе процессора.

**Значение поля ExhCode регистра Cause:**

Int

**Дополнительно сохраняемые состояния:****Таблица 2.11**

Состояние регистра	Значение
Cause <sub>IP</sub>	Указывает код прерывания

**Вектор исключения:**

Общий Вектор исключения (смещение 0x180), если бит IV регистра Cause равен 0;

Вектор прерывания (смещение 0x200), если бит IV регистра Cause равен 1.

**2.6.6 Алгоритмы обработки исключений**

В этом разделе приведены алгоритмы обработки следующих исключений:

- Общие исключения;
- Исключения пропуска при поиске по TLB;
- Исключения Reset и NMI;

Исключения аппаратно обрабатываются, а затем программно обслуживаются.

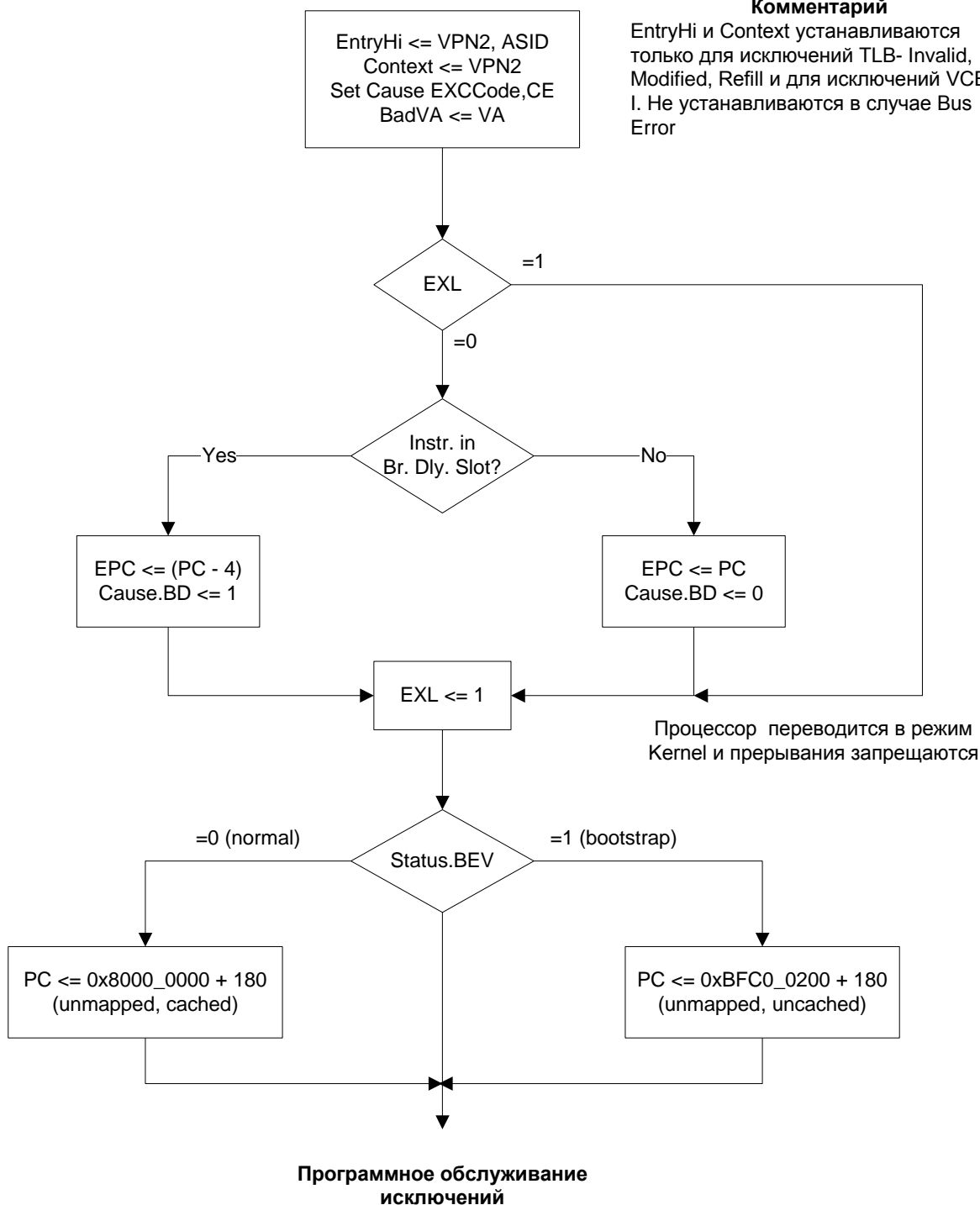
Алгоритмы обработки исключений приведены на Рисунок 2.16,

Рисунок 2.17, Рисунок 2.18.

Все исключения кроме Reset, NMI и TLB-miss первого уровня. Прерывания могут быть замаскированы битами IE и IM

**Комментарий**

EntryHi и Context устанавливаются только для исключений TLB- Invalid, Modified, Refill и для исключений VCED/I. Не устанавливаются в случае Bus Error



**Рисунок 2.16** Обработка общих исключений

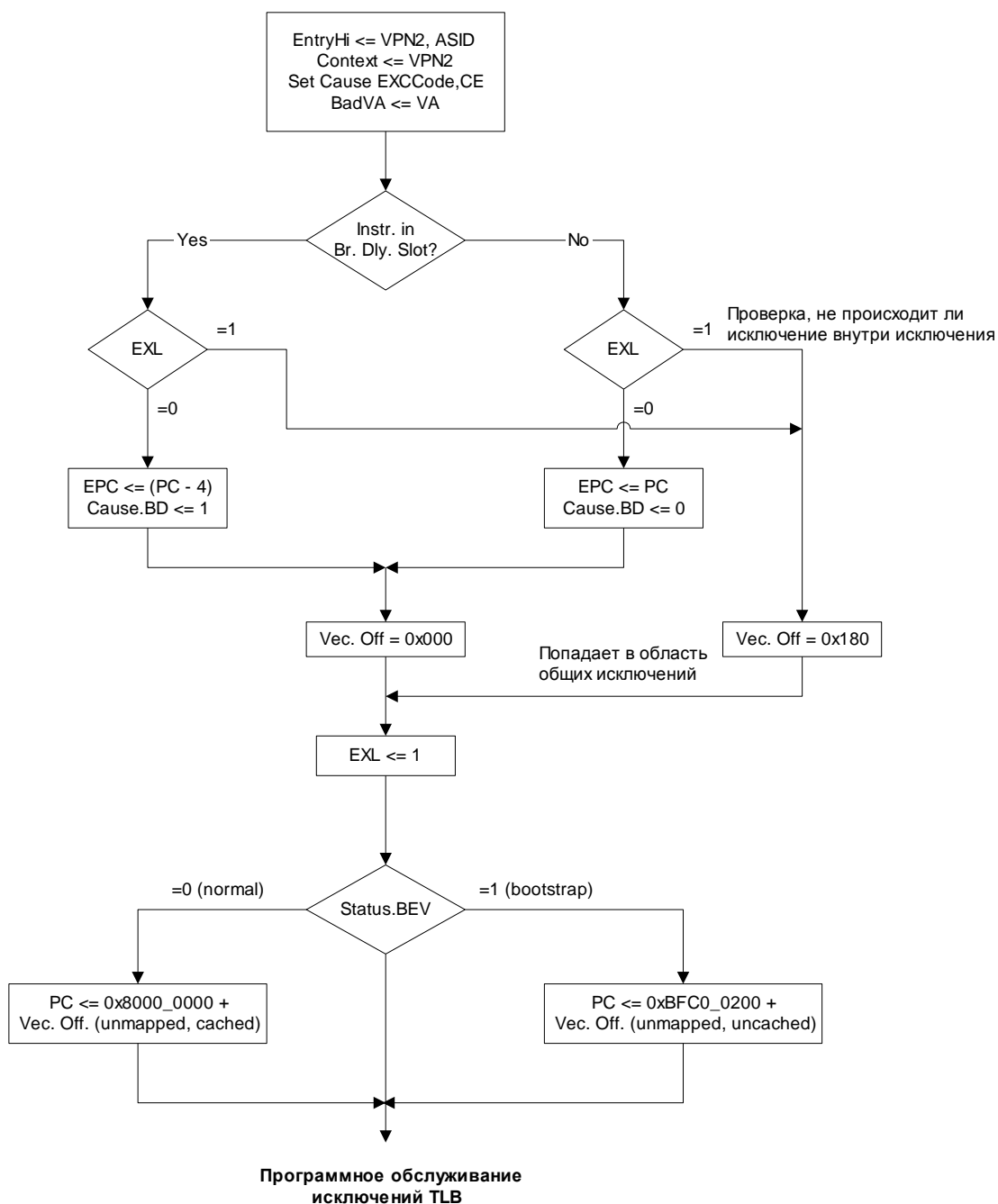
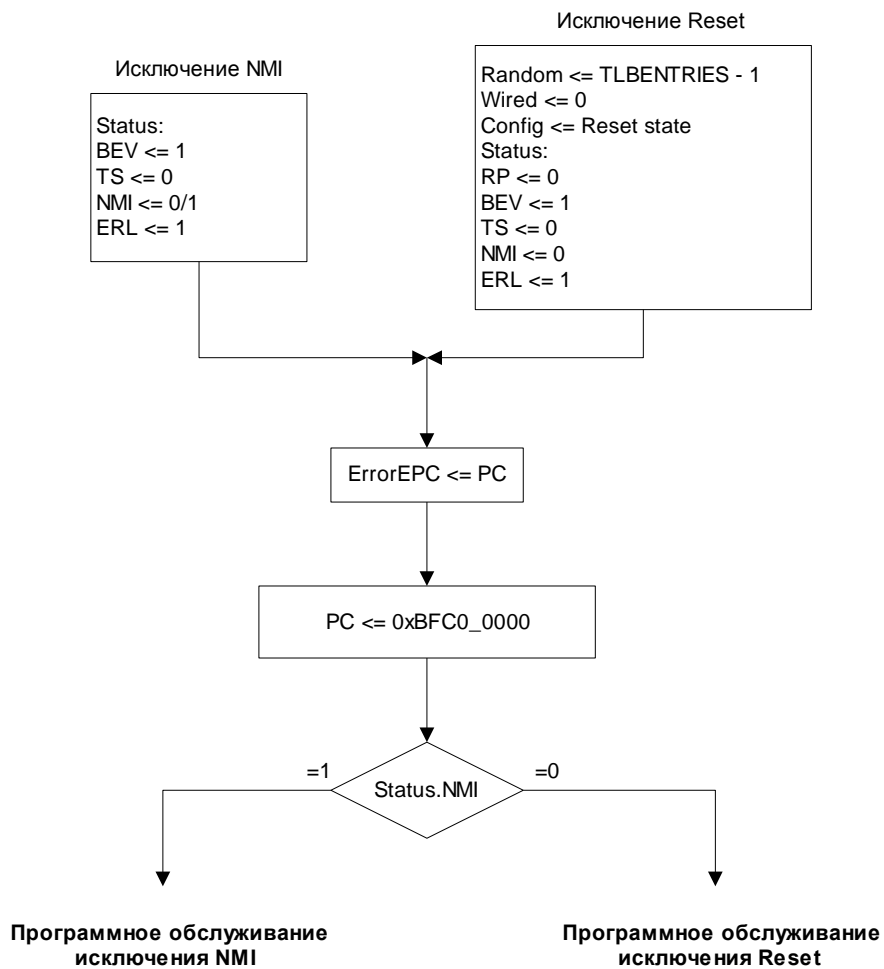


Рисунок 2.17 Обработка исключений TLB Refill и TLB Invalid



**Рисунок 2.18. Обработка исключений Reset и NMI**

## 2.7 Регистры CP0

### 2.7.1 Назначение

Системный Управляющий Сопроцессор (CP0) обеспечивает регистровый интерфейс с процессорным ядром MIPS32 и поддерживает управление памятью, преобразование адреса, обработку исключений и другие привилегированные операции. Каждому регистру CP0 соответствует определяющий его уникальный номер; этот номер называется *номером регистра*. Например, регистру PageMask соответствует 5-й номер регистра.

После записи нового значения в регистр CP0 (с помощью команды MTC0), его обновление происходит не сразу, а по прошествии периода от 0 и более команд. Этот период называется периодом особой ситуации.

### 2.7.2 Обзор регистров CP0

В таблице 2.13. приведены все регистры CP0 в порядке возрастания нумерации. В разделе 2.7.3 каждый из этих регистров описан отдельно.



Таблица 2.13. Регистры CP0

Номер регистра	Название регистра	Функция
0	Index <sup>1</sup>	Индекс матрицы TLB (режим TLB)
1	Random <sup>1</sup>	Случайным образом сгенерированный индекс для буфера TLB (режим TLB)
2	EntryLo0 <sup>1</sup>	Младшая часть строки TLB для виртуальных страниц с четными номерами (режим TLB)
3	EntryLo1 <sup>1</sup>	Младшая часть строки TLB для виртуальных страниц с нечетными номерами (режим TLB)
4	Context <sup>2</sup>	Указатель на строку в таблице страниц памяти (режим TLB)
5	PageMask <sup>1</sup>	Управление переменным размером страниц строк TLB (режим TLB)
6	Wired <sup>1</sup>	Управление количеством закрепленных “привязанных” строк TLB (режим TLB)
7	Reserved	Резерв
8	BadVAddr <sup>2</sup>	Содержит адрес, вызвавший последнее связанное с адресацией исключение
9	Count <sup>2</sup>	Счетчик процессорных циклов
10	EntryHi <sup>1</sup>	Старшая часть строки TLB (режим TLB)
11	Compare <sup>2</sup>	Управление прерыванием таймера
12	Status <sup>2</sup>	Состояние и управление процессором
13	Cause <sup>2</sup>	Причина последнего исключения
14	EPC <sup>2</sup>	Значение счетчика команд во время последнего исключения
15	PRId	Идентификация и ревизия процессора
16	Config/Config1	Конфигурационный регистр
17	LLAddr	Загрузка адреса сопряжения
18-19	Не реализованы	
20-22	Reserved	Резерв
23-24	Не реализованы	
25-27	Reserved	Резерв
28-29	Не реализованы	
30	ErrorEPC <sup>2</sup>	Значение счетчика команд при последней ошибке
31	Не реализован	

<sup>1</sup>Регистры, используемые при управлении памятью.

<sup>2</sup>Регистры, используемые при обработке исключений.

### 2.7.3 Регистры CP0

Регистры CP0 обеспечивают интерфейс между системой команд (ISA) и архитектурой процессора. Каждый регистр, описанный в этом разделе, представлен своим порядковым номером и значением поля select.

Все поля описанных регистров характеризуются свойствами записи / чтения, а также значением после аппаратного сброса. Свойства записи / чтения охарактеризованы в таблице 2.14.

Таблица 2.12

Свойства записи/чтения	Аппаратная интерпретация	Программная интерпретация
R/W	Поле, в котором все биты программно и аппаратно доступны по записи и чтению. Аппаратное обновление этого поля доступно для программы при чтении программой. Программное обновление этого поля доступно для процессора при чтении процессором. Если значение поля после сброса не определено, программа или процессор должны проинициализировать это поле, чтобы первое чтение возвратило предсказуемое значение.	
R	Поле, значение которого постоянно или обновляется только процессором. Значение поля после начальной установки восстанавливается также при включении питания. Если значение поля не определено после начальной установки, процессор обновляет его только при условиях, определенных при описании поля.	Поле, для которого значение, записанное программой, процессором игнорируется. Программное прочтение этого поля возвращает последнее обновленное процессором значение. Если значение поля не определено после начальной установки, программное прочтение этого поля возвратит непредсказуемое значение кроме тех случаев, когда произошло обновление процессором значения этого поля по возникновению условий, определенных в описании поля условий.
0	Поле, значение которого процессором не обновляется и всегда равно нулю.	Программное чтение всегда возвращает нуль.

### 2.7.3.1 Регистр Index (Регистр 0 CP0, Select 0)

Регистр Index является 32-х разрядным регистром, доступным для чтения и записи. Он содержит индекс доступа к TLB для команд TLBP, TLBR и TLBWI. Ширина поля индекса зависит от количества строк TLB и равна 4.

Функционирование процессора НЕОПРЕДЕЛЕНО, если в регистр Index записано значение большее или равное количеству строк TLB.

#### Формат регистра Index

31	30		
4	3	0	
R	0		Index

Таблица 2.13. Описание полей регистра Index

Поля		Описание	Чтение / Запись	Начальное состояние
Имя	Биты			
P	31	Неудачная проба. Устанавливается в 1, если предыдущей командой TLBProbe (TLBP) не было найдено соответствия в TLB.	R	Не определено
0	30:4	При чтении возвращается ноль	0	0
Index	3:0	Индекс строки TLB, к которой относятся команды TLBRead и TLBWrite	R/W	Не определено

### 2.7.3.2 Регистр Random (Регистр CP0 1, Select 0).

Регистр Random доступен только для чтения, и его значение используется как индекс TLB для команды TLBWR. Ширина поля Random определяется таким же образом, как для регистра Index.

Значение этого регистра изменяется между верхней и нижней границами следующим образом:

- Нижняя граница определяется количеством строк TLB, зарезервированных для использования операционной системой (содержимое регистра Wired). Строка, чей индекс равен значению Wired, является первой из доступных для записи командой TLB Write Random (TLBWR).
- Верхняя граница равна общему количеству строк TLB минус 1.

Регистр Random уменьшается на 1 при продвижении конвейера RISC, возвращаясь к максимальному значению по достижению величины, равной значению регистра Wired.

Процессор инициализирует регистр Random значением, равным верхней границе по возникновению исключения Reset и по записи в регистр Wired.

#### Формат регистра Random

31				
4	3	0		
0				Random

Таблица 2.14. Описание полей регистра Random

Поля		Описание	Чтение / Запись	Начальное состояние
Имя	Биты			
0	31:4	При чтении возвращается ноль	0	0
Random	3:0	Случайный индекс строки TLB	R	TLB Entries - 1

### 2.7.3.3 EntryLo0, EntryLo1 (Регистры 2 и 3 CP0, Select 0)

Пара регистров EntryLo действует как интерфейс между TLB и командами TLBR, TLBWI, TLBWR.

В режиме TLB EntryLo0 содержит строки для четных страниц TLB, а EntryLo1 – для нечетных страниц.

После ошибки адресации и возникновения исключений TLB refill, TLB invalid и TLB modified, содержимое регистров EntryLo0 и EntryLo1 не определено.

### Формат регистров EntryLo0, EntryLo1

31	30	29	26	25					6	5	
3	2	1	0								
R	0			PFN				C	D	V	G

Таблица 2.15. Описание полей регистров EntryLo0 и EntryLo1

Поля		Описание	Чтение / Запись	Начальное состояние
Имя	Биты			
R	31:30	Резервные. При чтении возвращается нуль	R	0
0	29:26	При чтении возвращается нуль	R	0
PFN	25:6	Номер страничного кадра. Соответствует битам 31:12 физического адреса.	R/W	Не определено
C	5:3	Атрибут когерентности страницы. См. табл.2.18.	R/W	Не определено
D	2	“Dirty” – бит, разрешающий запись. Указывает на то, что в страницу была сделана запись, и/или страница открыта для записи. Если этот бит равен 1, разрешается сохранение в этой странице. Если он равен 0, сохранение в этой странице вызывает исключение TLB Modified.	R/W	Не определено
V	1	Бит валидности. Указывает, на то, что строка TLB и, соответственно, отображение виртуальной страницы, является действительным. Если этот бит равен 1, доступ к странице разрешается. Если этот бит равен 0, доступ к странице вызывает исключение TLB Invalid.	R/W	Не определено
G	0	Бит глобальности. При записи в TLB битом G в строке TLB становится логическое “И” битов G EntryLo0 и EntryLo1. Если бит G строки TLB равен 1, результат сравнения полей ASID игнорируется при поиске по TLB. При чтении строки TLB биты G EntryLo0 и EntryLo1 отражают состояние бита G TLB.	R/W	Не определено

В Таблица 2.16. приведена кодировка для поля C регистров EntryLo0 и EntryLo1 и полей K0, K23 и KU регистра Config.

Таблица 2.16. Атрибуты когерентности Кэш

Значение C[5:3]	Описание
0, 1, 3*, 4, 5, 6	Кэшируемая, некогерентная область

2*, 7	Некэшируемая область
* - Архитектура MIPS32 предусматривает только эти два значения. Остальные значения не используются и отображаются в используемые значения. Например, 0, 1, 4, 5 и 6 отображаются в 3, а 7 – в 2.	

### 2.7.3.4 Регистр Context (Регистр 4 CP0, Select 0)

Регистр Context доступен для чтения и записи, и содержит указатель на строку в матрице PTE (page table entry). Эта матрица является структурой данных операционной системы, в которой содержатся преобразования виртуального адреса в физический. При возникновении промаха TLB, операционная система загружает в TLB недостающее преобразование из матрицы PTE. Регистр Context дублирует часть информации, содержащейся в регистре BadVAddr, но организован таким образом, что операционная система может прямо ссылаться к 8-байтной матрице PTE в памяти.

При возникновении исключения TLB (TLB Refill, TLB Invalid, или TLB Modified) биты VA<sub>31:13</sub> виртуального адреса записываются в поле BadVPN2 регистра Context. Поле PTEBase записывается и используется операционной системой.

После возникновения исключения ошибки адресации значение поля BadVPN2 регистра Context не определено.

#### Формат регистра Context

31	23	22
4 3	0	
PTEBase	BadVPN2	

Таблица 2.17. Описание полей регистра Context

Поля		Описание	Чтение / Запись	Начальное состояние
Имя	Биты			
PTEBase	31:23	Это поле используется операционной системой и обычно содержит значение, позволяющее операционной системе использовать регистр Context в качестве указателя на текущую матрицу PTE в памяти.	R/W	Не определено
BadVPN2	22:4	Это поле заполняется процессором при промахе TLB. Оно содержит биты VA <sub>31:13</sub> пропущенного виртуального адреса	R	Не определено
0	3:0	При чтении возвращается нуль	0	0

### 2.7.3.5 Регистр PageMask (Регистр 5 CP0, Select 0)

Регистр PageMask доступен для чтения и записи, и используется для чтения TLB и записи в TLB. Он содержит маску сравнения, которая устанавливает переменную размера страниц для каждой строки TLB, как показано в таблице 2.21.

Если значение регистра отлично от значений, приведенных в таблице, поведение процессора при поиске по TLB не определено.

### Формат регистра PageMask

31 0	25 24	13 12
0	Mask	0

Таблица 2.18. Описание полей регистра PageMask

Поля		Описание	Чтение / Запись	Начальное состояние
Имя	Биты			
Mask	24:13	Бит маски, содержащий “1”, указывает на то, что соответствующий бит виртуального адреса не должен принимать участие при поиске соответствия по TLB	R/W	Не определено
0	31:25, 12:0	При чтении возвращается нуль	0	0

Таблица 2.19. Таблица возможных значений поля Mask регистра PageMask.

Размер страницы	Бит											
	24	23	22	21	20	19	18	17	16	15	14	13
<b>4 КБАЙТ</b>	0	<b>0</b>	0	0	0	0	0	0	0	0	0	0
16 Кбайт	0	0	0	0	0	0	0	0	0	0	1	1
64 Кбайт	0	0	0	0	0	0	0	0	1	1	1	1
256 Кбайт	0	0	0	0	0	0	1	1	1	1	1	1
1 Мбайт	0	0	0	0	1	1	1	1	1	1	1	1
4 Мбайт	0	0	1	1	1	1	1	1	1	1	1	1
16 Мбайт	1	1	1	1	1	1	1	1	1	1	1	1

#### 2.7.3.6 Регистр Wired (Регистр 6 CP0, Select 0)

Регистр Wired доступен для чтения и записи. Этот регистр определяет границу между случайными и “привязанными” строками TLB, как показано на Рисунок 2.19. Ширина поля Wired определяется так же, как для описанного выше регистра Index. “Привязанные” строки зафиксированы, то есть они не являются удаляемыми и не могут быть перезаписаны командой TLBWR. Эти строки могут быть перезаписаны только командой TLBWI.

Регистр Wired устанавливается в нулевое состояние исключением по аппаратному сбросу (Reset). Запись в регистр Wired вызывает установку регистра Random в значение, равное его верхней границе.

Если значение, записанное в регистр Wired, больше или равно числу строк TLB, операция процессора не определена.



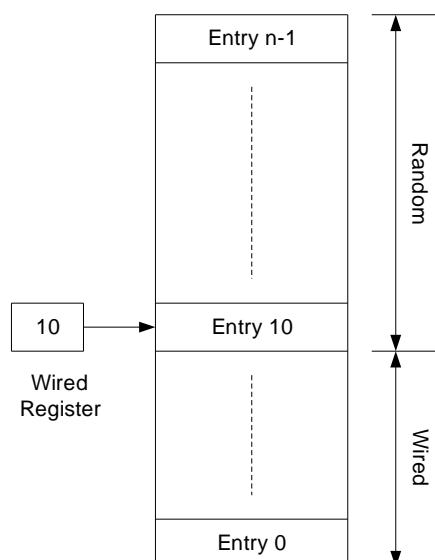


Рисунок 2.19. “Привязанные” и случайные строки TLB

#### Формат регистра Wired

31		
4	3	0
		0
		Wired

Таблица 2.20. Описание полей регистра Wired

Поля		Описание	Чтение / Запись	Начальное состояние
Имя	Биты			
0	31:4	При чтении возвращается нуль	0	0
Wired	3:0	Граница между “привязанными” и случайными строками TLB.	R/W	0

#### 2.7.3.7 Регистр BadVAddr (Регистр 8 CP0, Select 0)

Регистр BadVAddr доступен только для чтения и содержит последний виртуальный адрес, вызвавший одно из следующих исключений:

- Ошибка адреса (AdEL или AdES)
- TLB Refill
- TLB Invalid
- TLB Modified

#### Формат регистра BadVAddr

31		
0		
		BadVAddr

Таблица 2.21. Описание полей регистра BadVAddr

Поля		Описание	Чтение / Запись	Начальное состояние
Имя	Биты			
BadVAddr	31:0	Виртуальный адрес, вызвавший исключение	R	Не определено

### 2.7.3.8 Регистр Count (Регистр 9 CP0, Select 0)

Регистр Count действует как таймер, увеличивающий свое значение каждый такт.

Регистр Count может быть записан в функциональных или диагностических целях, включая установку или синхронизацию процессора.

#### Формат регистра Count

31 0
COUNT

Таблица 2.22. Описание полей регистра Count

Поля		ОПИСАНИЕ	Чтение / Запись	Начальное состояние
Имя	Биты			
Count	31:0	Счетчик	R/W	Не определено

### 2.7.3.9 Регистр EntryHi (Регистр 10 CP0, Select 0)

Регистр EntryHi содержит информацию соответствия виртуального адреса, используемого при чтении, записи и операциях доступа к TLB.

При возникновении исключений TLB (TLB Refill, TLB Invalid или TLB Modified) биты VA<sub>31:13</sub> виртуального адреса записываются в поле VPN2 регистра EntryHi. В поле ASID, которое используется в процессе сравнения при поиске по TLB, программно записывается идентификатор текущего адресного пространства.

Поле VPN2 регистра EntryHi не определено после прерывания по ошибке адресации.

#### Формат регистра EntryHi

31 0		
VPN2	0	ASID

Таблица 2.23. Описание полей регистра EntryHi

Поля		Описание	Чтение / Запись	Начальное состояние
Имя	Биты			
VPN2	31:13	Разряды VA <sub>31:0</sub> виртуального адреса (виртуальный номер страницы, деленный на 2). Это поле записывается аппаратно при исключении TLB или при чтении TLB, и программно перед записью в TLB.	R/W	Не определено
0	12:8	При чтении возвращается нуль	0	0
ASID	7:0	Идентификатор адресного пространства. Это поле записывается аппаратно при чтении TLB, и программно при установке текущего значения ASID для записи в TLB и для сравнения при поиске по TLB с соответствующими полями ASID в строках TLB.	R/W	Не определено

### 2.7.3.10 Регистр Compare (Регистр 11 CP0, Select 0)

Регистр Compare действует совместно с регистром Count с целью реализации функции таймера и прерывания по таймеру. Прерывание по таймеру является выходным сигналом процессора.

Результат сравнения регистров Count и Compare заведен на 19 разряд регистра QSTR. Когда значение регистра Count равняется значению регистра Compare, этот бит имеет единичное состояние. Он остается в этом состоянии, пока в регистр Compare не будет произведена запись.

Для диагностических целей регистр Compare доступен для чтения и записи. Однако при нормальном функционировании регистр Compare используется только для записи. При записи значения в регистр Compare в качестве побочного эффекта происходит очистка бита прерывания по таймеру.

#### Формат регистра Compare

31	Compare
0	

Таблица 2.24. Описание полей регистра Compare

Поля		Описание	Чтение / Запись	Начальное состояние
Имя	Биты			
Compare	31:0	Период счета таймера	R/W	Не определено

### 2.7.3.11 Регистр Status (Регистр 12 CP0, Select 0)

Регистр Status (SR) является регистром, доступным для чтения и записи. Он содержит поля рабочего режима, разрешения прерываний и диагностические состояния процессора. Для задания режимов функционирования процессора, поля этого регистра объединяются следующим образом:

**Разрешение прерываний:** Прерывания разрешаются, когда истинны все следующие условия:

- IE = 1
- EXL = 0
- ERL = 0

Если эти условия выполнены, прерывания разрешаются установкой битов IM.

**Рабочие режимы:** Процессор всегда находится в одном из двух режимов – Kernel или User. Режим задается установкой следующих битов регистра Status CPU.

- Режим User: UM = 1, EXL = 0, and ERL = 0
- Режим Kernel: UM = 0 или EXL = 1 или ERL = 1

#### Формат Status регистра

31	28	27	26	23	22	21	20	19	18	16	15	8	7	5	4	3
2	1	0														
CU3- CU0	0	0	BEV	TS	0	NMI	0	IM7- IM0	0	U M	0	ERL	EXL	I E		

Таблица 2.25. Описание полей регистра Status

Поля		Описание	Чтение / Запись	Начальное состояние
Имя	Биты			
CU3- CU0	31:28	Управление доступом к сопроцессорам 3, 2, 1 и 0 соответственно: 0 – доступ запрещен; 1 – доступ разрешен. Сопроцессор 0 всегда доступен в режиме kernel в не зависимости от состояния бита CU0. Сопроцессоров 1,2 и 3 в CPU нет. Обращение к ним запрещено, так как это приведет к непредсказуемой ситуации	R/W	Не определено
-	27	Не используется	0	0
-	26:23	При чтении возвращается нуль	0	0
BEV	22	Управление размещением векторов исключения: 0: Нормальный 1: Начальная загрузка	R/W	1
TS	21	TLB-закрытие системы. Этот бит устанавливается, если при выполнении команд TLBWI или TLBWR образуется команда, которая приводит к условию закрытия, если оно разрешено. Программа может записывать в этот разряд только 0, чтобы очистить его, и не может вызвать переход этого бита из 0 в 1.	R/W	0
NMI	19	Указывает, что вход в вектор исключения начальной установки был осуществлен по причине возникновения NMI. 0: Не NMI (Аппаратный сброс) 1: NMI Программное обеспечение может записывать в этот бит только 0, чтобы очистить его, и не может записать 1.	R/W	1 для NMI, иначе 0
-	18:16	При чтении возвращается нуль	0	0
IM[7:0]	15:8	Маска прерываний: управление разрешением внешних, внутренних и программных прерываний. Прерывание принимается в случае, если установлен бит IE регистра Status и установлены соответствующие биты как в поле IM[7:0] регистра Status, так и в поле IP[7:0] регистра Cause. 0: Запрос на прерывание не разрешен.	R/W	Не определено

Поля		Описание	Чтение / Запись	Начальное состояние
Имя	Биты			
		1: Запрос на прерывание разрешен.		
-	7:5	При чтении возвращается нуль	0	0
UM	4	Указывает на то, что процессор работает в непривилегированном режиме (User): 0: Процессор работает в привилегированном режиме (Kernel) 1: Процессор работает в непривилегированном режиме (User) Замечание: процессор может также находиться в режиме Kernel, если установлены биты EXL или ERL. Это условие не влияет на состояние бита UM.	R/W	Не определено
-	3	При чтении возвращается нуль	0	0
ERL	2	Уровень ошибки. Устанавливается процессором при возникновении исключений Reset и NMI. 0: Нормальный уровень 1: Уровень ошибки Когда бит ERL установлен: Процессор находится в режиме Kernel. Прерывания запрещены. Команда ERET использует адрес возврата, содержащийся в ErrorEPC вместо EPC. kuseg используется как неотображаемая и некэшируемая область. Это позволяет иметь доступ к главной памяти при ошибках кэш. Поведение процессора не определено, если бит ERL установлен при выполнении кода из useg/kuseg.	R/W	1
EXL	1	Уровень Исключения. Устанавливается процессором при возникновении любого исключения, кроме Reset и NMI. 0: Нормальный уровень 1: Уровень исключения Когда бит EXL установлен: Процессор переходит в привилегированный режим (Kernel). Прерывания запрещены. Исключения TLB Refill используют общий вектор исключения вместо	R/W	Не определено

Поля		Описание	Чтение / Запись	Начальное состояние
Имя	Биты			
		вектора TLB Refill. Если происходит другое исключение, ЕРС не модифицируется.		
IE	0	Разрешение Прерывания. 0: Отключает прерывания 1: Разрешает прерываниям	R/W	Не определено

### 2.7.3.12 Регистр Cause (Регистр 13 CP0, Select 0)

Регистр Cause, в основном, описывает причину последнего исключения. Кроме того, поля регистра управляют запросами на программные прерывания и определяют вектор, которым обрабатываются прерывания. Все поля регистра Cause, за исключением IP[1:0], IV и WP, доступны только для чтения.

#### Формат регистра Cause

31	30	24	23	22	16	15	10	9	8	7	6
2	1	0									
BD	0	IV		0	IP[7:2]	IP[1:0]	0	Exc Code	0		

Таблица 2.26. Описание полей регистра Cause

Поля		Описание	Чтение / Запись	Начальное состояние
Имя	Биты			
BD	31	Указывает на то, что последнее исключение произошло в слоте задержки перехода: 0: Не в слоте задержки 1: В слоте задержки Замечание: бит BD не модифицируется на новом исключении, если установлен бит EXL.	R	Не определено
0	30:24	При чтении возвращается нуль	0	0
IV	23	Указывает, какой вектор используется для обслуживания исключений прерывания – общий или специальный вектор прерываний: 0: Используется общий вектор исключения (0x180) 1: Используется специальный вектор прерываний (0x200)	R/W	Не определено
0	22:16	При чтении возвращается нуль	0	0
IP[7:2]	15:10	Указывает, какое прерывание установлено: 15: все внутренние прерывания от DMA и устройств микроконтроллера (объединены по ИЛИ); 14: не используется, всегда имеет	R	Не определено

Поля		Описание	Чтени е/ Запись	Начальное состояние
Имя	Биты			
		нулевое состояние; 13: внешнее прерывание nIRQ[3]; 12: внешнее прерывание nIRQ[2]; 11: внешнее прерывание nIRQ[1]; 10: внешнее прерывание nIRQ[0].		
IP[1:0]	9:8	Управляет запросами программных прерываний (посредством записи «1» в данные разряды): 9: Запрос программного прерывания 1; 8: Запрос программного прерывания 0.	R/W	Не определено
ID	7	Прерывание от встроенных средств отладки программ (OnCD).	R/W	0
Exc Code	6:2	Код исключения — см. Таблица 2.27		
0	1:0	При чтении возвращается нуль	0	0

Таблица 2.27. Описание поля Exc Code регистра Cause

Значение Exc Code	Мнемоник а	Описание
0	Int	Прерывание
1	Mod	TLB-исключение модификации
2	TLBL	TLB-исключение (загрузка или вызов команды)
3	TLBS	TLB-исключение (сохранение)
4	AdEL	Прерывание по ошибке адресации (загрузка или вызов команды)
5	AdES	Прерывание по ошибке адресации (сохранение)
6-7		Не используются
8	Sys	Системное исключение
9	Bp	Исключение Breakpoint
10	RI	Исключение зарезервированной команды
11	SpU	Исключение недоступности сопроцессора
12	Ov	Исключение целочисленного переполнения
13	Tr	Исключение Trap
14-22		Зарезервированы
23		Не используется
24	MCheck	Аппаратный контроль
25-31		Зарезервированы

### 2.7.3.13 Регистр EPC (Регистр 14 CP0, Select 0)

Программный счетчик исключения (EPC) является регистром, доступным для чтения и записи. EPC содержит адрес, начиная с которого возобновляется исполнение программы после завершения обработки исключения. Все биты регистра EPC значимы и должны перезаписываться.

Для синхронных (точных) исключений, EPC содержит одно из следующего:



- Виртуальный адрес команды, которая была прямой причиной исключения;
- Виртуальный адрес команды перехода (Branch или Jump), непосредственно предшествующей исключению, если команда, вызвавшая исключение, находится в слоте задержки перехода и установлен бит BD в регистре Cause.

Если установлен бит EXL в регистре Status, процессор не записывает адрес в регистр EPC при возникновении новых исключений. Однако, новое значение можно записать в EPC командой MTC0.

#### Формат регистра EPC

31 0	EPC
---------	-----

Таблица 2.28. Описание полей регистра EPC

Поля		Описание	Чтение / Запись	Начальное состояние
Имя	Биты			
EPC	31:0	Программный счетчик исключения	R/W	Не определено

#### 2.7.3.14 Регистр PRId (Регистр 15 CP0, Select 0)

Регистр идентификации процессора (PRId) – это 32-х разрядный регистр, доступный только для чтения. Он содержит информацию, идентифицирующую изготовителя, опции изготовителя, идентификацию процессора, и версию процессора.

#### Формат регистра PRId

31 0	24 23	16 15	8 7
R	Company ID	Processor ID	Revision

Таблица 2.29. Описание полей регистра PRId

Поля		Описание	Чтение / Запись	Начальное состояние
Имя	Биты			
R		При чтении возвращается нуль	R	0
Company ID	23:16	Идентификация компании, которая проектировала или изготовляла процессор.	R	1010
Processor ID	15:8	Идентификация типа процессора.	R	10010
Revision	7:0	Номер версии процессора. Позволяет программам различать разные версии одного типа процессора.	R	0

### 2.7.3.15 Регистр Config (Регистр 16 CP0, Select 0)

Регистр Config определяет различную конфигурационную информацию, а также информацию о возможностях процессора. Большинство полей регистра Config инициализируется аппаратно при выполнении исключения Reset или имеет постоянное значение, и только поле K0 должно быть проинициализировано программно обработчиком исключения Reset.

#### Формат регистра Config

31	30	28	27	25	24	21	20	19	18	17	16	15	14	13	12	10	9
7	6	3	2	0													
M	K23	KU	0	MD	R	MM	B	BE	AT	AR	MT	0	K0				
				U			M										

Таблица 2.30. Описание полей регистра Config

Поля		Описание	Чтение / Запись	Начальное состояние
Имя	Биты			
M	31	Этот бит аппаратно устанавливается в высокий уровень, указывая на наличие регистра Config1	R	1
K23	30:28	Это поле управляет кэшируемостью адресных сегментов kseg2 и kseg3 в режиме FM. В режиме TLB не используется. См. табл.2.33.	FM:R/W	FM:010
			TLB:R	TLB:000
KU	27:25	Это поле управляет кэшируемостью адресных сегментов kuseg и useg в режиме FM. В режиме TLB не используется. См. табл.2.33.	FM:R/W	FM:010
			TLB:R	TLB:000
0	24:21	Не используются	0	0
MDU	20	Тип MDU: итеративный умножитель и делитель	R	1
R	19	При чтении возвращается нуль	0	0
MM	18:17	Режим No Merging для 32 bit collapsing write buffer	R	0
BM	16	Тип передачи Burst: последовательный	R	0
BE	15	Режим endian: Little endian	R	0
AT	14:13	Тип архитектуры, реализованной процессором: MIPS32.	R	0
AR	12:10	Номер версии: 1	R	0
MT	9:7	Тип MMU: 1: Стандартный TLB (FM = 0) 3: Фиксированное отображение (FM = 1) 0, 2, 4-7: зарезервированы	R	TLB: 01
				FM: 11
R	6:3	При чтении возвращается нуль	0	0
K0	2:0	Алгоритм когерентности для kseg0, см. Таблица 2.16.	R/W	010

Таблица 2.31. Атрибуты когерентности кэш

Значение C[5:3]	
0, 1, 3*, 4, 5, 6	Кэшируемая, некогерентная область
2*, 7	Некэшируемая область
* - Архитектура MIPS32 предусматривает только эти два значения. Остальные значения не используются и отображаются в используемые значения. Например, 0, 1, 4, 5 и 6 отображается в 3, а 7 – в 2.	

### 2.7.3.16 Регистр Config1 (Регистр 16 CP0, Select 1)

Регистр Config1 является дополнением к регистру Config и кодирует дополнительную информацию о возможностях процессора. Все поля регистра Config1 доступны только для чтения.

#### Формат регистра Config1

31	30	25	24	22	21	19	18	16	15	13	12	10	9	7	6	5	4	3
2	1	0																
R	MMUSize	IS	IL	IA	DS	DL	DA	R	PC	WR	CA	EP	FP					

Таблица 2.32. Описание полей регистра Config1

Поля		Описание	Чтение / Запись	Начальное состояние
Имя	Биты			
R	31	При чтении возвращается нуль	0	0
Размер MMU	30:25	Это поле содержит количество строк TLB минус 1. В режиме TLB возвращается код 15 в десятичном формате, в режиме Fixed Mapping – 0.	R	001111 (FM=0)
				000000 (FM=1)
IS	24:22	Количество наборов кэш команд: резервная опция	R	111
IL	21:19	Размер строки кэш команд: 16 байт	R	011
IA	18:16	Тип кэш команд: Direct mapped	R	0
DS	15:13	Нет кэш данных	R	0
DL	12:10	Нет кэш данных	R	0
DA	9:7	Нет кэш данных	R	0
R	6:5	При чтении возвращается нуль	0	0
PC	4	Нет регистра Performance Counter	R	0
WR	3	Нет регистра WATCH	R	0
CA	2	Не реализовано	R	0
EP	1	EJTAG не реализован	R	0
FP	0	Нет плавающей арифметики	R	0

### 2.7.3.17 Регистр LLAddr – Load Linked Address (Регистр 17 CP0, Select 0)

Регистр LLAddr содержит физический адрес последней команды Load Linked (LL). Этот регистр используется только для диагностических целей.

### Формат LLAddr регистра

31 0	28 27	
0	Paddr[31:4]	

Таблица 2.33. Описание полей LLAddr регистра

Поля		ОПИСАНИЕ	Чтение / Запись	Начальное состояние
Имя	Биты			
0	31:28	При чтении возвращается нуль	0	0
Paddr[31: 4]	27:0	Физический адрес последней команды LL	R	Не определено

### 2.7.3.18 Регистр ErrorEPC (Регистр 30 CP0, Select 0)

Доступный для чтения и записи, регистр ErrorEPC полностью подобен регистру EPC, но используется при возникновении исключений ошибок. Все биты регистра ErrorEPC значимы и должны перезаписываться. Регистр ErrorEPC также используется для сохранения значения счетчика команд при возникновении исключений Reset и немаскируемого прерывании (NMI).

Регистр ErrorEPC содержит виртуальный адрес, начиная с которого может возобновиться исполнение программы после обработки ошибочной ситуации.

Этот адрес может быть:

- Виртуальным адресом команды, вызвавшей исключение;
- Виртуальным адресом команды перехода (Branch или Jump), непосредственно предшествующей исключению, если команда, вызвавшая ошибку, находится в слоте задержки перехода.

В отличие от регистра EPC, для регистра ErrorEPC не имеется соответствующего признака слота задержки перехода.

### Формат регистра ErrorEPC

31 0	
ErrorEPC	

Таблица 2.34. Описание полей регистра ErrorEPC

Поля		Описание	Чтение / Запись	Начальное состояние
Имя	Биты			
ErrorEPC	31:0	Счетчик команд при исключении ошибки	R/W	Не определен

Регистры WatchLo, WatchHi, Debug, DEPC, TagLo, DataLo, DeSave не реализованы

## 2.8 Кэш

### 2.8.1 Введение

В данной версии процессора реализован виртуально индексируемый и контролируемый по физическому тэгу кэш команд типа `direct mapped`. Это позволяет осуществлять доступ к кэш параллельно с преобразованием виртуального адреса в физический. Объем кэш составляет 16 Кбайт.

Загрузка кэш (операция `Refill`) выполняются посредством пачки (`burst`), состоящей из 4 команд. Адрес, по которому начинается `burst`, выровнен по 16-байтной границе. До получения критического слова кэш блокируется.

### 2.8.2 Протокол кэш

#### 2.8.2.1 Организация кэш

Кэш команд состоит из двух массивов – массива тэгов и массива данных. Кэш индексируется виртуально, поскольку для выбора соответствующей строки в обоих массивах используется виртуальный адрес. Контроль осуществляется по физическому тэгу, так-так массив тэгов содержит физический, а не виртуальный адрес.

На Рисунок 2.20 представлен формат каждой строки массивов тэгов и данных. Тэговая строка содержит 18 старших бита физического адреса (биты [31:14]) и бит валидности.

Строка данных содержит 4 32-х разрядных слова – всего 16 байт.

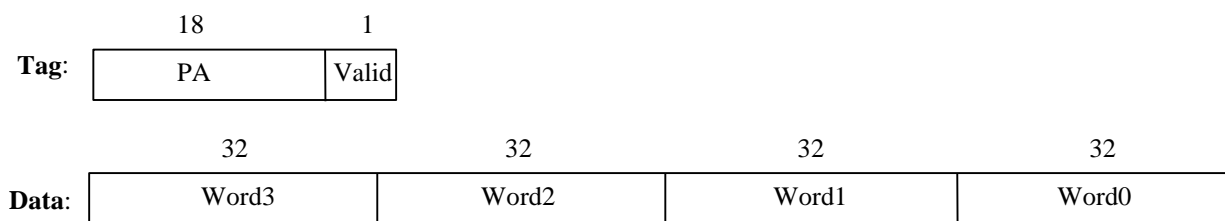


Рисунок 2.20 Формат массива кэш

#### 2.8.2.2 Атрибуты кэшируемости.

В данной версии реализовано только два атрибута. Область может быть либо кэшируемой, либо некэшируемой (см. Таблица 2.31)

## 2.9 Карта памяти CPU

Карта физической памяти CPU приведена в Таблица 2.35. Здесь и далее, если это не оговорено специально, коды адреса и данных указаны в шестнадцатеричной системе счисления. Объемы областей памяти указаны с учетом ее дальнейшего расширения.

Таблица 2.35. Карта физической памяти CPU

Диапазон адресов	Название области	Объем области, Мбайт

FFFF_FFFF 2000_0000	Внешняя память	3584
1FFF_FFFF 1C00_0000	Внешняя память (ПЗУ)	64
1BFF_FFFF 1800_0000	Внутренняя память	64
17FF_FFFF 0000_0000	Внешняя память	384

Вся внешняя память доступна через порт внешней памяти (MPORT).

Для CPU все адресное пространство памяти является 32-разрядным. Память SRAM, а также внешняя память, могут адресоваться с точностью до байта.

При DMA обменах вся память является словной (32 разряда).

Карта внутренней памяти 1892BM3T приведена в таблице 2.38.

**Таблица 2.36. Карта внутренней памяти 1892BM3T**

Диапазон адресов	Название области	Объем области, Кбайт
1BFF_FFFF 1880_0000	Резерв	56000
187F_FFFF 1840_0000	Память и регистры DSP-ядра	4096
183F_FFFF 1830_0000	Резерв	1024
182F_FFFF 182F_0000	Регистры CPU	64
182E_FFFF 1801_0000	Резерв	3000
1800_FFFF 1800_0000	Память SRAM	64

**Перечень программно доступных регистров для CPU приведен в**

**Таблица 2.37.**

Таблица 2.37

Условное обозначение регистра	Название регистра	Адрес регистра
<u>Регистры DMA SPORT</u>		
CSR_SpTx0	Регистр управления и состояния канала SpTx0	182F_0000
CP_SpTx0	Регистр указателя цепочки канала SpTx0	182F_0008
IR_SpTx0	Индексный регистр памяти канала SpTx0	182F_000C
OR_SpTx0	Регистр смещения памяти канала SpTx0	182F_0010
Y_SpTx0	Регистр параметров направления Y при двухмерной адресации памяти канала SpTx0	182F_0014
CSR_SpRx0	Регистр управления и состояния канала SpRx0	182F_0100
CP_SpRx0	Регистр указателя цепочки канала SpRx0	182F_0108
IR_SpRx0	Индексный регистр памяти канала SpRx0	182F_010C
OR_SpRx0	Регистр смещения памяти канала SpRx0	182F_0110
Y_SpRx0	Регистр параметров направления Y при двухмерной адресации памяти канала SpRx0	182F_0114
CSR_SpTx1	Регистр управления и состояния канала SpTx1	182F_0200
CP_SpTx1	Регистр указателя цепочки канала SpTx1	182F_0208
IR_SpTx1	Индексный регистр памяти канала SpTx1	182F_020C
OR_SpTx1	Регистр смещения памяти канала SpTx1	182F_0210
Y_SpTx1	Регистр параметров направления Y при двухмерной адресации памяти канала SpTx1	182F_0214
CSR_SpRx1	Регистр управления и состояния канала SpRx1	182F_0300
CP_SpRx1	Регистр указателя цепочки канала SpRx1	182F_0308
IR_SpRx1	Индексный регистр памяти канала SpRx1	182F_030C
OR_SpRx1	Регистр смещения памяти канала SpRx1	182F_0310
Y_SpRx1	Регистр параметров направления Y при двухмерной адресации памяти канала SpRx1	182F_0314
<u>Регистры DMA LPORT</u>		
CSR_LpCh0	Регистр управления и состояния канала LpCh0	182F_0400
CP_LpCh0	Регистр указателя цепочки канала LpCh0	182F_0408
IR_LpCh0	Индексный регистр памяти канала LpCh0	182F_040C
OR_LpCh0	Регистр смещения памяти канала LpCh0	182F_0410
Y_LpCh0	Регистр параметров направления Y при двухмерной адресации памяти канала LpCh0	182F_0414
CSR_LpCh1	Регистр управления и состояния канала LpCh1	182F_0500
CP_LpCh1	Регистр указателя цепочки канала LpCh1	182F_0508
IR_LpCh1	Индексный регистр памяти канала LpCh1	182F_050C
OR_LpCh1	Регистр смещения памяти канала LpCh1	182F_0510
Y_LpCh1	Регистр параметров направления Y при двухмерной адресации памяти канала LpCh1	182F_0514



CSR_LpCh2	Регистр управления и состояния канала LpCh2	182F_0600
CP_LpCh2	Регистр указателя цепочки канала LpCh2	182F_0608
IR_LpCh2	Индексный регистр памяти канала LpCh2	182F_060C
OR_LpCh2	Регистр смещения памяти канала LpCh2	182F_0610
Y_LpCh2	Регистр параметров направления Y при двухмерной адресации памяти канала LpCh2	182F_0614
CSR_LpCh3	Регистр управления и состояния канала LpCh3	182F_0700
CP_LpCh3	Регистр указателя цепочки канала LpCh3	182F_0708
IR_LpCh3	Индексный регистр памяти канала LpCh3	182F_070C
OR_LpCh3	Регистр смещения памяти канала LpCh3	182F_0710
Y_LpCh3	Регистр параметров направления Y при двухмерной адресации памяти канала LpCh3	182F_0714
<u>Регистры DMA MemCh</u>		
CSR_MemCh0	Регистр управления и состояния канала MemCh0	182F_0800
IOR_MemCh0	Регистр индекса и смещения внутренней памяти канала MemCh0	182F_0804
CP_MemCh0	Регистр указателя цепочки канала MemCh0	182F_0808
IR_MemCh0	Индексный регистр внешней памяти канала MemCh0	182F_080C
OR_MemCh0	Регистр смещения внешней памяти канала MemCh0	182F_0810
Y_MemCh0	Регистр параметров направления Y при двухмерной адресации внешней памяти канала MemCh0	182F_0814
Run0	Псевдорегистр управления состоянием бита RUN регистра CSR_MemCh0	182F_0818
CSR_MemCh1	Регистр управления и состояния канала MemCh1	182F_0900
IOR_MemCh1	Регистр индекса и смещения внутренней памяти канала MemCh1	182F_0904
CP_MemCh1	Регистр указателя цепочки канала MemCh1	182F_0908
IR_MemCh1	Индексный регистр внешней памяти канала MemCh1	182F_090C
OR_MemCh1	Регистр смещения внешней памяти канала MemCh1	182F_0910
Y_MemCh1	Регистр параметров направления Y при двухмерной адресации внешней памяти канала MemCh1	182F_0914
Run1	Псевдорегистр управления состоянием бита RUN регистра CSR_MemCh1	182F_0918
CSR_MemCh2	Регистр управления и состояния канала MemCh2	182F_0A00
IOR_MemCh2	Регистр индекса и смещения внутренней памяти канала MemCh2	182F_0A04
CP_MemCh2	Регистр указателя цепочки канала MemCh2	182F_0A08

IR_MemCh2	Индексный регистр внешней памяти канала Memh2	182F_0A0C
OR_MemCh2	Регистр смещения внешней памяти канала MemCh2	182F_0A10
Y_MemCh2	Регистр параметров направления Y при двухмерной адресации внешней памяти канала MemCh2	182F_0A14
Run2	Псевдорегистр управления состоянием бита RUN регистра CSR_MemCh2	182F_0A18
CSR_MemCh3	Регистр управления и состояния канала MemCh3	182F_0B00
IOR_MemCh3	Регистр индекса и смещения внутренней памяти канала MemCh3	182F_0B04
CP_MemCh3	Регистр указателя цепочки канала MemCh3	182F_0B08
IR_MemCh3	Индексный регистр внешней памяти канала Memh3	182F_0B0C
OR_MemCh3	Регистр смещения внешней памяти канала MemCh3	182F_0B10
Y_MemCh3	Регистр параметров направления Y при двухмерной адресации внешней памяти канала MemCh3	182F_0B14
Run3	Псевдорегистр управления состоянием бита RUN регистра CSR_MemCh3	182F_0B18
<u>Регистры LPORT</u>		
LTx0	Буфер передачи порта LPORT0	182F_7000
LRx0	Буфер приема порта LPORT0	182F_7000
LCSR0	Регистр управления и состояния порта LPORT0	182F_7004
LDIR0	Регистр управления порта ввода-вывода LPORT0	182F_7008
LDR0	Регистр данных порта ввода-вывода LPORT0	182F_700C
LTx1	Буфер передачи порта LPORT1	182F_8000
LRx1	Буфер приема порта LPORT1	182F_8000
LCSR1	Регистр управления и состояния порта LPORT1	182F_8004
LDIR1	Регистр управления порта ввода-вывода LPORT1	182F_8008
LDR1	Регистр данных порта ввода-вывода LPORT1	182F_800C
LTx2	Буфер передачи порта LPORT2	182F_9000
LRx2	Буфер приема порта LPORT2	182F_9000
LCSR2	Регистр управления и состояния порта LPORT2	182F_9004
LDIR2	Регистр управления порта ввода-вывода LPORT2	182F_9008
LDR2	Регистр данных порта ввода-вывода LPORT2	182F_900C
LTx3	Буфер передачи порта LPORT3	182F_A000
LRx3	Буфер приема порта LPORT3	182F_A000
LCSR3	Регистр управления и состояния порта LPORT3	182F_A004
LDIR3	Регистр управления порта ввода-вывода LPORT3	182F_A008

<u>Регистры SPORT</u>		
STx0	Буфер передачи данных порта SPOTR0	182F_5000
Rx0	Буфер приема данных SPOTR0	182F_5000
STCTL0	Регистр управления передачей данных SPOTR0	182F_5004
SRCTL0	Регистр управления приемом данных SPOTR0	182F_5008
TDIV0	Регистр коэффициентов деления при передаче данных SPOTR0	182F_500C
RDIV0	Регистр коэффициентов деления при приеме данных SPOTR0	182F_5010
MTCS0	Выбор канала передачи данным в многоканальном режиме SPOTR0	182F_5014
MRCS0	Выбор канала приема данным в многоканальном режиме SPOTR0	182F_5018
KEYWD0	Регистр кода сравнения SPOTR0	182F_501C
KEYMASK0	Регистр маски сравнения SPOTR0	182F_5020
MRCE0	Выбор канала для сравнения принимаемых данных SPOTR0	182F_5024
STx1	Буфер передачи данных порта SPOTR1	182F_6000
SRx1	Буфер приема данных SPOTR1	182F_6000
STCTL1	Регистр управления передачей данных SPOTR1	182F_6004
SRCTL1	Регистр управления приемом данных SPOTR1	182F_6008
TDIV1	Регистр коэффициентов деления при передаче данных SPOTR1	182F_600C
RDIV1	Регистр коэффициентов деления при приеме данных SPOTR1	182F_6010
MTCS1	Выбор канала передачи данным в многоканальном режиме SPOTR1	182F_6014
MRCS1	Выбор канала приема данным в многоканальном режиме SPOTR1	182F_6018
KEYWD1	Регистр кода сравнения SPOTR1	182F_601C
KEYMASK1	Регистр маски сравнения SPOTR1	182F_6020
MRCE1	Выбор канала для сравнения принимаемых данных SPOTR1	182F_6024
<u>Регистры UART</u>		
RBR	Приемный буферный регистр	182F_3000
THR	Передающий буферный регистр	182F_3000
IER	Регистр разрешения прерываний	182F_3004
IIR	Регистр идентификации прерывания	182F_3008
FCR	Регистр управления FIFO	182F_3008
LCR	Регистр управления линией	182F_300C
MCR	Регистр управления модемом	182F_3010
LSR	Регистр состояния линии	182F_3014
MSR	Регистр состояния модемом	182F_3018
SPR	Регистр Scratch Pad	182F_301C
DLL	Регистр делителя младший	182F_3000
DLM	Регистр делителя старший	182F_3004
SCLR	Регистр предделителя (scaler)	182F_3014

<u>Регистры ИТ</u>		
ITCSR	Регистр управления	182F_D000
ITPERIOD	Регистр периода работы таймера	182F_D004
ITCOUNT	Регистр счетчика	182F_D008
ITSCALE	Регистр предделителя	182F_D00C
<u>Регистры WDT</u>		
WTCSR	Регистр управления	182F_D010
WTPERIOD	Регистр периода работы таймера	182F_D014
WTCOUNT	Регистр счетчика	182F_D018
WTSCALE	Регистр предделителя	182F_D01C
<u>Регистры RTT</u>		
RTCSR	Регистр управления	182F_D020
RTPERIOD	Регистр периода работы таймера	182F_D024
RTCOUNT	Регистр счетчика	182F_D028
<u>Регистры MPORT</u>		
CSCON0	Регистр конфигурации 0.	182F_1000
CSCON1	Регистр конфигурации 1.	182F_1004
CSCON2	Регистр конфигурации 2.	182F_1008
CSCON3	Регистр конфигурации 3.	182F_100C
CSCON4	Регистр конфигурации 4.	182F_1010
SDRCON	Регистр конфигурации памяти SDRAM	182F_1014
CKE_CTR	Регистр управления состоянием вывода СКЕ микросхемы	182F_1018
<u>Системные регистры</u>		
MASKR	Регистр маски	182F_4000
QSTR	Регистр заявок	182F_4004
CSR	Регистр управления и состояния	182F_4008

## 3. ЦИФРОВОЙ СИГНАЛЬНЫЙ ПРОЦЕССОР (DSP)

### 3.1 Функциональные параметры и возможности DSP

В СБИС 1892ВМ3Т в качестве DSP используется процессорное ядро ELcore-14. Оно имеет типичную для многих цифровых процессоров обработки сигналов (ЦПОС) гарвардскую архитектуру с внутренним параллелизмом по потокам обрабатываемых данных и предназначено для высокоскоростной обработки информации в форматах с фиксированной и с плавающей точкой.

Система инструкций и гибкие адресные режимы DSP-ядра ELcore-14 позволяют эффективно реализовать алгоритмы сигнальной обработки. Время выполнения минимизируется за счет использования программного конвейера и высокопроизводительных инструкций, реализующих параллельно несколько вычислительных операций и пересылок.

DSP функционирует под управлением CPU и расширяет его возможности по обработке сигналов. Система команд DSP обеспечивает программирование всех базовых процедур сигнальной обработки. Краткая система инструкций DSP –ядра приведена в Приложении 1 к настоящему документу.

DSP имеет следующие основные технические характеристики:

- “Гарвардская” RISC – подобная архитектура с оригинальной системой команд и преимущественно однопоточным исполнением инструкций;
- **SISD (Single Instructions Single Data)** организация потоков команд и данных;
- Система инструкций обеспечивает одновременное выполнение в течение одного командного цикла до двух вычислительных операций и до двух пересылок;
- 3-ступенчатый конвейер по выполнению 32– и 64–разрядных инструкций;
- Расширенные возможности по динамическому диапазону обрабатываемых данных, позволяющие обрабатывать данные в 8/16/32–разрядных форматах с фиксированной точкой, либо в одном из форматов с плавающей точкой – 24E8 (стандарт IEEE 754) или 32E16 (расширенный формат). Обеспечение при этом компромиссного выбора между точностью и производительностью. Аппаратные меры повышения точности и динамического диапазона (блочная плавающая точка; режим насыщения; инструкции преобразования форматов);
- Аппаратная поддержка программных циклов;
- Память программ PRAM объемом 16 Кбайт (4К 32-разрядных слов);
- Общий объем памяти данных (включая X- и Y-области): 36К 32-разрядных слов. Двухпортовые памяти данных XRAM и YRAM объемом 96 и 48 Кбайт соответственно. Наличие двух портов у памяти программ и данных обеспечивает возможность подкачки и отвода данных без приостановки вычислений;

- Пиковая производительность DSP:
  - 240 млн. оп/с 32-битных операций с плавающей точкой (IEEE 754);
  - 1440 млн. оп/с 8-битных операций с фиксированной точкой;
  - 640 млн. оп/с 16-битных операций с фиксированной точкой;
  - 320 млн. оп/с 32-битных операций с фиксированной точкой.

Дополнительная информация о работе DSP содержится в документе: «DSP-ядро ELcore\_x4. Система инструкций».

## 3.2 Архитектура DSP

### 3.2.1 Структурная схема DSP

Структурная схема DSP приведена на Рисунок 3.1.

В состав DSP входят следующие блоки:

1. Операционные блоки:
  - ALU (Arithmetic & Logic Unit) – арифметико-логическое устройство;
  - AGU (Address Generator Unit) – устройство генерации адреса для X- и Y-памяти данных DSP;
  - AGU-Y – устройство генерации адреса для Y-памяти данных DSP;
2. Блоки программного управления:
  - PCU (Program Control Unit), содержащий:
  - PAG (Program Address Generator) - генератор адреса программ;
  - PDC (Program Decoder) - программный декодер.
3. Блоки коммутации:
  - IDBS (Internal Data Bus Switch) - внутренний коммутатор шин данных;
  - EDBS (External Data Bus Switch) - внешний коммутатор шин данных;
4. Блоки памяти:
  - PRAM - память программ DSP;
  - XRAM – X-память данных DSP;
  - YRAM – Y-память данных DSP;

Элементами архитектуры DSP также являются:

- внутренние шины адреса (XAB, YAB, PAB);
- внутренние шины данных (XDB, PDB, GDB, YDB);

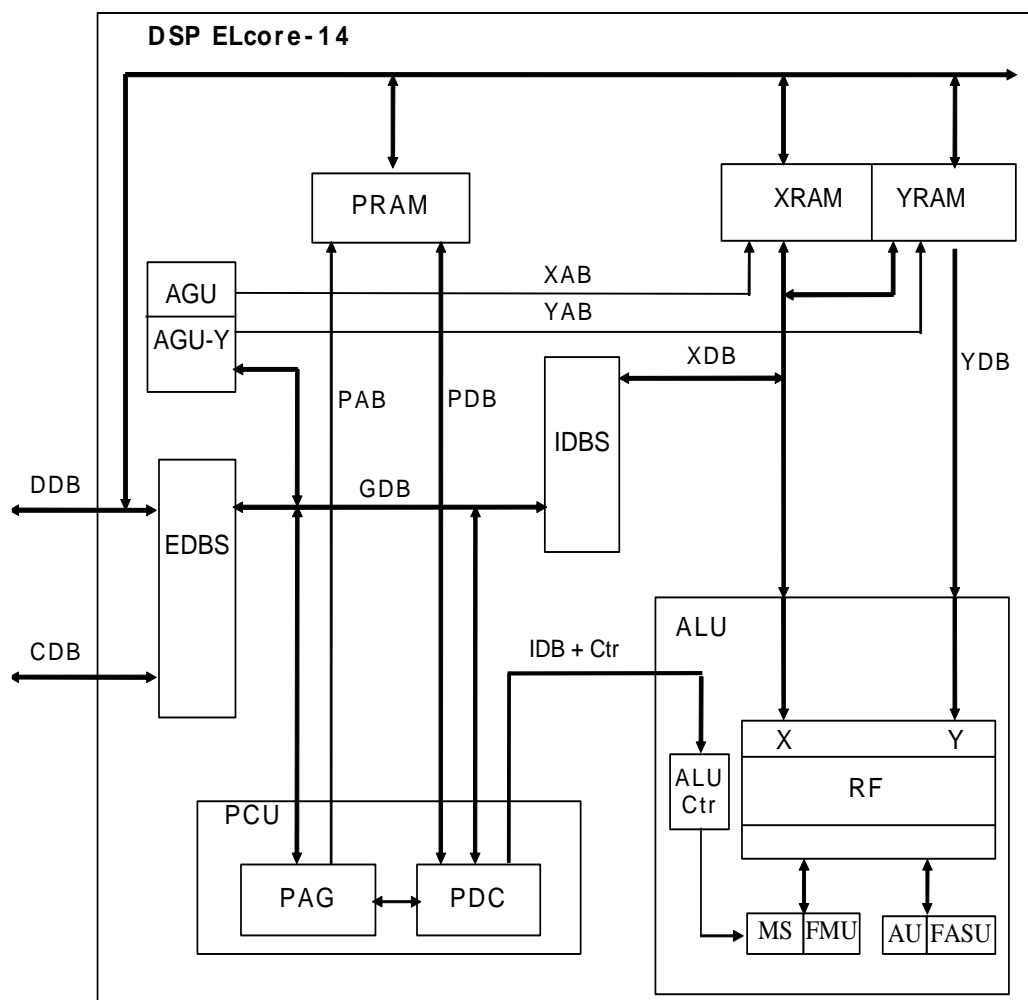


Рисунок 3.1 Структурная схема DSP ELcore-14.

### 3.2.2 Арифметико-логическое устройство (ALU)

Арифметико-логическое устройство (ALU) выполняет все вычислительные операции.

Арифметико-логическое устройство содержит в своем составе регистровый файл RF, регистры PDNR и CCR, регистры-аккумуляторы AC0 и AC1, а также вычислительные (операционные) устройства: умножитель/сдвигатель для форматов с фиксированной точкой (MS/SH); арифметическое устройство для форматов с фиксированной точкой (AU/LU), умножитель для форматов с плавающей точкой IEEE-754 (FMU); арифметическое устройство для форматов с плавающей точкой (FASU).

#### Регистровый файл

Регистровый файл (RF) представляет собой многопортовую оперативную память с организацией 32 слова по 16 бит или 16 слов по 32 бита. При помощи RF осуществляется параллельное чтение и запись нескольких операндов в соответствии с исполняемой операцией.

## Операционные блоки (MS/SH, FMU, AU/LU, FASU)

Операционные блоки выполняют следующие операции.

### Умножитель-сдвигатель для форматов с фиксированной точкой (MS/SH):

- операции умножения с целыми числами со знаком и без знака;
- операции умножения чисел со знаком в дробном формате с фиксированной точкой (fractional);
- операции многоразрядного арифметического и логического сдвига в форматах с фиксированной точкой;

### Умножитель для формата с плавающей точкой IEEE-754 (FMU):

- операции умножения чисел в формате с плавающей точкой IEEE-754;
- операции FIN (получение 8-разрядного приближения обратной величины);
- операции FINR (получение 8-разрядного приближения обратной величины квадратного корня).

### Арифметическое устройство для форматов с фиксированной точкой (AU), включая логическое устройство (LU) и узел битовой обработки (BFU):

- арифметические операции в форматах с фиксированной точкой;
- преобразования форматов чисел;
- ограничение результатов с целью устранения выхода за пределы разрядной сетки (Saturation).
- логические операции;
- операции с битовыми полями;

### Арифметическое устройство для формата с плавающей точкой (FASU):

- арифметические операции в форматах с плавающей точкой;
- преобразования форматов чисел.

## Регистры CCR, PDNR, AC0, AC1

Регистры CCR, PDNR являются 16-разрядными программно-доступными по записи и чтению регистрами, выполняющими следующие функции:

- регистр CCR предназначен для хранения признаков результата последней выполненной арифметической операции, а также для управления режимами округления (rounding) и насыщения (saturation);
- регистр PDNR предназначен для аппаратного измерения параметра денормализации массива данных и автоматического масштабирования результатов сложения/вычитания сдвигом вправо на 0/1/2 бита.

Регистры-аккумуляторы AC0, AC1 являются специализированными 32-разрядными регистрами данных, предназначенными для накопления результата в операциях умножения с накоплением. В операциях MAC, MACL регистры AC0, AC1 объединяются в один 64-разрядный регистр для получения 64-разрядного результата.

### 3.2.3 Устройства генерации адреса (AGU, AGU-Y)

Устройства AGU, AGU-Y выполняют вычисление адресов операндов в памяти данных XRAM, YRAM, используя целочисленную арифметику. При этом используется три



типа арифметики: линейная, модульная и арифметика с обратным переносом. Устройства генерации адресов функционируют параллельно с другими ресурсами DSP, что обеспечивает высокую производительность обработки данных.

### 3.2.4 Устройство программного управления (PCU)

DSP поддерживает набор типовых инструкций и режимов стандартного ЦПОС.

Выборка и декодирование инструкции осуществляется на базе трехступенчатого конвейера, что обеспечивает короткую (два командных цикла) скалярную задержку для вычислений.

Устройство программного управления (PCU) включает в себя два блока:

- Программный адресный генератор (PAG);
- Программный декодер (PDC).

Устройство PDC декодирует инструкции, поступающие из программной памяти, и генерирует сигналы управления программным конвейером.

Программный адресный генератор PAG выполняет вычисление адреса инструкции в программной памяти, организует выполнение программных циклов DO, управляет работой системного стека.

### 3.2.5 Коммутаторы шин данных (IDBS, EDBS)

Внутренний коммутатор шин данных IDBS предназначен для коммутации шин данных при выполнении пересылок и выполнения операции транспонирования матриц (см. в последующих разделах).

Внешний коммутатор шин данных EDBS предназначен для коммутации внешних системных шин на соответствующие внутренние шины при выполнении обменов с CPU и DMA.

### 3.2.6 Блоки памяти

Внутренняя память DSP включает в себя 4 независимых компоненты (пространства памяти):

- 1) память программ PRAM (пространство P);
- 2) память данных (включает область X-памяти и область Y-памяти);
- 3) регистры управления, включая регистры AGU, AGU-Y и PCU, а также регистры CCR, PDNR, AC0, AC1 (пространство C);
- 4) регистры данных - регистровый файл ALU (пространство R).

Внутренние модули памяти и внутренние регистры DSP (последние как устройства, расположенные в адресном пространстве) составляют подсистему памяти, т.е. устройства, доступные программно по адресным пространствам X, Y, P, C, R. Каждое из указанных устройств характеризуется следующими особенностями доступа:

- внутренние пространства памяти X, Y, P доступны только по одной (одноименной) шине, обращения одноклеточные, т.е. выполняются в течение одного командного цикла.

- регистры доступны по шине GDB, обращения одноклеточные.

При обращениях внутри DSP выбор конкретного устройства подсистемы памяти определяется адресом и пространством обращения. Для ускорения выбора устройства подсистемы памяти формироваатели адресов (AGU, AGU-Y, PAG) формируют также специальные признаки адресного пространства.

### ***Память программ и память данных***

Память программ PRAM имеет 64-разрядную организацию, позволяющую осуществлять хранение и выборку в течение одного такта как 32-разрядных, так и 64-разрядных инструкций. DSP ELcore\_14 имеет память PRAM объемом 4К 32-разрядных (или 2К 64-разрядных) слов.

Общее пространство памяти данных DSP состоит из двух областей: X- и Y-памяти (XRAM, YRAM), имеющих 32-разрядную организацию.

Память XRAM и память YRAM имеют следующий объем:

XRAM – 24К 32-разрядных слов;

YRAM - 12К 32-разрядных слов;

Модули памяти XRAM, YRAM, PRAM является двухпортовыми, что обеспечивает возможность одновременного доступа к ним как со стороны DSP, так и со стороны CPU или DMA.

### **3.2.7 Шины адреса и данных**

DSP-ядро имеет внешние шины адреса и данных DDB и CDB для обменов с CPU и DMA. Обмены CPU или DMA с памятью DSP происходят через отведенные для этого порты модулей памяти XRAM, YRAM и не прерывают работы DSP. В обменах по указанным шинам DSP является ведомым устройством (Slave) и не может самостоятельно инициировать обмен.

В пределах DSP передача данных и управляющей информации осуществляется при помощи внутренних шин:

- 32-разрядных шин данных памяти данных (XDB0, YDB0);
- 64-разрядной шины программных данных (PDB);
- 16-разрядной глобальной шины данных (GDB).

Внутренние модули памяти XRAM, YRAM и PRAM адресуются соответственно по однонаправленным адресным шинам: XAB, YAB и PAB.

Пересылки программ и выборки команд осуществляются по шине программных данных PDB. 16-разрядная шина GDB используется для обменов между регистрами DSP.

### 3.3 Арифметико-логическое устройство (ALU)

Арифметико-логическое устройство (ALU) является исполнительным устройством DSP, выполняющим все вычислительные операции с данными. В настоящем разделе описывается архитектура, программная модель и режимы работы ALU.

#### 3.3.1 Архитектура ALU

Арифметико-логическое устройство (Рисунок 3.2) содержит в своем составе следующие блоки:

- Регистровый файл (RF);
- Умножитель чисел в формате с плавающей точкой 24e8 (FMU);
- Параллельный умножитель и сдвигатель чисел в форматах с фиксированной точкой 8/16/32 (MS/SH);
- Сумматор, вычитатель и преобразователь чисел с плавающей точкой формата 24e8 (FASU);
- Арифметическое устройство (AU/LU), поддерживающее обработку 16/32-разрядных чисел в форматах с фиксированной точкой, включающий 16/32-разрядное логическое устройство, устройство преобразования битовых полей и устройство определения параметра денормализации;
- Два 32-разрядных регистра-аккумулятора (AC0, AC1);
- 16-разрядный регистр параметра денормализации (PDNR);
- 16-разрядный регистр кодов условий (CCR);
- Устройство управления ALU (ALU\_CTR).

Наличие в архитектуре ALU многопортового регистрового файла и нескольких операционных устройств (ОУ) делает возможным одновременное выполнение **до двух вычислительных операций и до двух операций пересылок**.

Операции, исполняемые блоками AU/LU/FASU, называются операциями типа **OP1**, операции, исполняемые блоками MS/SH/FMU, имеют тип **OP2**.

Все вычислительные операции и операции пересылок выполняются ALU за один такт (командный цикл). Новая команда может быть инициализирована на каждом такте.

Результат каждой арифметической операции может использоваться как исходный операнд для следующей операции.

Временная диаграмма взаимодействия RF с операционными устройствами (ОУ) ALU приведена на Рисунок 3.3.

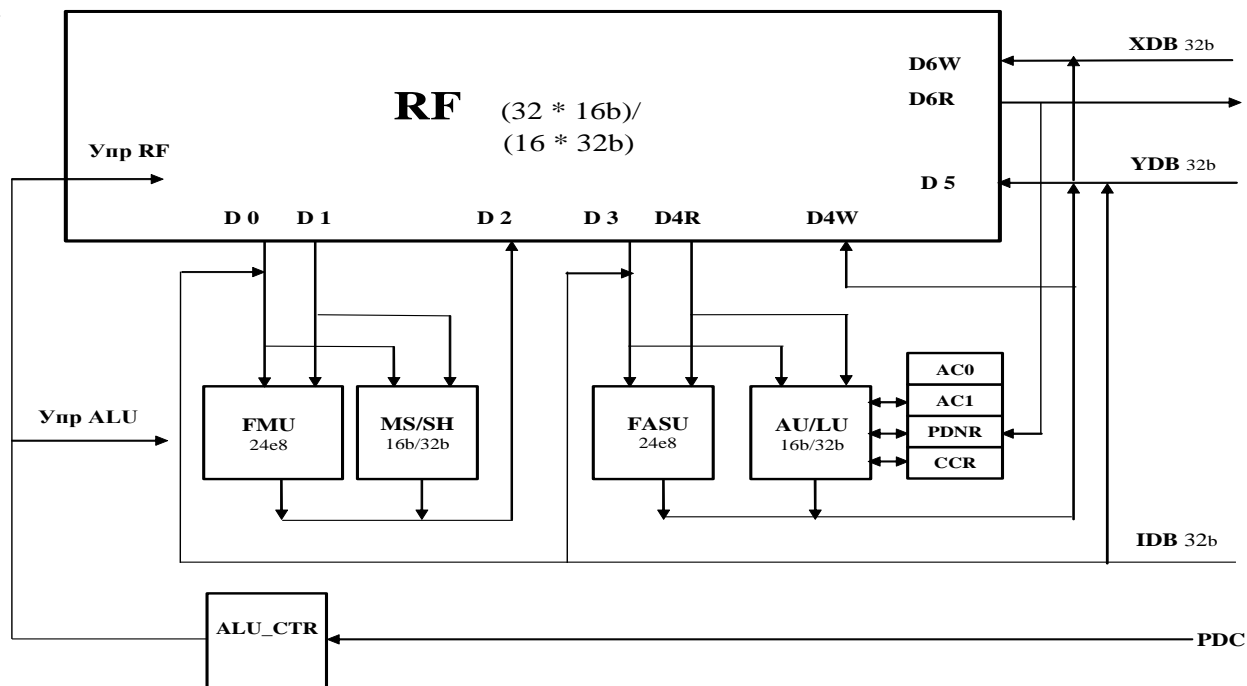


Рисунок 3.2. Структурная схема устройства ALU DSP-ядра ELcore-14

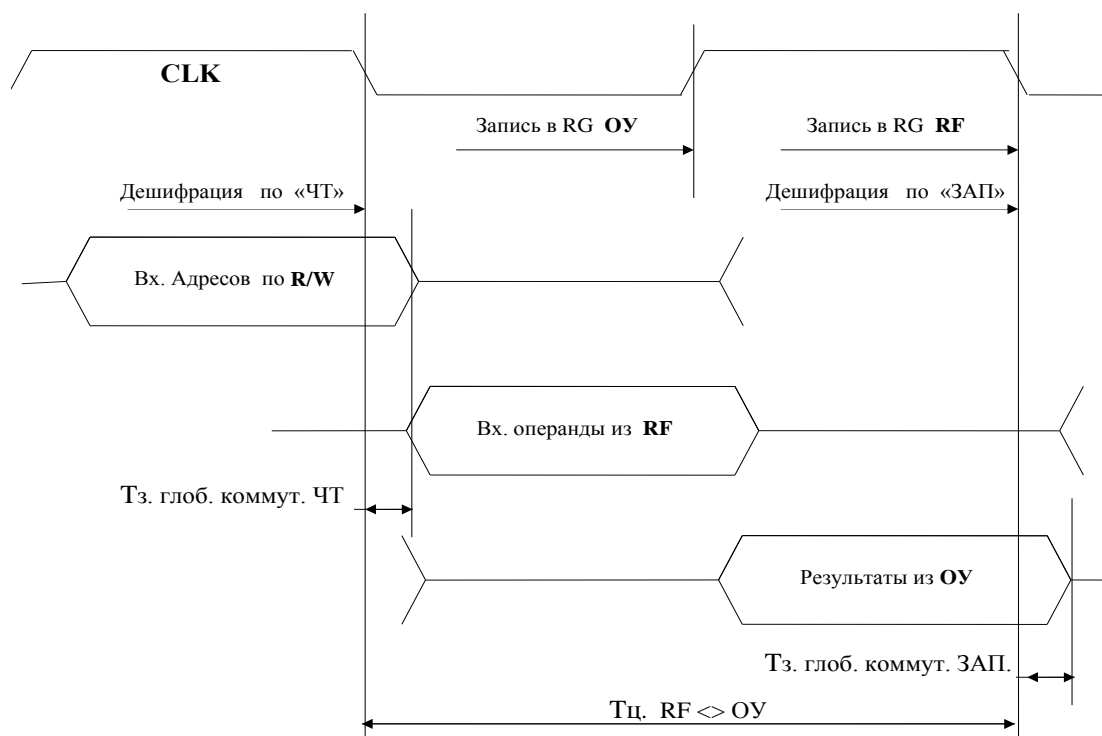


Рисунок 3.3. Временная диаграмма взаимодействия операционных устройств (ОУ) с регистровым файлом (RF)

### 3.3.1.1 Регистровый файл

Исходные операнды и результаты операций ALU хранятся в регистровом файле (RF), который представляет собой набор из 32-х программно-доступных 16-разрядных регистров R0–R31, которые могут конфигурироваться в 16 32-разрядных регистров.

Регистровый файл состоит из двух банков: нулевого банка с четными адресами регистров (R0, R2, ..., R30) и первого банка - с нечетными адресами (R1, R3, ..., R31). При 32-разрядных обращениях соответствующие регистры двух банков объединяются попарно и образуют 16 32-разрядных регистров, причем младшие 16 бит представлены в регистрах с четными номерами, старшие 16 бит - в регистрах с нечетными номерами (см. Таблица 3.1, Таблица 3.2).

Разрядность обращения определяется формируемым в командном слове признаком L: при L=0 происходит обращение к 16-разрядным операндам, при L=1 - к 32-разрядным. Признак L определяется разрядностью выполняемой операции. При 32-разрядных обращениях должен использоваться четный адрес регистра, соответствующий младшим 16 разрядам адресуемого операнда.

**Таблица 3.1 Программная модель RF при 16-разрядных обращениях**

L	Разрядность операнда	Адрес операнда	Старшие 16 бит операнда	Младшие 16 бит операнда
0	16	R0	-	R0[15: 0]
0	16	R1	-	R1[15: 0]
0	16	R2	-	R2[15: 0]
0	16	R3	-	R3[15: 0]
0	16	R4	-	R4[15: 0]
0	16	R5	-	R5[15: 0]
0	16	R6	-	R6[15: 0]
0	16	R7	-	R7[15: 0]
0	16	R8	-	R8[15: 0]
0	16	R9	-	R9[15: 0]
0	16	R10	-	R10[15: 0]
0	16	R11	-	R11[15: 0]
0	16	R12	-	R12[15: 0]
0	16	R13	-	R13[15: 0]
0	16	R14	-	R14[15: 0]
0	16	R15	-	R15[15: 0]
0	16	R16	-	R16[15: 0]
0	16	R17	-	R17[15: 0]
0	16	R18	-	R18[15: 0]
0	16	R19	-	R19[15: 0]
0	16	R20	-	R20[15: 0]
0	16	R21	-	R21[15: 0]
0	16	R22	-	R22[15: 0]
0	16	R23	-	R23[15: 0]
0	16	R24	-	R24[15: 0]
0	16	R25	-	R25[15: 0]
0	16	R26	-	R26[15: 0]
0	16	R27	-	R27[15: 0]
0	16	R28	-	R28[15: 0]

L	Разрядность операнда	Адрес операнда	Старшие 16 бит операнда	Младшие 16 бит операнда
0	16	R29	-	R29[15: 0]
0	16	R30	-	R30[15: 0]
0	16	R31	-	R31[15: 0]

**Таблица 3.2 Программная модель RF при 32-разрядных обращениях**

L	Разрядность операнда	Адрес операнда	Старшие 16 бит операнда	Младшие 16 бит операнда
1	32	R0	R1[15: 0]	R0[15: 0]
1	32	R2	R3[15: 0]	R2[15: 0]
1	32	R4	R5[15: 0]	R4[15: 0]
1	32	R6	R7[15: 0]	R6[15: 0]
1	32	R8	R9[15: 0]	R8[15: 0]
1	32	R10	R11[15: 0]	R10[15: 0]
1	32	R12	R13[15: 0]	R12[15: 0]
1	32	R14	R15[15: 0]	R14[15: 0]
1	32	R16	R17[15: 0]	R16[15: 0]
1	32	R18	R19[15: 0]	R18[15: 0]
1	32	R20	R21[15: 0]	R20[15: 0]
1	32	R22	R23[15: 0]	R22[15: 0]
1	32	R24	R25[15: 0]	R24[15: 0]
1	32	R26	R27[15: 0]	R26[15: 0]
1	32	R28	R29[15: 0]	R28[15: 0]
1	32	R30	R31[15: 0]	R30[15: 0]

Регистровый файл имеет 10 32-разрядных портов - 5 портов записи и 5 портов чтения. Это позволяет одновременно выполнять до трех арифметических операций и до двух пересылок данных.

Доступ к данным регистрового файла со стороны DSP-ядра может производиться по нескольким внутренним шинам:

- По 32-разрядной шине данных XDB для передачи данных из памяти XRAM;
- По 32-разрядной шине данных YDB для передачи данных из памяти YRAM,
- По 32-разрядной шине IDB для непосредственных операндов.

### 3.3.1.2 Операционные устройства

Операционные устройства (ОУ) выполняют следующие операции:

#### 1.1.1.1.1 Умножитель-сдвигатель для форматов с фиксированной точкой (MS/SH):

- операции умножения с целыми числами со знаком и без знака;

- операции умножения чисел со знаком в дробном формате с фиксированной точкой (fractional);

- операции многоразрядного арифметического и логического сдвига в форматах с фиксированной точкой;

#### **1.1.1.1.2 Умножитель для формата с плавающей точкой IEEE-754 (FMU):**

- операции умножения чисел в формате с плавающей точкой IEEE-754;
- операции FIN (получение 8-разрядного приближения обратной величины);
- операции FINR (получение 8-разрядного приближения обратной величины квадратного корня).

Арифметическое устройство для форматов с фиксированной точкой (AU), включая логическое устройство (LU) и узел битовой обработки (BFU):

- арифметические операции в форматах с фиксированной точкой;
- преобразования форматов чисел;
- ограничение результатов с целью устранения выхода за пределы разрядной сетки (Saturation).
- логические операции;
- операции с битовыми полями.

Арифметическое устройство для формата с плавающей точкой IEEE-754 (FASU):

- арифметические операции в форматах с плавающей точкой;
- преобразования форматов чисел.

### 3.3.1.3 Регистр PDNR

Назначение разрядов в регистре PDNR приведено ниже.

**PDNR:**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Esc	-	-	-	-	-	SC	Epdn	-	F	Cpdn					

где:

- **Cpdn** – текущий код PDN;
- **F (X/L)** – формат анализируемой информации в PDN (0 – 32 бит, 1 – 32 бит, комплексная);
- **Epdn** – программный признак разрешения детектирования и изменения PDN (Epdn: 0 – нет разрешения, 1 – разрешение);
- **SC** – величина масштабирования результата в AU;
- **Esc** – признак разрешения масштабирования результата в AU (0 – нет разрешения, 1 – разрешение).

Начальное состояние регистра PDNR = 0x0000.

### 3.3.1.4 Регистр CCR

Регистр CCR предназначен для хранения признаков результатов вычислительных операций. Регистр CCR содержит два поля признаков: основное {Ev,U,N,Z,V,C} (разряды [5:0]) и дополнительное {Evm,Um,Nm,Zm,Vm,Cm} (разряды [15:10]). Поле признаков в младшем байте регистра CCR является основным, т.к. на его основе формируются условия исполнения команд.



Поля признаков формируются по следующим правилам:

1) При выполнении одной операции типа OP1 (AU/LU/FASU) ее признаки помещаются только в основное поле.

2) При выполнении одной операции типа OP2 (MS/SH/FMU) ее признаки помещаются в оба поля.

3) При одновременном выполнении двух вычислительных операций признаки, формируемые операцией типа OP1, поступают в основное поле, признаки операции типа OP2 - в дополнительное поле.

4) В тех случаях, когда операция типа OP1 заполняет только часть признаков в основном поле, оставшиеся признаки формируются операцией OP2.

Регистр CCR содержит также специальные признаки E, t и два управляющих разряда RND и S. Назначение разрядов в регистре CCR приведено ниже.

**CCR:**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Evm	Um	Nm	Zm	Vm	Cm	RND	S	t	E	Ev	U	N	Z	V	C

где:

- **C** – признак переноса, сформированного в результате выполнения операции (0 – нет переноса, 1 – есть перенос);
- **V** – признак переполнения результата (0 – нет переполнения, 1 – есть переполнение);
- **Z** – признак нулевого результата (0 – результат не нулевой, 1 – результат нулевой);
- **N** – знак результата (0 – знак положительный, 1 – знак отрицательный);
- **U** – признак ненормализованного результата (0 – нормализованный результат, 1 – ненормализованный результат);
- **Ev** – запомненный ранее возникший признак переполнения результата (0 – не было переполнения, 1 – было переполнение);
- **E** – экспоненциальный признак (формируется командой CMPE);
- **t** – признак истинности условия после исполнения условной команды (t=0 – безусловная команда либо условие ложно; t=1 – условие истинно);
- **S** – бит включения режима насыщения результата (0 – отключение режима насыщения, 1 – включение режима насыщения);
- **RND** – бит управления режимом округления результата (0 – CR (Convergent Rounding), 1 – TCR (Two's-Complement Rounding));
- **Cm** – признак переноса сформированного в результате выполнения операции OP2 (0 – нет переноса, 1 – есть перенос);
- **Vm** – признак переполнения результата операции OP2 (0 – нет переполнения, 1 – есть переполнение);
- **Zm** – наличие нулевого результата операции OP2 (0 – результат не нулевой, 1 – результат нулевой);
- **Nm** – значение знака результата операции OP2 (0 – знак положительный, 1 – знак отрицательный);

- **Um** – признак ненормализованного результата операции OP2 (0 – нормализованный результат, 1 – ненормализованный результат);
- **Evm** – запомненный ранее возникший признак переполнения результата операции OP2 (0 – не было переполнения, 1 – было переполнение);

Начальное состояние регистра CCR = 0x0000.

### Стандартные определения признаков результата

Ниже приводятся стандартные правила формирования признаков результата вычислительной операции: **U**(unnormalized), **N**(negative), **Z**(zero), **V**(overflow), **C**(carry). Для отдельных операций некоторые признаки могут формироваться по иным специально оговоренным правилам. В дальнейшем при описании правил формирования признаков используются следующие обозначения: **msb** – номер старшего (знакового) разряда результата **D**, т.е.  $msb=31$  для 32-разрядных чисел и  $msb=15$  для 16-разрядных.

Кроме указанных основных признаков, при выполнении операций могут формироваться и некоторые дополнительные признаки, определение которых дается в описании регистра CCR.

Признак	Стандартные правила формирования признаков	
	Все вычислительные операции (кроме сдвига)	Операции сдвига: ASL, ASLL, ASLX, ASR, ASRL, ASRX, ASRLE, LSL, LSLX, LSR, LSRL, LSRX, ROL, ROLL, ROR, RORL
<b>U</b>	$U = 0$ , если $D[msb] \neq D[msb-1]$ ; $U = 1$ , если $D[msb] = D[msb-1]$ ;	
<b>N</b>	$N = D[msb]$ ;	
<b>Z</b>	$Z = 1$ , если $D=0$ ; $Z = 0$ , если $D \neq 0$ ;	
<b>V</b>	$V = 1$ , если $D[msb+1] \neq D[msb]$ ; $V = 0$ , если $D[msb+1] = D[msb]$ ;	Операции ASL, ASLL, ASLX: $V = 0$ , если хотя бы один разряд, выдвигаемый за пределы разрядной сетки или на место знака, не равен знаку; $V = 1$ , иначе;
<b>C</b>	$C = Cout[msb]$ , если режим Scaling выключен; $C = Cout[msb+1]$ , если режим Scaling включен.	$C$ принимает значения последнего из битов, выдвинутых за разрядную сетку результата $D[msb:0]$ вправо или влево, в зависимости от направления сдвига.

#### Пояснения.

Арифметическое устройство выполнено как полный 33-разрядный сумматор-вычитатель с дополнительным старшим разрядом под номером  $msb+1$ , используемым только для формирования признаков. На выход поступают 32 младших разряда результата  $D[msb:0]$ . Каждый из 33-х каскадов сумматора формирует, как соответствующий бит результата  $D[i]$ , так и перенос в следующий разряд  $Cout[i]$ .

### 3.3.1.5 Регистры-аккумуляторы AC0, AC1

Регистры-аккумуляторы AC0, AC1 являются специализированными 32-разрядными регистрами данных (по своим адресам регистры AC0, AC1 относятся к регистрам управления), предназначенными для накопления результата в операциях умножения с накоплением (MAC, MAC2, MACL, MACX, SAC2). В операциях MAC, MACL регистры AC0, AC1 объединяются в один 64-разрядный регистр для получения 64-разрядного результата.

Начальное состояние AC0 = AC1 = 0x00000000.

### 3.3.2 Режимы работы ALU

В ряде случаев результат выполнения арифметической операции зависит не только от самой этой операции и исходных операндов, но и от установленного режима вычислений (способа формирования результата). К числу таких режимов относятся:

- Режимы (способы) округления (Rounding);
- Режим масштабирования (Scaling);
- Режим насыщения (Saturation);
- Режим отслеживания блочной экспоненты (Block Floating Point Support);

#### 3.3.2.1 Округление (Rounding)

Округление (Rounding) может выполняться как самостоятельная операция (RNDL), либо в составе более сложных операций для преобразования 32-разрядного формата данных в 16-разрядный.

Перечень операций, в которых используется округление, приведен ниже.

<i>Long</i>	<i>Short</i>	<i>Complex</i>
<b>Блок AU</b>		
RNDL		
ADDLR		
SUBLR		
ADDLRTR		
SUBLRTR		
FTRL		

Округление может выполняться одним из двух способов: округление к ближайшему (convergent rounding) и округление дополнительного кода (two's-complement rounding). Способ (режим) округления устанавливается 9-м разрядом (бит RND) регистра CCR.

### 3.3.2.1.1 Режим округления к ближайшему (Convergent Rounding)

Округление к ближайшему (также называется “к ближайшему четному числу”) – способ округления по умолчанию.

Традиционный метод округления округляет вверх при большем значении числа, чем половина, и округляет вниз для любого значения меньше, чем половина. Вопрос возникает только относительно того, как эта половина должна быть округлена. Если это всегда будет округляться одним способом, то результаты в конечном счете будут смещены в том же направлении.

*Округление к ближайшему решает эту проблему так:*

- Округление осуществляется в меньшую сторону, если число четное (младший бит равен нулю).
- Округление выполняется в большую сторону, если число нечетно (младший бит равен единице).

В результате алгоритм округления 32-разрядного числа  $R[31:0]$  к 16-ти разрядам  $R[31:16]$  описывается следующим логическим выражением:

$$r = (\sim R[15] | (\sim R[16] \& R[15] \& (\sim (|R[14:0]))) ) ? 0'b:1'b;$$

где:  $r$  - единица округления.

Результат округления:  $d[15:0] = R[31:16] + r$ .

### 3.3.2.1.2 Режим округления дополнительного кода (Two's-Complement Rounding)

Все значения, большие или равные половине, округляются вверх, а все меньшие, чем половина, округлены в меньшую сторону.

В результате алгоритм округления описывается следующим логическим выражением:

$$r = (\sim R[15]) ? 0'b:1'b;$$

где:  $r$  - единица округления.

Результат округления:  $d[15:0] = R[31:16] + r$ .

### 3.3.2.2 Масштабирование (Scaling)

Масштабирование позволяет избежать переполнения при выполнении арифметических операций путем сдвига вправо полученного результата.

Этот режим может быть полезен, в частности, при реализации алгоритма БПФ с прореживанием по частоте (Decimation-In-Frequency), когда при выполнении операций сложения/вычитания над комплексными числами необходимо избежать переполнения на выходе сумматора.

Масштабирование выполняется путем арифметического сдвига результата операции вправо на 0/1/2 бита, при этом величина сдвига определяется полем SC (разряды 9, 8) регистра PDNR.

Включение режима масштабирования осуществляется установкой в «1» бита 15 (Esc) регистра PDNR. Другой способ включения этого режима состоит в установке в «1» поля M непосредственно в командном слове (формат 8). Синтаксически это выражается в добавлении к мнемоническому имени команды суффикса “s”, например, ADDLs, SUBXs и т.п. Более подробная информация об этом содержится в документе: «DSP-ядро ELcore\_x4. Система инструкций».

Перечень операций, в которых может быть использован режим масштабирования, приведен ниже.

<i>Long</i>	<i>Short</i>	<i>Complex</i>
<b>Блок AU</b>		
ABSL	ABS	
NEGL	NEG	
ADDL	ADD	ADDX
SUBL	SUB	SUBX
ADCL	ADC	
ADC16L	AD1	
SBCL	SBC	
ADDSUBL	ADDSUB	ADDSUBX
RNDL		
ADDLR	ASH	
SUBLR	SAH	
ADDLRTR		
SUBLRTR		
FTRL		

### 3.3.2.3 Поддержка режима блочной экспоненты

Данный режим обеспечивает определение блочного порядка для массива данных в формате с фиксированной точкой, в частности, при выполнении алгоритма БПФ и заключается в аппаратном измерении так называемого *параметра денормализации* (PDN) массива.

Число D в формате с фиксированной точкой считается нормализованным, если у него знаковый и следующий за ним разряд не совпадают, т.е.

$$D[\text{msb}] \neq D[\text{msb}-1],$$

где msb – номер знакового разряда числа D: msb=31 для 32-разрядных чисел и msb=15 для 16-разрядных.

Параметр денормализации числа D определяется формулой:

$$\text{PDN} = \text{msb} - n - 1,$$

где n – номер старшего «значащего» разряда числа D, т.е. старшего из разрядов, не равных знаковому.

Для комплексных чисел PDN определяется как наименьшее из значений параметра денормализации отдельно для действительной и мнимой частей.

Для определения параметра денормализации *отдельных чисел*, представленных в различных форматах, в системе инструкций DSP-ядра ELcore\_x4 имеются специальные операции: ***PDN, PDNX, PDNL***.

Для определения параметра денормализации *массивов данных*, пересылаемых между регистровым файлом и памятью данных XRAM, предусмотрен *режим автоматического отслеживания блочной экспоненты*.

При этом под параметром денормализации массива понимается наименьшее значение PDN входящих в него чисел.

Режим автоматического отслеживания блочной экспоненты включается посредством установки в «1» бита 7 (Epdn) регистра PDNR, при этом 5-й бит регистра определяет тип анализируемых данных.

Результат измерения PDN помещается в поле Crdn регистра PDNR.

### 3.3.2.4 Режим насыщения (Saturation)

Устройство ALU имеет режим работы с насыщением (Saturation), в котором производится ограничение результата сверху и снизу рамками разрешенного диапазона значений. Включение этого режима происходит под управлением 8-го бита (бит S) регистра CCR. Ниже приводится перечень операций, в которых может быть использован режим насыщения.

<i>Long</i>	<i>Short</i>	<i>Complex</i>
<b>Блок MS</b>		
	MPF	
	MPF2	
	MPF2S	
		MPX
ASLL	ASL	ASLX
<b>Блок AU</b>		
ABSL	ABS	
NEGL	NEG	
ADDL	ADD	ADDX
SUBL	SUB	SUBX
ADCL	ADC	
ADC16L	AD1	
SBCL	SBC	
ADDSUBL	ADDSUB	ADDSUBX
RNDL		
ADDLR	ASH	
SUBLR	SAH	
ADDLRTR		
SUBLRTR		
FTRL		

Обработка режима насыщения.

Результаты операций в форматах с фиксированной точкой, имеющие знак, представлены *в дополнительном коде*. Включение режима насыщения подразумевает присвоение результату операции граничного значения в случае выхода результата за пределы разрешенного диапазона.

Ниже в таблице приводятся граничные значения для указанных типов чисел.

При выполнении насыщения знак результата сохраняется. Вырабатываются признаки переполнения - V, Ev.

Среди операций, использующих режим насыщения, имеются такие, при которых формируются более одного результата. Это парные операции ADDSUB, ADDSUBL,

ASH, SAH и операции с комплексными числами – ADDX, SUBX, ADDSUBX, MPX, ASLX.

Насыщение для указанных операций выполняется по каждой компоненте независимо, с использованием компонентных признаков переполнения.

Граничные значения		Форматы		
		16 разрядов	32 разряда	64 разряда
Наименьшее значение	16-ричное представление	0x8000	0x80000000	0x8000000000000000
	дробное	-1.0	-1.0	-1.0
	целое	$-2^{15}$	$-2^{31}$	$-2^{63}$
Наибольшее значение	16-ричное представление	0x7FFF	0x7FFFFFFF	0x7FFFFFFFFFFFFFFF
	дробное	$2^{-15}$	$1 - 2^{-31}$	$1 - 2^{-63}$
	целое	$2^{15} - 1$	$2^{31} - 1$	$2^{63} - 1$

### 3.4 Устройства генерации адресов памяти данных (AGU, AGU-Y)

Общее пространство памяти данных DSP-ядра состоит из двух областей: X- и Y-памяти. Генерация адресов для памяти данных при внутренних обменах DSP осуществляется адресными генераторами - AGU и AGU-Y.

Устройства AGU, AGU-Y производят вычисление адресов, используя целочисленную 16-разрядную арифметику. При этом используется три типа арифметики: линейная, модульная и арифметика с обратным переносом. Устройства генерации адресов функционируют параллельно с другими ресурсами DSP, что обеспечивает высокую производительность обработки данных.

#### 3.4.1 Архитектура AGU

Адресный генератор AGU формирует адрес XAB, обслуживающий память данных XRAM, а также, при определенных условиях, адрес YAB для памяти данных YRAM.

Блок-схема адресного генератора AGU приведена на Рисунке 3.4.

AGU содержит восемь наборов из трех регистров (триплетов), в число которых входят: регистр адреса  $A_n$ , регистр смещения  $I_n$  и регистр модификатора  $M_n$  ( $n=0,1,\dots,7$ ).

AGU может модифицировать один адресный регистр из своего набора регистров в течение одного командного цикла. При этом содержание соответствующего регистра модификатора определяет тип используемой арифметики.

Входящее в состав адресного генератора арифметическое устройство АУ содержит три сумматора.

Первый 16-разрядный полный сумматор, называемый сумматором смещения, выполняет следующие операции модификации адреса:

- увеличение на 1;
- уменьшение на 1;
- увеличение на величину смещения  $I_n$ ;

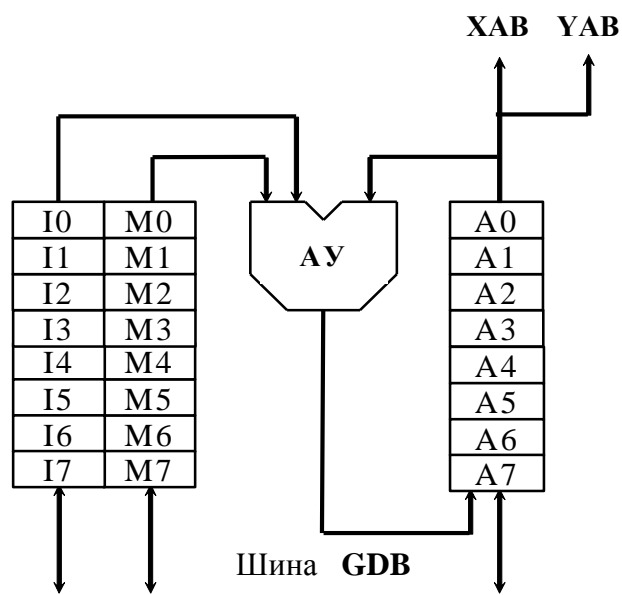
- уменьшение на величину смещения  $I_n$ ;

Второй полный сумматор, называемый модульным сумматором, добавляет (или вычитает) к результату первого сумматора величину модуля, которая хранится в соответствующем регистре модификатора  $M_n$ .

Третий полный сумматор, называемый сумматором обратного переноса, выполняет следующие операции модификации адреса с обратным направлением распространения переноса (от старших разрядов к младшим):

- увеличение на 1;
- уменьшение на 1;
- увеличение на величину смещения  $I_n$ ;
- уменьшение на величину смещения  $I_n$ ;

Сумматор смещения работает параллельно с сумматором обратного переноса и имеет с ним общие входы. Единственная разница между ними состоит в направлении распространения переноса. Управляющая логика определяет, результат которого из трех сумматоров является выходом адресного генератора.



**Рисунок 3.4** Блок-схема адресного генератора AGU

В состав AGU входят регистры адреса  $A_0$ - $A_7$ , регистры смещения  $I_0$ - $I_7$  и регистры модификатора  $M_0$ - $M_7$ . Регистры  $A_n$ ,  $I_n$ ,  $M_n$ , где  $n=0, \dots, 7$ , составляют триплет. Это означает, что при модификации адресного регистра  $A_n$  могут быть использованы только регистры, имеющие тот же индекс –  $I_n$ ,  $M_n$ .

Восемь регистровых триплетов адресного генератора:

- $A_0:I_0:M_0$
- $A_1:I_1:M_1$
- $A_2:I_2:M_2$
- $A_3:I_3:M_3$
- $A_4:I_4:M_4$
- $A_5:I_5:M_5$
- $A_6:I_6:M_6$



- A7:I7:M7

Запись или чтение каждого из указанных регистров осуществляются через глобальную шину данных (GDB) DSP.

### 3.4.2 Программная модель AGU

С точки зрения программиста, адресный генератор AGU представляет собой восемь наборов по три регистра, как показано на Рисунок 3.5. Эти регистры могут использоваться для хранения адресных указателей или других данных. При косвенной адресации операндов в памяти автоматически включается механизм обновления адресных указателей. Адресные регистры могут быть запрограммированы для линейной адресации, модульной адресации или реверсивной адресации.



Рисунок 3.5. Программная модель AGU.

#### 3.4.2.1 Адресный регистровый файл

Восемь 16-разрядных адресных регистров A0-A7 могут содержать адреса, либо произвольные данные. Содержимое адресного регистра может непосредственно указывать на данные в памяти либо используется для формирования указателя со смещением.

Адресный регистр обновляется после формирования адресного указателя (пост-модификация).

#### 3.4.2.2 Регистровый файл смещений

Восемь 16-разрядных регистров смещений I0-I7 могут содержать значения смещений, используемых для инкрементации или декрементации адресных регистров при выполнении обновления адреса. Эти регистры могут также использоваться для хранения произвольных данных.

#### 3.4.2.3 Регистровый файл модификаторов

Восемь 16-разрядных регистров модификаторов M0-M7 определяют тип адресной арифметики, применяемой при модификации адреса.

Адресные АЛУ поддерживают три типа арифметики: *линейную, модульную и арифметику с обратным переносом*. Для модульной арифметики содержимое регистров модификаторов определяет также модуль.

### 3.4.3 Архитектура AGU-Y

Адресный генератор AGU-Y формирует адрес YAB для памяти данных YRAM.

В каждой секции DSP имеется отдельное устройство AGU-Y для генерации адресов сегмента памяти YRAM соответствующей секции.

Блок-схема адресного генератора AGU-Y приведена на Рисунок 3.6.

AGU-Y содержит набор регистров, в число которых входят: регистры адреса AT, регистры смещения IT и DT регистр и модификатора MT.

AGU-Y может модифицировать адресный регистр AT в течение одного командного цикла. При этом содержание соответствующего регистра модификатора MT определяет тип используемой арифметики.

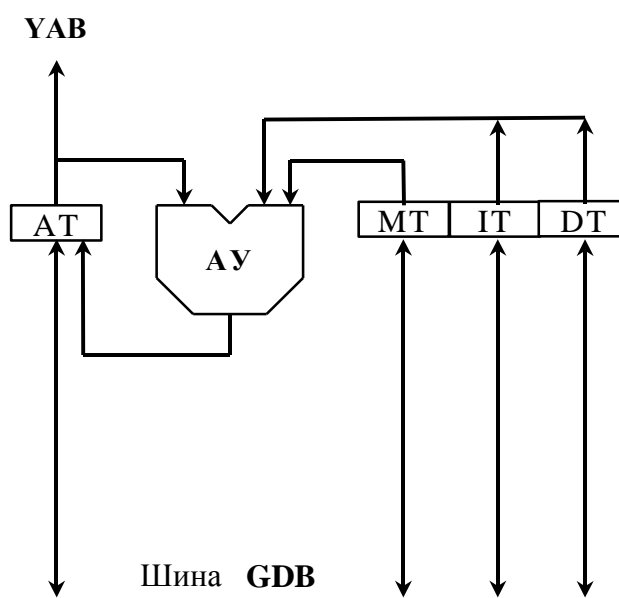


Рисунок 3.6 Блок-схема адресного генератора AGU-Y

Адрес, генерируемый AGU-Y, подается на адресную шину YAB.

Входящее в состав адресного генератора арифметическое устройство AU содержит три сумматора.

Первый 16-разрядный полный сумматор, называемый сумматором смещения, выполняет следующие операции модификации адреса:

- увеличение на величину смещения IT;
- увеличение на величину смещения DT;

Второй полный сумматор, называемый модульным сумматором, добавляет (или вычитает) к результату первого сумматора величину модуля, которая хранится в регистре модификатора MT.

Третий полный сумматор, называемый сумматором обратного переноса, может выполнять следующие операции модификации адреса с обратным направлением распространения переноса – от старших разрядов к младшим:

- увеличение на величину смещения IT;
- увеличение на величину смещения DT;

Сумматор смещения работает параллельно с сумматором обратного переноса и имеет с ним общие входы. Единственная разница между ними состоит в направлении распространения переноса. Управляющая логика определяет один из трех сумматоров, результат которого является выходом адресного генератора.

В состав AGU-Y входят регистр адреса AT, регистры смещения IT, DT и регистр модификатора MT.

Запись или чтение каждого из указанных регистров осуществляется через глобальную шину данных (GDB) DSP.

### 3.4.4 Программная модель AGU-Y

С точки зрения программиста, адресный генератор представляет собой восемь наборов по три регистра (AALU1) и набор из четырех регистров (AALU2), как показано на Рисунок 3.7. Регистр MT может быть запрограммирован для линейной адресации, модульной адресации или реверсивной адресации.



**Рисунок 3.7 Программная модель AGU-Y.**

### 3.4.5 Виды адресации

Применяются следующие виды (способы) адресации: прямая адресация (для регистров управления и данных), косвенная адресация (для памяти данных и программ), абсолютная адресация и адресация относительно программного счетчика (для программной памяти).

*Прямая адресация* используется при пересылках данных между регистрами данных или управления DSP-ядра.

*Косвенная адресация* используется при обменах с памятью данных.

*Абсолютная адресация программной памяти и адресация программной памяти относительно программного счетчика* используется при организации программных переходов и циклов.

Рассматриваемые в настоящем разделе адресные генераторы AGU и AGU-Y обеспечивают *косвенную адресацию памяти данных*.

Другие виды адресации обеспечиваются блоками, входящими в состав устройства программного управления PCU, рассматриваемого в следующем разделе:

- Прямая адресация регистров выполняется программным декодером PDC.

- Все виды адресации программной памяти обеспечиваются программным адресным генератором PAG.

Перечень используемых видов адресации приведен в таблице 3.3.

### 3.4.5.1 Прямая регистровая адресация

Прямая регистровая адресация определяет, что операндом является один или более регистров данных или управления (включая регистры адресного генератора).

Операндом может быть один, два или три регистра, как это определяется соответствующей командой. Используемая при этом в команде ссылка называется регистровой ссылкой.

Пример: MOVE R7,CCR

R7 – регистровая ссылка на регистр данных R7 (ссылка типа R);

CCR – регистровая ссылка на регистр управления CCR (ссылка типа C).

**Таблица 3.3 Виды адресации**

Виды адресации	Использование регистров AGU			Тип ссылки					Ассемблерный синтаксис	
	An (AT)	In (IT, DT)	Mn (MT)	C	R	P	X	Y		
Прямая регистровая адресация										
Регистр данных или управления	-	-	-	√	√					<имя регистра>
Косвенная регистровая адресация										
Отсутствие модификации адреса (XRAM)	+	-	-				√			(An)
Отсутствие модификации адреса (YRAM)	+	-	-					√		(AT)
Пост – инкремент на 1	+	-	+				√			(An) +
Пост – инкремент на In	+	+	+				√			(An) + In
Пост – инкремент на IT	+	+	+					√		(AT) + IT
Пост – инкремент на DT	+	+	+					√		(AT) + DT
Пост – декремент на 1	+	-	+				√			(An) -
Пост – декремент на In	+	+	+				√			(An) - In
Адресация со смещением на In (XRAM)	+	+	+				√			(An + In)
Адресация со смещением на IT (YRAM)	+	+	+					√		(AT + IT)
Непосредственное смещение	+	-	+				√			(displ)
Абсолютная адресация программной памяти										
Абсолютная прямая адресация	-	-	-				√			#I16
Абсолютная косвенная адресация	+	-	-				√			(An)
Адресация программной памяти относительно программного счетчика (PC)										
Относительная прямая адресация	-	-	-				√			PC + #I16
Относительная косвенная адресация	+	-	-				√			PC + An
Обозначения:										
C – ссылка на регистр управления RC;										
R – ссылка на регистр данных R;										
P – ссылка на память программ PRAM;										
X – ссылка на память данных XRAM;										
Y – ссылка на память данных YRAM;										

### 3.4.5.2 Виды адресации программной памяти

При формировании адреса программной памяти может использоваться абсолютная и относительная, прямая и косвенная адресация.

Абсолютная адресация программной памяти применяется в операциях программных переходов и циклов, использующих абсолютный адрес перехода – J, JD, JS, DO, DO\_R. Относительная адресация памяти программ применяется в операциях переходов и циклов, формирующих адрес перехода относительно программного счетчика PC – B, BD, BS, DOR, DOR\_R. И абсолютная, и относительная адресация может быть либо прямой, когда адрес перехода задается непосредственным операндом, либо косвенной когда адрес перехода содержится в адресном регистре.

### 3.4.5.3 Косвенная адресация памяти данных

При косвенной адресации для указания на ячейку памяти (XRAM или YRAM) используется адресный регистр An, а в общем случае – группа регистров An, In, Mn, позволяющих по определенным правилам вычислить значение указателя. Используемые режимы генерации адреса приводятся ниже.

#### 3.4.5.3.1 Отсутствие модификации адреса (An)

Адрес операнда содержится в адресном регистре. При выполнении команды значение адреса не изменяется.

#### 3.4.5.3.2 Пост – инкремент на 1

Адрес операнда содержится в адресном регистре An. После использования адреса его значение увеличивается на 1 и сохраняется в том же адресном регистре. Тип используемой арифметики определяется соответствующим регистром модификатора. Регистр смещения не используется.

#### 3.4.5.3.3 Пост – инкремент на In

Адрес операнда содержится в адресном регистре An. После использования адреса его значение увеличивается на величину смещения, содержащуюся в регистре In, и сохраняется в том же адресном регистре An. Тип используемой арифметики определяется соответствующим регистром модификатора Mn. Содержимое регистра смещения не изменяется.

#### 3.4.5.3.4 Пост – декремент на 1

Адрес операнда содержится в адресном регистре An. После использования адреса его значение уменьшается на 1 и сохраняется в том же адресном регистре. Тип используемой арифметики определяется соответствующим регистром модификатора. Регистр смещения не используется.

#### 3.4.5.3.5 Пост – декремент на In

Адрес операнда содержится в адресном регистре  $A_n$ . После использования адреса его значение уменьшается на величину смещения, содержащуюся в регистре  $I_n$ , и сохраняется в том же адресном регистре  $A_n$ . Тип используемой арифметики определяется соответствующим регистром модификатора  $M_n$ . Содержимое регистра смещения не изменяется.

### 3.4.5.3.6 Адресация со смещением на $I_n$

Адресом операнда является сумма значений, хранящихся в адресном регистре  $A_n$  и в регистре смещения  $I_n$ . Тип используемой арифметики определяется соответствующим регистром модификатора  $M_n$ . Содержимое регистра адреса  $R_n$  и регистра смещения  $I_n$  остается неизменным.

### 3.4.5.3.7 Непосредственное смещение ( $A_n + displ$ )

Адресом операнда является сумма значений, хранящихся в адресном регистре  $A_n$  и непосредственного смещения, содержащегося в поле команды. Тип используемой арифметики определяется соответствующим регистром модификатора  $M_n$ . Содержимое регистра адреса  $A_n$  остается неизменным. Регистр смещения  $I_n$  не используется.

## 3.4.6 Типы адресной арифметики

Адресный генератор поддерживает четыре типа адресной арифметики:

- линейная,
- модульная,
- модульная с кратным обращением,
- арифметика с обратным переносом.

Предоставляемые возможности достаточны для организации в памяти структур данных типа очередей (FIFO), линий задержки, циклических буферов, стеков, буферов с обратным порядком адресации для реализации БПФ.

Работа с данными при этом сводится в большей степени к манипуляциям с адресами, чем к пересылкам больших блоков данных.

Тип используемой адресной арифметики определяется значением, хранящимся в регистре модификатора. Для модульной арифметики содержимое регистров модификаторов определяет также модуль. Каждый адресный регистр имеет один связанный с ним регистр модификатора.

Значения модификатора  $M_n$  и соответствующие им типы адресной арифметики указаны в Таблица 3.4.

**Таблица 3.4 Типы адресной арифметики**

Модификатор $M_n$	Адресная арифметика
\$0000	Арифметика с обратным переносом
\$0001	Модуль 2
\$0002	Модуль 3
...	...
\$7FFE	Модуль 32767 ( $2^{15} - 1$ )
\$7FFF	Модуль 32768 ( $2^{15}$ )

\$8001	Модуль 2 с кратным обращением
\$8003	Модуль 4 с кратным обращением
\$8007	Модуль 8 с кратным обращением
...	...
\$9FFF	Модуль $2^{13}$ с кратным обращением
\$BFFF	Модуль $2^{14}$ с кратным обращением
\$FFFF	Линейная арифметика (Модуль $2^{16}$ )
Остальные комбинации – резерв	

### 3.4.6.1.1 Линейная адресная арифметика ( $Mn = \$FFFF$ )

Модификация адреса выполняется с использованием обычной 16-разрядной линейной (по модулю 65536) арифметики. 16-разрядное смещение,  $In$ ,  $+1$  или  $-1$  могут использоваться для вычисления адреса. Диапазон значений может рассматриваться как знаковый (от  $-32768$  до  $+32767$ ) либо как беззнаковый (от 0 до 65535), так как адресное ALU работает в обоих случаях одинаково.

### 3.4.6.1.2 Адресная арифметика с обратным переносом ( $Mn = \$0000$ )

Этот вариант адресной арифметики выбирается посредством установки регистра модификатора в 0. Модификация адреса в этом случае выполняется аппаратно с распространением переноса в обратном направлении – от старших разрядов к младшим.

Операция модификации адреса с обратным переносом эквивалентна последовательному выполнению следующих процедур:

- Изменению на обратный порядок следования разрядов в регистрах адреса и смещения (при этом старший бит становится младшим и т.д.);
- Модификации адреса посредством нормальной операции сложения;
- Возвращению первоначального порядка следования разрядов адреса.
- В случае, когда величина смещения составляет  $2^{(k-1)}$  (целая степень двойки), такая модификация адреса эквивалентна:
- Обращению порядка следования  $k$  младших разрядов  $An$ ;
- Увеличению на 1;
- Возвращению исходного порядка следования  $k$  младших разрядов  $An$ .

Рассматриваемый режим адресной арифметики удобен при реализации алгоритма быстрого преобразования Фурье (БПФ).

### 3.4.6.1.3 Модульная адресная арифметика ( $Mn = \text{Modulus} - 1$ )

Модификация адреса выполняется по модулю  $M$ , где  $M$  - целое число в пределах от 2 до 32768. Арифметика по модулю  $M$  вынуждает значение адреса оставаться в пределах диапазона значений, отличающихся друг от друга не более чем на  $M-1$ .

Величина  $M-1$  хранится в регистре модификатора адреса. Нижняя граница диапазона (базовый адрес) должна иметь нули в младших  $k$  разрядах, где  $2^k \geq M$ . Верхняя граница диапазона определяется как сумма нижней границы и модуля минус единица (базовый адрес  $+ M - 1$ ).

$$base\_addr = \{An[15:k], \{k\{0\}\}\};$$

$$base\_addr \leq XAB \leq base\_addr + M - 1 ;$$

Нижняя и верхняя границы диапазона определяются значением  $A_n$ . При этом необязательно устанавливать  $A_n$  равным базовому адресу. Достаточно того, чтобы величина  $A_n$  находилась в пределах требуемого диапазона.

Если при вычислении адреса в этом режиме используется смещение  $In$ , его величина не должна превышать  $M$ . Выходной адрес  $XAB$  для этого случая определяется формулой:

$$XAB = base\_addr + (A_n[k-1:0] \pm In)_{\text{mod}M} ;$$

Рассматриваемый тип адресной арифметики удобен при организации циклических буферов для реализации на их основе структур данных типа очередей (FIFO), линий задержки и т.п.

#### 3.4.6.1.4 Кратная модификация адреса по модулю

Этот тип адресной арифметики выбирается посредством установки в «1» 15-го разряда регистра модификатора  $M_n$ , как это показано в Табл.3.4.

Модификация адреса выполняется по модулю  $M$ , где  $M$  - степень двойки в пределах от  $2^1$  до  $2^{14}$ . Арифметика по модулю  $M$  вынуждает значение адреса оставаться в пределах диапазона значений, отличающихся друг от друга не более чем на  $M-1$ .

Величина  $M-1$  хранится в младших 15-ти разрядах регистра модификатора адреса  $M_n$ . Нижняя граница диапазона (базовый адрес) должна иметь нули в младших  $k$  разрядах, где  $2^k \geq M$ . Верхняя граница диапазона определяется как сумма нижней границы и модуля минус единица (базовый адрес +  $M - 1$ ).

Выходной адрес  $XAB$  и границы диапазона определяются по тем же формулам, что и при обычной модульной арифметике:

$$XAB = base\_addr + (A_n[k-1:0] \pm In)_{\text{mod}M} ;$$

$$base\_addr = \{A_n[15:k], \{k\{0\}\}\};$$

$$base\_addr \leq XAB \leq base\_addr + M - 1 ;$$

Отличие состоит в том, что для данного типа адресной арифметики величина смещения  $In$  может быть произвольной.

### 3.4.7 Режимы адресации

#### 3.4.7.1 Режимы адресации AGU

Режимы адресации приведены в Таблица 3.5.

**Таблица 3.5 Режимы адресации памяти данных**

Номер режима адресации	Режим адресации	Описание
0	-	Отмена пересылки
1	$(A_n)$	Косвенная
2	$(A_n)+$	Пост - автоинкремент
3	$(A_n)-$	Пост – автодекремент
4	$(A_n)+In$	Пост - автоувеличение
5	$(A_n)-In$	Пост – автоуменьшение



Номер режима адресации	Режим адресации	Описание
6	(An+In)	Индексирование (An не меняется)
7	(An+dspl)	С непосредственным смещением (A не меняется)

Примечание. По установленному признаку “u” в командном слове вычисляется исполнительный адрес без выполнения самой пересылки.

### 3.4.7.2 Режимы адресации AGU-Y

Режимы адресации AGU-Y сведены в Таблица 3.6. Режим адресации определяется полем “AT” командного слова и управляющим параметром YM (11-й разряд регистра SR).

**Таблица 3.6** Виды адресации памяти YRAM

Код режима адресации	YM	Режим адресации	Описание
00	X	-	Отмена пересылки
01	X	(AT)	Косвенная
10	X	(AT)+IT	Пост – автоувеличение
11	0	(AT)+IT	Индексирование (An не меняется)
11	1	(AT)+DT	Пост – автоувеличение

Выбор адресной арифметики для памяти YRAM определяется состоянием регистра MT в соответствии с правилами, описанными выше.

## 3.5 Устройство программного управления (PCU)

В настоящем разделе рассматривается устройство программного управления (PCU) и работа программного конвейера DSP.

### 3.5.1 Назначение и состав PCU

Устройство программного управления PCU контролирует выборку команд, их декодирование, аппаратно поддерживает организацию цикла DO. Программная модель PCU содержит следующие регистры и стеки:

- Регистр управления и состояния DCSR – 16 бит, чтение/запись;
- Программный счетчик PC – 16 бит, чтение/запись;
- Регистр состояния SR – 16 бит, разряды [7:0] – только чтение, разряды [15:8] – чтение/запись;
- Регистр-идентификатор IDR – 16 бит, доступен только по чтению;
- Регистр адреса окончания цикла LA – 16 бит, чтение/запись;
- Регистр счетчика циклов LC – 16 бит, чтение/запись;
- Системный стек SS – 16 бит, чтение/запись;
- Стек циклов CSH – 16 бит, чтение/запись;
- Стек циклов CSL – 16 бит, чтение/запись;

- Регистр указателей стека SP – 16 бит, чтение/запись;
- Счетчик команд CNTR – 16 бит, чтение/запись;
- Регистр адреса останова SAR – 16 бит, чтение/запись.

Системный стек SS представляет собой внутреннюю последовательно адресуемую память объемом 15 16-разрядных слов, используемую для автоматического сохранения содержимого регистра программного счетчика PC при входе в подпрограмму или в программный цикл (DO, DOFOR).

Стек циклов CS предназначен для сохранения содержимого регистров счетчика цикла и адреса окончания цикла (LC и LA) при организации вложенных программных циклов. Каждая 32-разрядная ячейка стека адресуется как два 16-разрядных регистра – верхний CSH и нижний CSL регистры стека. Адресация стеков осуществляется при помощи регистра указателей стека SP.

Другие данные могут сохраняться в стеках и считываться из них при соответствующих обращениях. Стеки участвуют в обменах как 16-разрядные регистры управления – SS, CSL и CSH.

Устройство PCU управляет режимами работы DSP-ядра. DSP-ядро всегда находится в одном из трех возможных состояний (режимов):

- режим сброса (RESET);
- режим останова (STOP);
- режим выполнения программы (RUN).

В штатном режиме функционирования устройство PCU организует выполнение инструкций при помощи программного конвейера, включающего три фазы.

### 3.5.2 Архитектура PCU

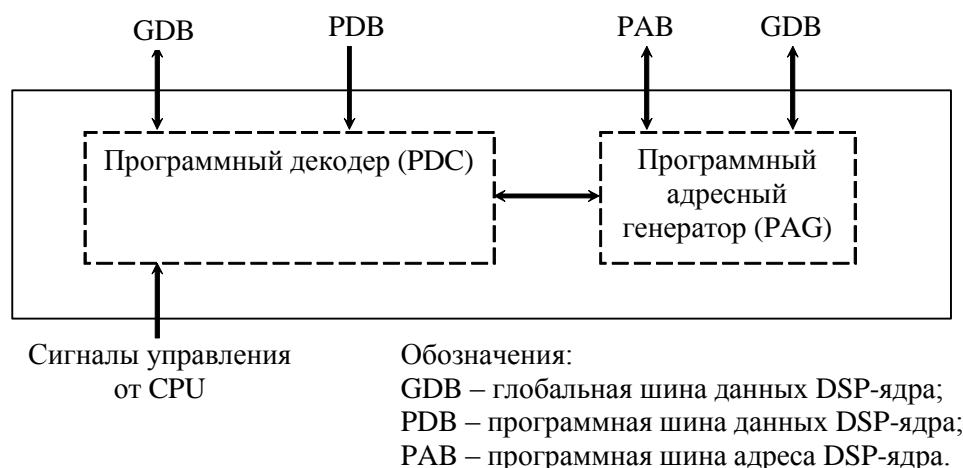
Устройство PCU включает в себя два аппаратных блока:

- Программный адресный генератор PAG;
- Программный декодер PDC.

Устройство PDC декодирует инструкции, поступающие из программной памяти, и генерирует сигналы управления программным конвейером.

Программный адресный генератор PAG выполняет вычисление адреса инструкции в программной памяти, организует выполнение программных циклов DO и операции REPEAT, управляет работой системного стека.

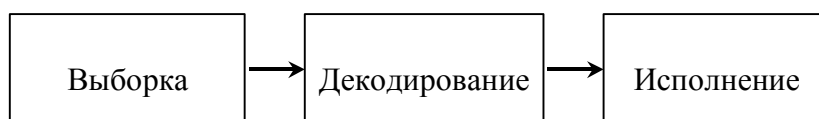
Ниже приведена структурная схема PCU.



### 3.5.3 Программный конвейер

Устройство программного управления организует конвейерный механизм исполнения инструкций DSP-ядра.

Программный конвейер включает в себя три стадии (фазы): стадию выборки команды из программной памяти (Fetch), стадию декодирования команды (Decode), стадию исполнения (Execute). Стадии конвейера изображены на рисунке ниже.



Конвейеризация выполнения инструкций приводит к тому, что в один и тот же момент времени происходит обработка нескольких инструкций, находящихся в разных стадиях исполнения.

Описание стадий конвейера приведено в таблице ниже.

При этом для большинства инструкций скорость их выполнения в конвейерном режиме составляет одну инструкцию в течение одного командного цикла. Исключение составляют инструкции программных переходов.

Стадия конвейера	Описание
Выборка	Чтение инструкции из программной памяти. Генерация адреса следующей инструкции.
Декодирование	Декодирование инструкции
Исполнение	Исполнение инструкции

Полная информация о времени выполнения различных типов инструкций содержится в документе «DSP-ядро ELcore-x4. Система инструкций».

### 3.5.4 Программная модель PCU

Устройство PCU содержит регистры LA и LC, предназначенные для аппаратной поддержки программного цикла DO, а также стандартные ресурсы программного управления, такие как программный счетчик PC, регистр состояния SR, стек циклов CS и системный стек SS. Все регистры доступны как по записи, так и по чтению, что облегчает отладку системы.

Программная модель PCU представлена на рисунке ниже. Далее дается описание назначения всех программно-доступных регистров и стеков.

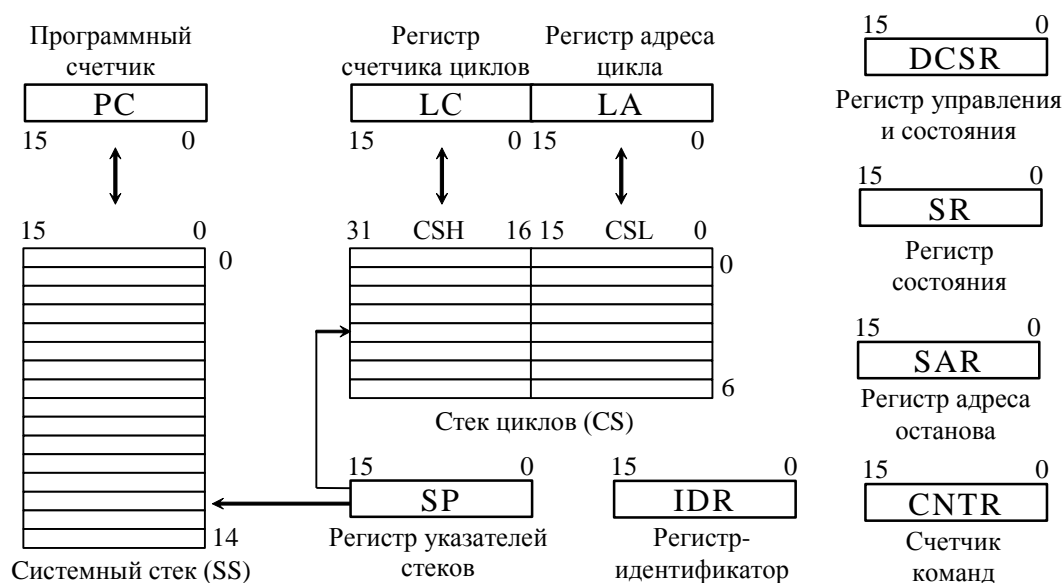


Рисунок 3.8

### 3.5.4.1 Регистр-идентификатор (IDR)

Регистр-идентификатор IDR содержит код версии DSP-ядра согласно приводимой ниже таблице. Доступен только по чтению.

IDR[15:0]	Модификация DSP-ядра
0x0003	DSP-ядро ELcore_14
Другие коды	Другие модификации DSP-ядра

### 3.5.4.2 Регистр управления и состояния (DCSR)

Регистр управления и состояния (DCSR) содержит разряды управления, определяющие состояние и режим работы DSP-ядра, а также прерывания, формируемые DSP для обработки в CPU. Назначение разрядов регистра DCSR указано ниже.

#### DCSR:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RST	RUN	-	DBG	-	-	-	-	DE3	DE2	DE1	DE0	STP	BRK	SE	PI

RST – программный RESET;

RUN - состояние исполнения программы;

DBG – режим отладки.

DE3- запуск DMA (3 канал);

DE2- запуск DMA (2 канал);

DE1- запуск DMA (1 канал);

DE0- запуск DMA (0 канал);

STP – прерывание по останову STOP;

BRK – прерывание по останову BREAK;

SE – прерывание по ошибке стека SE;

PI – программное прерывание PI.

Начальное состояние DCSR = 0x0000.

#### 3.5.4.2.1 Флаг прерывания PI

Флаг прерывания PI (программное прерывание) устанавливается в «1» в случае наличия программного прерывания со стороны DSP. Это прерывание формируется исполняемой программой DSP при помощи команды пересылки данных MOVE DSP-ядра. После обработки прерывания в CPU этот бит может быть снова установлен в «0» как по команде DSP, так и по команде CPU.

#### 3.5.4.2.2 Флаг прерывания SE

Флаг прерывания SE (ошибка стека) устанавливается в «1» в случае наличия признака ошибки одного из стеков DSP (разряды SSE или CSE регистра указателя стека SP). Это прерывание формируется при выходе указателя стека за пределы разрешенных значений. После обработки прерывания этот бит может быть сброшен в «0» по команде CPU.

#### 3.5.4.2.3 Флаг прерывания BRK

Флаг прерывания BRK (останов “BREAK”) устанавливается в «1» в случае останова DSP по одной из следующих причин:

- 1) по достижении адреса останова при исполнении программы до адреса останова;
  - 2) по завершении требуемого числа шагов при пошаговом исполнении программы.
- После обработки прерывания этот бит может быть сброшен в «0» по команде CPU.

#### 3.5.4.2.4 Флаг прерывания STP

Флаг прерывания STP устанавливается в «1» в случае останова DSP-ядра при исполнении команды STOP. После обработки прерывания этот бит может быть сброшен в «0» по команде CPU.

#### 3.5.4.2.5 Механизм взаимной синхронизации DSP и DMA. Флаги обменов DE0 - DE3

Биты DE0, DE1, DE2, DE3 (разряды 4-7 регистра DCSR) являются признаками готовности DSP-ядра к обменам с DMA.

- DE0 - Флаг запуска DMA со стороны DSP для обмена по каналу #0 DSP
- DE1 - Флаг запуска DMA со стороны DSP для обмена по каналу #1 DSP
- DE2 - Флаг запуска DMA со стороны DSP для обмена по каналу #2 DSP
- DE3 - Флаг запуска DMA со стороны DSP для обмена по каналу #3 DSP

Наличие этих бит позволяет синхронизировать обращения к двухпортовой памяти данных со стороны DSP и DMA.

В соответствии с состоянием признаков DE0, DE1, DE2, DE3 DMA по соответствующему каналу производит загрузку или выгрузку очередных блоков данных в память данных, после завершения которых, DMA аппаратно запускает DSP при помощи сигнала START.

Далее этот процесс может повторяться неограниченное число раз, позволяя синхронизировать процесс выполнения программы DSP-ядра и обменов между ним и DMA без участия управляющего RISC-ядра (CPU).

Установленные в «1» признаки DE0, DE1, DE2, DE3 находятся в этом состоянии в течение одного командного цикла, после чего аппаратно автоматически сбрасываются в «0» (при условии, что DSP находится в состоянии исполнения программы).

### 3.5.4.2.6 Бит DBG

Этот бит (совместно с битом RUN) используется для запуска исполнения программы DSP-ядра в режиме отладки.

### 3.5.4.2.7 Бит RUN

Управление состоянием DSP-ядра производится при помощи управляющего бита RUN (разряд 14 регистра DCSR).

Установка бита RUN в «1» переводит DSP-ядро в состояние исполнения программы, установка в «0» - в состояние останова.

### 3.5.4.2.8 Бит RST

Установка DSP-ядра в начальное состояние (состояние RESET) может быть произведена посредством записи «1» в бит RST (разряд 15 регистра DCSR).

Переход DSP-ядра в начальное состояние происходит в течение одного командного цикла, после чего бит RST автоматически сбрасывается в «0».

## 3.5.4.3 Регистр программного счетчика (PC)

Регистр программного счетчика PC предназначен для хранения 16-разрядного адреса инструкции в программной памяти. Инкрементированное значение PC заносится в системный стек при инициализации нового программного цикла DO, DOFOR и при входе в подпрограмму.

Начальное состояние PC = 0x0000.

## 3.5.4.4 Регистр состояния (SR)

Разряды [7:0] регистра SR доступны только по чтению, остальные - по записи/чтению.

Назначение разрядов регистра SR указано ниже.

**SR:**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SI	SRSI	BC	YM	-	-	-	t	E	Ev	U	N	Z	V	C	

SI – признак режима SIMD;

SRSI – способ формирования интегральных признаков в режиме SIMD;

BC - признак режима “BroadCasting”, т.е. одновременной загрузки памяти данных всех секций DSP-ядра;

YM – режим адресации памяти YRAM;

C – перенос;

V – признак переполнения;

Z - признак нулевого результата;

N - признак отрицательного результата;

U - признак ненормализованного результата;

Ev- флаг переполнения (с сохранением);

E – экспоненциальный признак;

t – признак истинности последнего условия.

Разряды [7:0] регистра SR содержат интегральные признаки предыдущей арифметической операции.

Разряд 11 регистра SR (бит YM) предназначен для выбора режима адресации генератора AGU-Y.

Остальные разряды регистра SR предназначены для работы в режиме SIMD в многосекционных модификациях DSP-ядра.

При начальной установке все разряды регистра SR обнуляются.

### 3.5.4.5 Регистр счетчика циклов (LC)

Регистр счетчика циклов содержит:

- 1) Текущее значение 14-разрядного счетчика программных циклов  $N_c$  – разряды 0-13 регистра LC;
- 2) LF – Флаг цикла DO – разряд 14 регистра LC;
- 3) FV - Флаг цикла DOFOR – разряд 15 регистра LC.

Формат регистра LC приведен ниже.

**LC:**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FV	LF	$N_c$													

Начальное состояние LC = 0x0000.

Значение счетчика программных циклов  $N_c$  определяет количество повторений программного цикла DO, в пределах от 1 до  $(2^{14} - 1)$ . Этот регистр заносится в верхнюю (старшую) половину стека циклов CSH по команде DO (образуется вложенный программный цикл) и извлекается обратно по окончании вложенного цикла либо по команде ENDDO.

Флаг цикла DO (LF) устанавливается в «1» в случае выполнения команды DO. Бит LF сохраняется в стеке при инициализации другого программного цикла. При окончании программного цикла происходит выталкивание из стека этого флага. Такой механизм позволяет организовывать вложенные циклы.

Флаг цикла выталкивается из стека при завершении цикла.

Исполнение программного цикла начинается с команды DO и продолжается до тех пор, пока адрес выбранной команды не сравнивается с содержимым регистра адреса цикла (последним адресом программного цикла).

После этого содержимое счетчика циклов сравнивается с единицей: если оно не равно (единице), то значение счетчика уменьшается на один и "верхнее" слово стека считывается в PC, не извлекаясь при этом из стека, для того чтобы возвратиться в начало цикла.

Если же содержимое счетчика циклов равно единице, то это означает, что программный цикл завершен. При этом прибавляется единица к содержимому PC, флаг предыдущего цикла считывается из верхнего слова соответствующего стека в регистры LC, LA и PC, сами стеки очищаются (т.е. выталкивается верхнее слово и заменяется его содержимое), из него извлекаются предыдущие значения (регистров) LA и LC и восстанавливаются в соответствующих регистрах.

По завершении цикла флаг цикла, LA и LC регистры, также как и указатели стеков, восстанавливаются.

Флаг цикла DOFOR (FV) устанавливается в «1» в случае выполнения команды DOFOR.

Бит FV сохраняется в системном стеке при вызове подпрограммы или инициализации другого программного цикла. При выходе из подпрограммы или окончании программного цикла происходит выталкивание из стека этого флага. Такой механизм позволяет организовывать вложенные циклы.

### 3.5.4.6 Регистр адреса цикла (LA)

Регистр адреса цикла (LA) является специализированным 16-разрядным регистром, содержащим адрес последней инструкции в программном цикле DO. Этот регистр заносится в нижнюю (младшую) половину стека циклов CSL по команде DO и извлекается обратно по окончании вложенного цикла либо по команде ENDDO.

### 3.5.4.7 Системный стек (SS)

Системный стек (SS) представляет собой специализированный модуль памяти объемом 16 слов по 16 разрядов. Системный стек используется для хранения состояния программного счетчика при вызовах подпрограмм и при организации программных циклов.

При входе в подпрограмму (т.е. при выполнении команд JS, BS) адрес возврата автоматически сохраняется в SS.

При возврате из подпрограммы по команде RTS содержимое верхней ячейки SS загружается обратно в PC.

Стек используется также при реализации вложенных программных циклов DO, DOFOR. При входе в программный цикл DO адрес первой инструкции программного цикла сохраняется в SS.

Глубина стека – 15 слов по 16 разрядов (16-е слово не используется) – определяет количество вложенных процедур.

*Всего могут быть вложенными друг в друга до семи программных циклов, либо до пятнадцати подпрограмм, либо их различные комбинации.*

Адрес ячейки стека, к которой производится обращение, определяется 4-разрядным указателем стека SP[3:0], хранящемся в регистре указателя стека SP. При этом адрес записи совпадает с текущим значением указателя, адрес чтения на единицу меньше. Все внутренние обращения к стеку (т.е. обращения, происходящие по командам DSP) приводят к изменению указателя: при записи он инкрементируется, при чтении – декрементируется. Внешние обращения к стеку, т.е. обращения со стороны RISC-процессора или устройства отладки OnCD, не изменяют значение указателя.

При выходе значения указателя стека за разрешенные пределы формируется флаг “ошибка стека” SSE.

### 3.5.4.8 Стек цикла (CS)

Стек цикла (CS) представляет собой специализированный модуль памяти объемом 8 слов по 32 разряда. Стек состоит из двух половин объемом каждая  $8 * 16$  – верхней CSH и нижней CSL. Стек цикла используется для хранения содержимого регистров LA и LC при организации вложенных программных циклов.

При входе в программный цикл DO предыдущее содержимое регистра счетчика циклов (LC) автоматически сохраняется в CSH, а предыдущее содержимое регистра адреса цикла (LA) автоматически сохраняется в CSL и инкрементируются соответствующие указатели стеков SP. (Адрес первой инструкции программного цикла DO сохраняется в SS).

*Глубина стека – 7 слов по 32 разряда (8-е слово не используется) – определяет количество вложенных циклов. Всего могут быть вложенными друг в друга до семи программных циклов DO.*

Адрес ячейки стека, к которой производится обращение, определяется 3-разрядным указателем стека SP[2:0], хранящемся в регистре указателя стека SP. При этом адрес



записи совпадает с текущим значением указателя, адрес чтения на единицу меньше. Все внутренние обращения (т.е. обращения, происходящие по командам DSP) к стеку CSH приводят к изменению указателя: при записи он инкрементируется, при чтении – декрементируется. Внешние обращения к стеку CSH, т.е. обращения со стороны RISC-процессора или устройства отладки OnCD, не изменяют значение указателя. Также не влияют на значение указателя любые обращения к стеку CSL.

При выходе значения указателя стека за разрешенные пределы формируется флаг “ошибка стека” CSE.

### 3.5.4.9 Регистр указателей стека (SP)

Регистр указателей стека SP содержит указатели на последнее записанное в стеки SS, CSH слово. Младший байт регистра SP содержит указатель и флаги системного стека; старший байт - указатель и флаги стека циклов.

Назначение разрядов регистра SP указано ниже.

**SP:**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	UFC	CSE	CP[2:0]		-	-	UFS	SSE	SP[3:0]				

CP[2:0] – указатель стека циклов;

CSE – флаг ошибки стека циклов;

UFC – флаг переполнения стека циклов .

Начальное состояние SP = 0x0000.

SP[2:0] – указатель системного стека;

SSE – флаг ошибки системного стека;

UFS – флаг переполнения системного стека.

Значения указателей и флагов приведены в Таблица 3.7, Таблица 3.8.

#### 3.5.4.9.1 Указатель системного стека (SP[3:0])

Указатель системного стека - разряды SP[3:0] регистра SP указывает на незанятую ячейку стека SS с наименьшим адресом. По сигналу ALU начальной загрузки (RESET) эти разряды устанавливаются в нулевое состояние, показывая, что стек пуст.

Данные поступают в стек с одновременной инкрементацией указателя. Выборка данных из стека сопровождается декрементацией указателя.

#### 3.5.4.9.2 Флаг ошибки системного стека (SSE)

Флаг ошибки стека (разряд SSE регистра SP) в состоянии «1» указывает на выход указателя стека за пределы разрешенных значений.

При заполненном системном стеке значение, хранящееся в разрядах [5:0] SP, равно 001111. Попытка записи данных в системный стек в этом случае приводит к возникновению “ошибки стека” и переходу SP[5:0] в состояние 010000.

Любая операция выборки из пустого стека (SP=0) приводит его в состояние 111111. В этом случае флаг ошибки стека SSE также устанавливается в “1”.

После перехода в состояние “1” флаг ошибки стека сохраняется в этом состоянии до тех пор, пока не будет сброшен пользователем.

**Таблица 3.7 Разрешенные значения указателя системного стека**

UFS	SSE	SP3	SP2	SP1	SP0	Описание
1	1	1	1	1	1	Переполнение стека «вниз»
0	0	0	0	0	0	Стек пуст. Попытка чтения приводит к переполнению стека «вниз»
0	0	0	0	0	1	Ячейка стека 1
.	.	.	.	.	.	Ячейки стека 2-13
0	0	1	1	1	0	Ячейка стека 14
0	0	1	1	1	1	Ячейка стека 15. Стек полон. Попытка записи приводит к переполнению стека «вверх»
0	1	0	0	0	0	Переполнение стека «вверх»

### 3.5.4.9.3 Флаг исчерпания системного стека (UFS)

Флаг исчерпания системного стека (разряд UFS регистра SP) в состоянии «1» указывает на выход указателя стека за пределы разрешенных значений «вниз», т.е. попытку считать из пустого стека. При этом одновременно флаг ошибки стека переходит в состояние «1».

После перехода в состояние «1» флаг переполнения стека сохраняется в этом состоянии до тех пор, пока не будет сброшен пользователем.

### 3.5.4.9.4 Указатель стека циклов (CS[2:0])

Указатель стека циклов - разряды CS [2:0] регистра SP указывает на незанятую ячейку стека циклов CS с наименьшим адресом. По сигналу начальной загрузки (RESET) эти разряды устанавливаются в нулевое состояние, показывая, что стек пуст.

Данные поступают в стек циклов с одновременной инкрементацией указателя CS. Выборка данных из стека сопровождается декрементацией указателя.

### 3.5.4.9.5 Флаг ошибки стека циклов (CSE)

Флаг ошибки стека (разряд CSE регистра SP) в состоянии «1» указывает на выход указателя стека за пределы разрешенных значений (см. табл.6.5.2).

При заполненном стеке значение, хранящееся в разрядах [12:8] SP, равно 00111. Попытка записи данных в стек в этом случае приводит к возникновению «ошибки стека» CSE и переходу SP[12:8] в состояние 01000.

Любая операция выборки из пустого стека циклов (CS=0) приводит его в состояние 11111. В этом случае флаг ошибки стека циклов CSE устанавливается в «1».

После перехода в состояние «1» флаг ошибки стека циклов сохраняется в этом состоянии до тех пор, пока не будет сброшен пользователем.

**Таблица 3.8 Разрешенные значения указателя стека циклов**

UFC	CSE	CS2	CS1	CS0	Описание
1	1	1	1	1	Переполнение стека циклов «вниз»
0	0	0	0	0	Стек циклов пуст. Попытка чтения приводит к переполнению стека «вниз»
0	0	0	0	1	Ячейка стека циклов 1
.	.	.	.	.	Ячейки стека циклов 2-5

0	0	1	1	0	Ячейка стека циклов 6
0	0	1	1	1	Ячейка стека 7. Стек циклов полон. Попытка записи приводит к переполнению стека «вверх»
0	1	0	0	0	Переполнение стека циклов «вверх»

### 3.5.4.9.6 Флаг исчерпания стека циклов - (UFC)

Флаг исчерпания стека циклов (разряд UFC регистра SP) в состоянии «1» указывает на выход указателя стека за пределы разрешенных значений «вниз», т.е. попытку считать из пустого стека. При этом одновременно флаг ошибки стека переходит в состояние «1».

После перехода в состояние «1» флаг переполнения стека циклов сохраняется в этом состоянии до тех пор, пока не будет сброшен пользователем.

### 3.5.4.9.7 Регистр адреса останова (SAR)

Регистр адреса останова SAR является специализированным 16-разрядным регистром, используемым при отладке DSP-ядра. Регистр SAR определяет точку останова (Breakpoint) - адрес инструкции, непосредственно перед исполнением которой должен произойти останов DSP-ядра. Перед исполнением инструкции с указанным адресом DSP-ядро переходит в состояние останова (RUN=0) и флаг прерывания BRK устанавливается в «1».

Начальное состояние SAR = 0xFFFF.

### 3.5.4.10 Счетчик команд (CNTR)

Счетчик команд CNTR - специализированный 16-разрядный регистр, предназначенный для отладки DSP-ядра. Регистр CNTR задает пошаговый режим исполнения программ в соответствии с приводимой ниже таблицей.

CNTR	Режим исполнения программ
0x0000	Нормальный режим исполнения программ. Число исполняемых команд не ограничено.
N > 0	Пошаговый режим исполнения программ. После исполнения N инструкций DSP-ядро переходит в состояние останова (RUN=0) и флаг прерывания BRK устанавливается в «1».

## 3.6 Программная модель DSP

Программная модель DSP представлена на Рисунок 3.9.

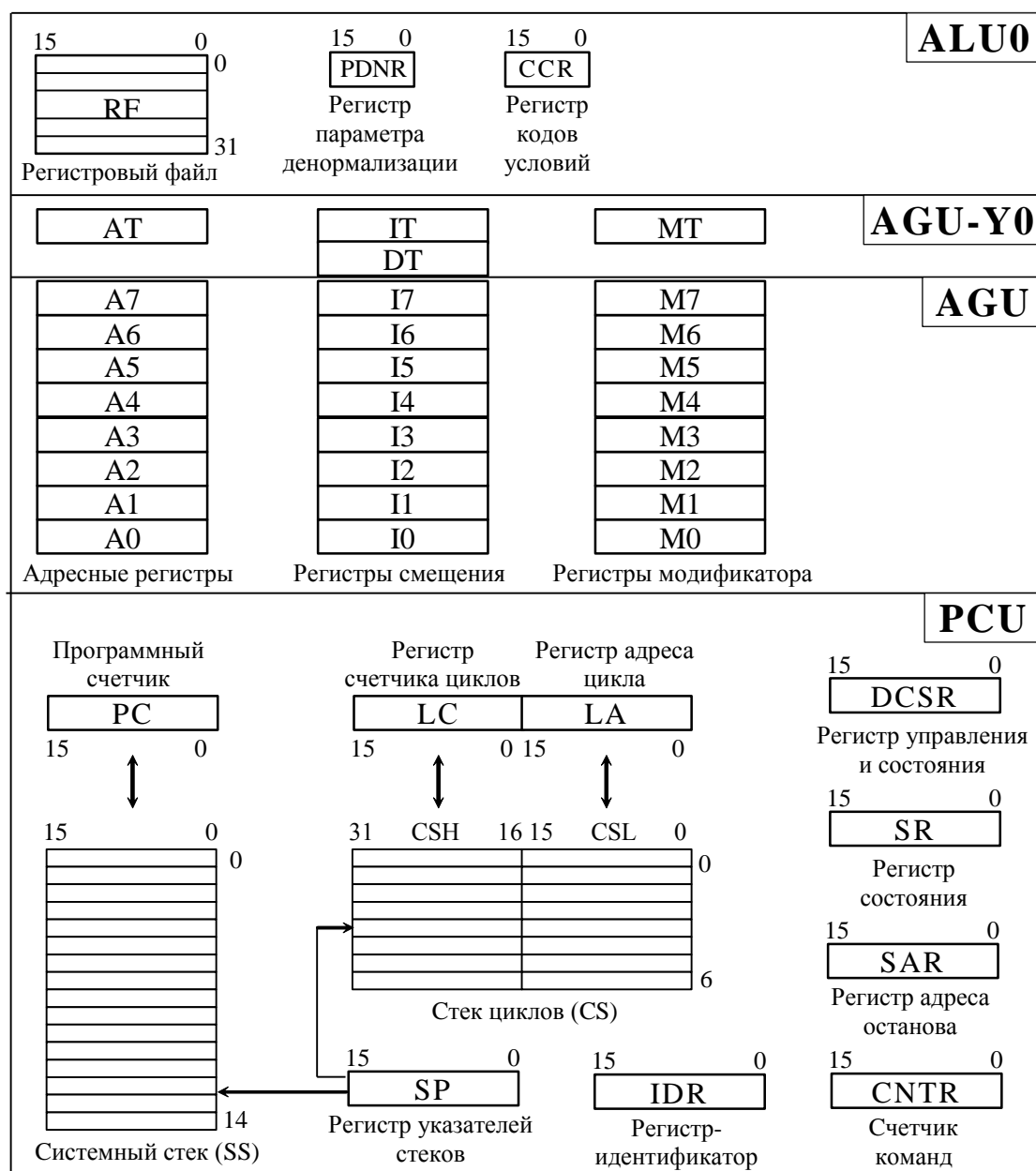


Рисунок 3.9 Программная модель DSP-ядра ELcore-14

### 3.7 Состояния DSP

В этом разделе описываются состояния (режимы функционирования) DSP. Управление состояниями DSP может выполняться при помощи сигнала аппаратного сброса RESET либо путем изменения соответствующих разрядов регистра DCSR.

DSP-ядро всегда находится в одном из трех возможных состояний:

- состояние начальной установки (**RESET**);
- состояние останова (**STOP**);
- состояние исполнения программы (**RUN**).

Ниже дается описание указанных состояний.

### 3.7.1 Состояние начальной установки (RESET)

DSP-ядро переходит в состояние начальной установки в двух случаях:

- 1) при поступлении сигнала аппаратного сброса RESET (аппаратный RESET);
- 2) при записи «1» в 15-й разряд регистра DCSR (программный RESET).

В обоих этих случаях производятся следующие установки:

- Регистры управления DCSR, SR, PC, LC, CNTR, SP адресные регистры A0-A7, AT, секционные регистры CCR, PDNR, AC0, AC1 устанавливаются в состояние 0x0000;
- Регистр адреса останова SAR и регистры модификатора адреса M0-M7, MT устанавливаются в состояние 0xFFFF.

### 3.7.2 Состояние останова (STOP)

При переходе в состояние останова DSP-ядро прекращает выполнение текущей программы. Программный счетчик не инкрементируется, состояние регистров и памяти сохраняется неизменным, за исключением тех случаев, когда производятся обмены по шинам CPU-ядра или DMA.

DSP-ядро переходит в состояние останова при отсутствии аппаратного сброса в одном из описанных ниже случаев:

- при установке в «0» бита RUN регистра DCSR;
- по достижении адреса останова при исполнении программы до адреса останова (при этом устанавливается в «1» флаг прерывания BREAK регистра DCSR);
- по завершении требуемого числа шагов при пошаговом исполнении программы;
- при отработке команды STOP DSP (при этом устанавливается в «1» флаг прерывания STOP регистра DCSR);
- при установке флага ошибки в одном из регистров указателей стеков – SSE или CSE (при этом устанавливается в «1» флаг прерывания SE регистра DCSR);

### 3.7.3 Состояние исполнения программы (RUN)

DSP-ядро находится в этом состоянии при одновременном наличии следующих условий:

- 1) Бит RUN регистра DCSR установлен в «1»;
- 2) Не установлены (находятся в состоянии «0») флаги прерываний SE, BREAK, STOP регистра DCSR.

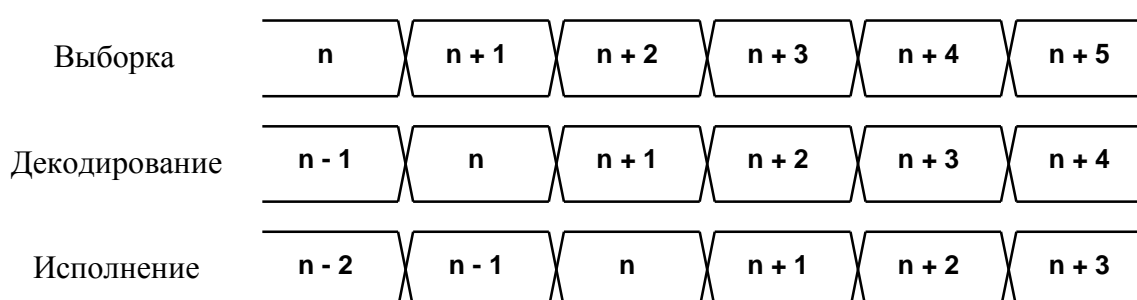
Состояние DSP-ядра RUN связано с выполнением команд (инструкций). Выполнение инструкций в DSP-ядре организовано в виде конвейера, включающего три фазы. При этом для большинства инструкций скорость их выполнения в конвейерном режиме составляет одну инструкцию в течение одного командного цикла.

Выполнение некоторых инструкций требует большего количества командных циклов. К ним относятся инструкции, вызывающие программные переходы.

Конвейеризация выполнения инструкций приводит к тому, что в один и тот же момент времени происходит обработка нескольких инструкций, находящихся в разных стадиях исполнения.

Программный конвейер включает в себя три стадии (фазы): Выборка (Fetch), Декодирование (Decode), Исполнение (Execute). Хотя от выборки первой инструкции до окончательного ее исполнения проходит три командных цикла, с каждым следующим циклом завершается очередная инструкция.

Работа программного конвейера при последовательной выборке команд из программной памяти иллюстрируется временной диаграммой на Рисунок 3.10 ( $n$  – номер инструкции).



**Рисунок 3.10 Работа программного конвейера при последовательной выборке команд**

Приведенный в таблице порядок следования инструкций имеет место для большинства инструкций, исполнение которых не требует дополнительных командных циклов. Исключение составляют инструкции программных переходов.

Внешние обращения (со стороны RISC-ядра или DMA) к регистрам или к сегментам программной памяти DSP-ядра вызывают приостановку программного конвейера и приводят, таким образом, к увеличению времени исполнения инструкций на соответствующее число тактов.

Состояние DSP-ядра при этом не меняется.

Обращения к двухпортовой памяти данных XRAM, YRAM происходят без приостановки программного конвейера.

### 3.8 Карта памяти DSP

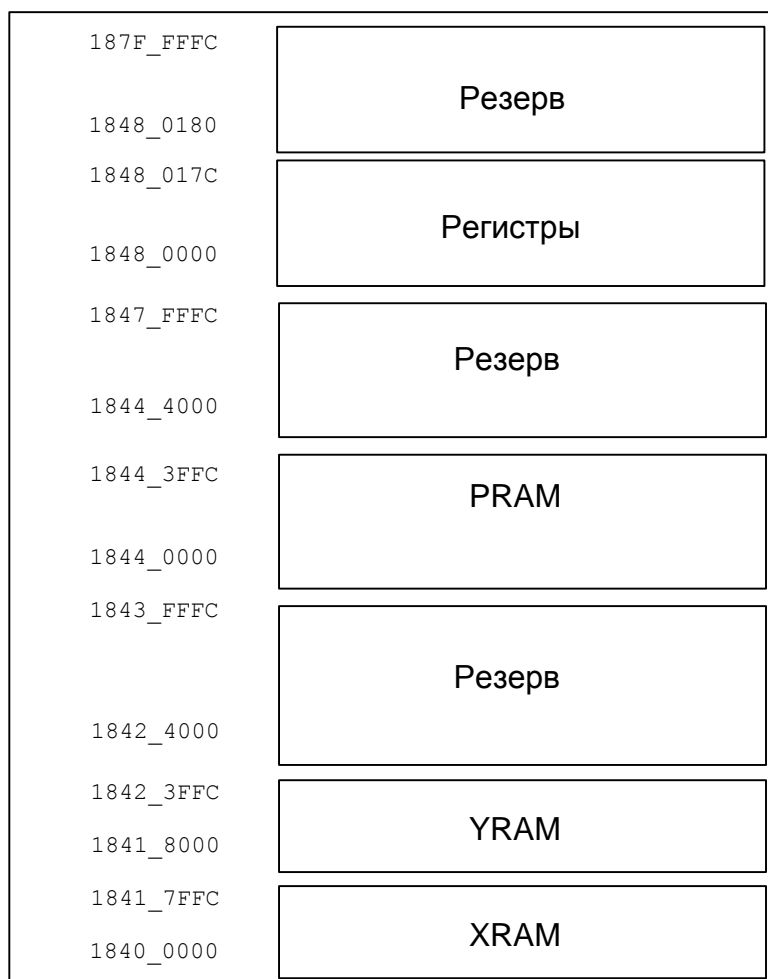
Внутренняя оперативная память DSP входит в общее пространство памяти CPU.

Положение сегментов памяти DSP в пространстве CPU приведено на Рисунок 3.11. Адреса указаны в шестнадцатеричной системе счисления с точностью до одного байта. При этом в пространстве памяти DSP возможны только 32-разрядные обмены. Поэтому при обменах с CPU и DMA два младших разряда адреса считаются всегда равными нулю.

Под память данных XRAM и YRAM отведен диапазон адресов с 0x1840\_0000 по 0x1842\_3FFC.

Под память программ PRAM отведен диапазон адресов с 0x1844\_0000 по 0x1844\_3FFC.

Программно-доступные регистры располагаются в диапазоне адресов с 0x1848\_0000 по 0x1848\_017C.



**Рисунок 3.11** Карта памяти DSP-ядра ELcore-14.

Обмены с памятью могут быть только 32-разрядными. Обмены с адресуемыми регистрами DSP могут производиться только CPU и могут быть 16- или 32-разрядными. При обменах с 16-разрядными регистрами данные находятся в младшем полуслове.

Во внешних обменах (с CPU или DMA) DSP является ведомым устройством (Slave) и не может самостоятельно инициировать обмен. Обмены CPU или DMA с памятью DSP (XRAM, YRAM или PRAM) происходят через отдельные порты модулей памяти и не прерывают работы DSP.

### 3.8.1 Организация обменов с памятью данных

Общее пространство памяти данных DSP состоит из двух областей: X- и Y-памяти (XRAM, YRAM). Под память данных XRAM отведен диапазон адресов с 0x1840\_0000 по 0x1841\_7FFC. Под память данных YRAM отведен диапазон адресов с 0x1841\_8000 по 0x1842\_3FFC.

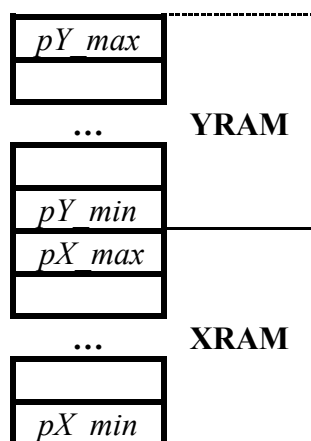
Генерация адресов для X- и Y-памяти данных при внутренних обменах DSP осуществляется адресными генераторами DSP - AGU и AGU-Y.

Устройство AGU-Y предназначено для генерации адресов Y-памяти.

Адресный генератор AGU является общим для всего DSP и производит адресацию всех сегментов X- и Y-памяти данных DSP.

Устройство AGU-Y адресует только Y-память и только по чтению. При одновременном обращении к Y-памяти со стороны обоих генераторов, - AGU и AGU-Y, - приоритет имеет генератор AGU.

При этом внутренняя адресация памяти XRAM начинается с нулевого адреса, а памяти YRAM - с адреса, следующего за последним адресом XRAM в соответствии с приводимой ниже диаграммой, где  $pX\_min$ ,  $pX\_max$  – соответственно минимальный и максимальный адрес X-памяти  $pY\_min$ ,  $pY\_max$  – соответственно минимальный и максимальный адрес Y-памяти.



Ниже приводятся граничные адреса X- и Y-памяти для ELcore-14 (адреса приводятся в шестнадцатеричной системе счисления).

$pX\_min$	$pX\_max$	$pY\_min$	$pY\_max$
0x0000	0x5FFF	0x6000	0x8FFF



### 3.8.2 Организация памяти программ PRAM

Под память программ PRAM отведен диапазон адресов с 0x1844\_0000 по 0x1847\_FFFC.

Память программ PRAM имеет 64-разрядную организацию, позволяющую осуществлять хранение и выборку в течение одного такта как 32-разрядных, так и 64-разрядных инструкций. Объем памяти PRAM - 4К 32-разрядных (или 2К 64-разрядных) слов.

Память PRAM адресуется программным адресным генератором, входящим в состав устройства программного управления.

При последовательном ходе программы адрес программной памяти определяется состоянием программного счетчика PC, при программных переходах адрес определяется инструкцией перехода.

Доступ к программной памяти DSP со стороны RISC-ядра происходит *без приостановки программного конвейера*.

### 3.8.3 Адресуемые регистры

Перечень адресуемых регистров DSP-ядра с указанием их адреса в пространстве адресов памяти CPU-ядра приведен в Таблица 3.9.

**Таблица 3.9 Перечень адресуемых регистров DSP-ядра**

Условное обозначение	Разрядность	Название регистра	Адрес регистра
		<u>PCU</u>	
DCSR	16	Регистр режима работы	0x1848_0100
SR	16	Регистр состояния	0x1848_0104
IDR	16	Регистр-идентификатор	0x1848_0108
PC	16	Программный счетчик	0x1848_0120
SS	16	Стек программного счетчика	0x1848_0124
LA	16	Регистр адреса цикла	0x1848_0128
CSL	16	Стек адреса цикла	0x1848_012C
LC	16	Счетчик циклов	0x1848_0130
CSH	16	Стек счетчика циклов	0x1848_0134
SP	16	Регистр указателя стека	0x1848_0138
SAR	16	Регистр адреса останова	0x1848_013C
CNTR	16	Счетчик исполненных команд	0x1848_0140
		<u>AGU</u>	
A0	16	Регистр адреса A0	0x1848_0080
A1	16	Регистр адреса A1	0x1848_0084
A2	16	Регистр адреса A2	0x1848_0088
A3	16	Регистр адреса A3	0x1848_008C
A4	16	Регистр адреса A4	0x1848_0090

Условное обозначение	Разрядность	Название регистра	Адрес регистра
A5	16	Регистр адреса A5	0x1848_0094
A6	16	Регистр адреса A6	0x1848_0098
A7	16	Регистр адреса A7	0x1848_009C
I0	16	Регистр индекса I0	0x1848_00A0
I1	16	Регистр индекса I1	0x1848_00A4
I2	16	Регистр индекса I2	0x1848_00A8
I3	16	Регистр индекса I3	0x1848_00AC
I4	16	Регистр индекса I4	0x1848_00B0
I5	16	Регистр индекса I5	0x1848_00B4
I6	16	Регистр индекса I6	0x1848_00B8
I7	16	Регистр индекса I7	0x1848_00BC
M0	16	Регистр модификатора M0	0x1848_00C0
M1	16	Регистр модификатора M1	0x1848_00C4
M2	16	Регистр модификатора M2	0x1848_00C8
M3	16	Регистр модификатора M3	0x1848_00CC
M4	16	Регистр модификатора M4	0x1848_00D0
M5	16	Регистр модификатора M5	0x1848_00D4
M6	16	Регистр модификатора M6	0x1848_00D8
M7	16	Регистр модификатора M7	0x1848_00DC
AT	16	Регистр адреса AT	0x1848_00E0
IT	16	Регистр индекса IT	0x1848_00E4
MT	16	Регистр модификатора MT	0x1848_00E8
DT	16	Регистр модификатора DT	0x1848_00EC
<u>Регистры данных RF</u>			
R0.L	32	Регистр данных R0.L	0x1848_0000
R2.L	32	Регистр данных R2.L	0x1848_0004
R4.L	32	Регистр данных R4.L	0x1848_0008

Условное обозначение	Разрядность	Название регистра	Адрес регистра
R6.L	32	Регистр данных R6.L	0x1848_000 C
R8.L	32	Регистр данных R8.L	0x1848_0010
R10.L	32	Регистр данных R10.L	0x1848_0014
R12.L	32	Регистр данных R12.L	0x1848_0018
R14.L	32	Регистр данных R14.L	0x1848_001 C
R16.L	32	Регистр данных R16.L	0x1848_0020
R18.L	32	Регистр данных R18.L	0x1848_0024
R20.L	32	Регистр данных R20.L	0x1848_0028
R22.L	32	Регистр данных R22.L	0x1848_002 C
R24.L	32	Регистр данных R24.L	0x1848_0030
R26.L	32	Регистр данных R26.L	0x1848_0034
R28.L	32	Регистр данных R28.L	0x1848_0038
R30.L	32	Регистр данных R30.L	0x1848_003 C
<u>Секционные регистры состояния</u>			
CCR	16	Регистр кодов условий	0x1848_0160
PDNR	16	Регистр параметра денормализации	0x1848_0164
AC0	32	Регистр-аккумулятор 0	0x1848_0168
AC1	32	Регистр-аккумулятор 1	0x1848_016 C

Примечания:

1) Все регистры доступны как по записи, так и по чтению, за следующими исключениями:

- младший байт регистра SR доступен только по чтению;
- регистр IDR доступен только по чтению.

2) Обращение к любому из регистров приводит к приостановке программного конвейера, за следующими исключениями: чтение из регистров DCSR, SR, IDR, SAR, CNTR происходит без приостановки программного конвейера.

## 4. СИСТЕМНОЕ УПРАВЛЕНИЕ

### 4.1 Система синхронизации

1892BM3T имеет два входа синхронизации:

- Вход системной частоты ХТИ/ХТО. Сюда может подключаться кварцевый резонатор или внешний генератор;
- Вход частоты реального времени RTCХТИ.

Схема синхронизации узлов 1892BM3T приведена на Рисунок 4.1. Схема синхронизации узлов 1892BM3T

Для синхронизации работы узлов 1892BM3T используется умножитель частоты на основе схемы фазовой автоподстройки частоты PLL. Управление PLL осуществляется при помощи поля CLK\_SEL[4:0] (выбор коэффициента умножения/деления входной частоты) и внешнего вывода PLL\_EN:

при PLL\_EN=0 системная тактовая частота микроконтроллера равна входной частоте ХТИ;

при PLL\_EN=1 системная тактовая частота микроконтроллера поступает из PLL и равна входной частоте ХТИ, умноженной на коэффициент умножения/деления.

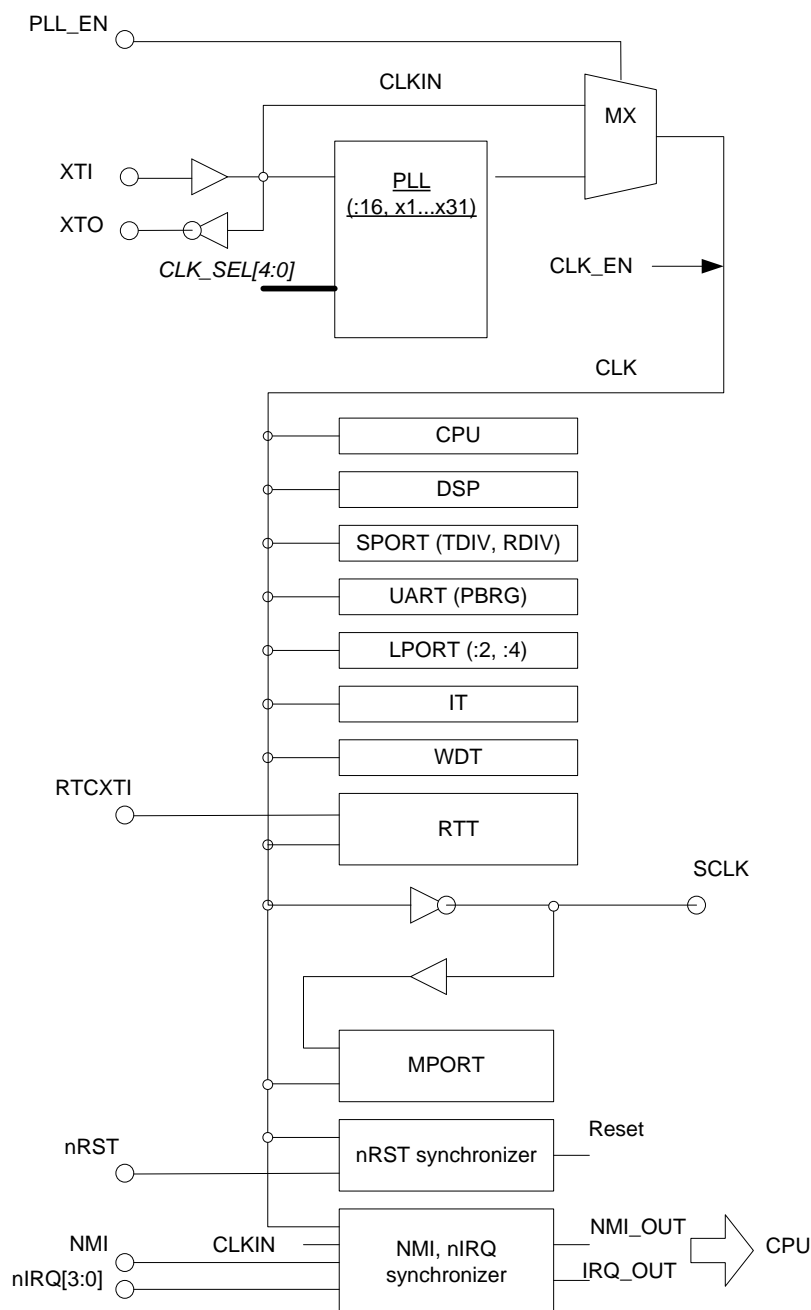
Сигнал CLKEN поступает с регистра CSR и управляет формированием частоты CLK.

CPU, DSP, IT, WDT, MPORT работают на частоте CLK.

Частота передачи данных линковыми портами (LPORT) – от CLK/2 до CLK/4.

Частота передачи данных последовательными портами (SPORT) определяется коэффициентом деления частоты CLK, который содержится в регистрах TDIV и RDIV.

Частота передачи данных UART определяется коэффициентом деления частоты CLK, который содержится в регистрах программируемого делителя (PBRG).



Reset - установка исходного состояния    NMI\_OUT, IRQ\_OUT - сигналы прерывания, поступающие на вход CPU  
 CLK - системная тактовая частота  
 CLKIN - входная тактовая частота    nRST, NMI, nIRQ synchronizer - схемы синхронизации входных сигналов

**Рисунок 4.1. Схема синхронизации узлов 1892BM3T**

## 4.2 Отключение и включение тактовой частоты

В 1892ВМ3Т имеется два режима энергосбережения:

- перевод DSP в режим STOP;
- отключение внутренней тактовой частоты CLK.

Перевод DSP в режим STOP осуществляется посредством регистра DCSR. Это позволяет уменьшить энергопотребление не менее чем на 30%.

Отключение внутренней тактовой частоты выполняется следующим образом:

- программа CPU должна выполняться из кэш программ или из внутренней памяти CRAM;
- SPORT, UART, DMA должны быть в неактивном состоянии;
- перевести DSP в режим STOP;
- записать 1 в 31 разряд регистра SDRCON (поле RFR не должно быть изменено). По данной операции SDRAM деактивируется (выполняется команда PRECHARGE);
- произвести запись нулей по адресу 182F\_1018 (установка выходного сигнала СКЕ в нулевое состояние);
- произвести запись 0 в разряд CLKEN регистра CSR. По этой операции внутренняя тактовая частота отключается. За этой командой должна стоять команда NOP.

При отключении внутренней тактовой частоты энергопотребление уменьшается не менее чем в 100 раз.

Включение внутренней тактовой частоты осуществляется по любому внешнему прерыванию nIRQ[3:0] или NMI. Обработка исключения по данным прерываниям в этом случае должна выполняться следующим образом:

- для определения факта того, что прерывание произошло при выключенной частоте, можно опросить состояние бита CLKEN=0;
- записать 1 в бит CLKEN;
- произвести запись всех единиц по адресу 182F\_1018 (установка сигнала СКЕ в единичное состояние);
- ожидание не менее 10 тактов.

## 4.3 Системные регистры

### 4.3.1 Регистр управления и состояния CSR

Формат регистра CSR приведен в Таблица 4.1.

Таблица 4.1

Номер разряда	Условное обозначение	Назначение	Доступ	Исходное состояние
0	FM	Режим преобразования виртуальных адресов CPU в физические адреса: 0 – с использованием TLB; 1 – Fixed Mapped (FM).	R/W	1
3:1	-	Резерв	-	0
8:4	CLK_SEL[4:0]	Управление PLL: выбор коэффициента умножения/деления входной частоты: 0 – 1/16; 1 – 1; 2 – 2; ... 29 – 29; 30 – 30; 31 – 31.	R/W	1
11:9	-	Резерв	-	0
12	FLUSH	При записи 1 в данный разряд кэш команд CPU устанавливается в исходное состояние, то есть ее содержимое девалидируется. Эта процедура может использоваться для обеспечения когерентности кэш при работе DMA.	W	0
15:13	-	Резерв	-	0
16	CLKEN	Управление формированием частоты CLK: 1 – частота включена; 0 – частота выключена.	R/W	1
31:17	-	Резерв	-	0

Нумерация разрядов регистров 1892BM3T соответствует нумерации разрядов памяти CPU. Если разряды регистров 1892BM3T доступны только по записи или не используются (резерв), то при чтении из них считываются нули. Если разряды регистров 1892BM3T доступны только по чтению или не используются, то при записи в них необходимо указывать нули.

### 4.3.2 Регистр запросов прерывания QSTR

Все сигналы внутренних прерываний поступают на вход псевдорегистра QSTR, формат которого приведен в Таблица 4.2

Данный регистр не имеет элементов памяти, и доступен только по чтению.

Каждый разряд регистра QSTR содержит запрос прерывания от внутренних узлов 1892BM3T вне зависимости от состояния соответствующих разрядов регистра MASKR:

0 – нет запроса;

1 – есть запрос.

Сигналы внутренних прерываний формируются в соответствующих устройствах при выполнении определенных условий. В процессе обслуживания прерывания необходимо проанализировать состояние устройства для определения причины его возникновения. Сброс прерывания осуществляется в момент исключения причины возникновения данного прерывания. Например, прерывание от LPORT (при неактивизированном DMA) сбрасывается при записи данных в буфер LTx или при чтении данных из буфера LRx.

Все незамаскированные прерывания объединяются по «или» и поступают в разряд IP[7] регистра Cause CPU.

Исходное состояние регистра QSTR – нули.

**Таблица 4.2**

Номер разряда	Условное обозначение прерывания	Название прерывания
0	SRx0	Прерывание от порта SPORT0 при приеме данных или от канала DMA SportRxCh0
1	STx0	Прерывание от порта SPORT0 при выдаче данных или от канала DMA SportTxCh0
2	SRx1	Прерывание от порта SPORT1 при приеме данных или от канала DMA SportRxCh1
3	STx1	Прерывание от порта SPORT1 при выдаче данных или от канала DMA SportTxCh1
4	Uart	Прерывание от UART
6:5	-	Резерв
7	LTRx0	Прерывание от порта LPORT0 при обмене данными или от канала DMA LportCh0
8	LSrq0	Запрос обслуживания от порта LPORT0
9	LTRx1	Прерывание от порта LPORT1 при обмене данными или от канала DMA LportCh0
10	LSrq1	Запрос обслуживания от порта LPORT1
11	LTRx2	Прерывание от порта LPORT2 при обмене данными или от канала DMA LportCh0
12	LSrq2	Запрос обслуживания от порта LPORT2
13	LTRx3	Прерывание от порта LPORT3 при обмене данными или от канала DMA LportCh0
14	LSrq3	Запрос обслуживания от порта LPORT3
18:15	-	Резерв
19	Compare	Прерывание от таймера CPU
20	-	Резерв
21	MemCh0	Прерывание от канала DMA MemCh0
22	MemCh1	Прерывание от канала DMA MemCh1
23	MemCh2	Прерывание от канала DMA MemCh2
24	MemCh3	Прерывание от канала DMA MemCh3
28:25	-	Резерв
29	Timer	Прерывание от таймеров IT, WDT, RTT
30	PI	Программное прерывание от DSP-ядра.



31	SBS	Признаки: Переполнение стека DSP-ядра. Остановка DSP-ядра в результате сравнения содержимого программного счетчика с адресом останова. Остановка DSP-ядра при завершении требуемого числа шагов при пошаговом исполнении программы. Выполнение DSP-ядром команды STOP.
----	-----	--

### 4.3.3 Регистр маски MASKR

Каждое внутреннее прерывание маскируется при помощи 32-разрядного регистра маски MASKR, формат которого аналогичен формату регистра QSTR. Исходное состояние данного регистра – нули (все внутренние прерывания запрещены). Регистр доступен по записи и чтению.

## 4.4 Процедура начальной загрузки

После снятия сигнала nRST выполняется следующее:

- Все устройства 1892BM3T устанавливаются в исходное состояние;
- DSP устанавливается в состояние STOP;
- в CPU возникает исключение, вектор которого расположен по физическому адресу 0x1FC0\_0000 внешней памяти. В этой области, как правило, расположено постоянное запоминающее устройство (ПЗУ) или, например память типа Flash.

В зависимости от состояния сигнала на выводе BYTE ПЗУ может быть 8 – или 32 – разрядным.

В ПЗУ может находиться или только программа начальной загрузки или все программы 1892BM3T. В первом случае основная программа 1892BM3T может быть загружена через линковые или последовательные порты.

Программа начальной загрузки должна обеспечивать конфигурирование всех устройств 1892BM3T.

## 4.5 Логика взаимодействия CPU и DSP

### 4.5.1 Функции CPU

CPU является ведущим. Он имеет свою операционную систему (планировщик или монитор) и выполняет основную программу.

CPU имеет доступ к следующим ресурсам DSP:

- памяти данных;
- регистру управления и состояния DCSR;
- программному счетчику PC;

- регистру адреса останова SAR;
- памяти программ;
- архитектурным регистрам.

Обмен данными с этими ресурсами выполняется по командам Load, Store. Память DSP и его регистры для CPU являются словными, то есть состояние двух младших разрядов адреса является безразличным.

При штатной работе доступ к архитектурным регистрам DSP, как правило, не используется, а применяется только для его диагностики или для отладки программного обеспечения.

DSP выдает следующие прерывания в CPU, которые поступают на регистр QSTR:

- программное;
- по переполнению стека;
- при выполнении команды STOP;
- при достижении адреса останова при исполнении программы до адреса останова или завершении требуемого числа шагов при пошаговом исполнении программы.

CPU в DSP прерываний не формирует.

CPU управляет работой DSP посредством передачи ему задания (макрокоманды) и его запуска (перевод из режима STOP в режим RUN). Данная процедура выполняется в следующей последовательности:

- CPU передает в память DSP данные и параметры их обработки. Эта операция может отсутствовать;
- CPU передает в программную память DSP программный код, который должен быть выполнен. Эта операция может отсутствовать;
- CPU передает в DSP адрес первой выполняемой команды посредством записи в программный счетчик. Эта операция может отсутствовать, например, если следующая макрокоманда DSP должна выполняться с его текущего состояния;
- CPU переводит DSP в состояние RUN посредством записи в его регистр управления и состояния DCSR.

#### 4.5.2 Функции DSP

DSP является ведомым. Он работает под управлением CPU и выполняет его макрокоманды (задания). Операционной системы и какого-либо монитора не имеет.

Для управления своей работой DSP имеет программно доступный регистр управления и состояния DCSR. Формат этого регистра приведен в главе 3.

DSP может находиться в состояниях STOP или RUN и работает в старт стоповом режиме. То есть, после выполнения очередного задания CPU он останавливается и переходит в режим STOP посредством выполнения одноименной команды. DSP из состояния STOP в состояние RUN может перейти:

- по команде CPU;
- по сигналам от каналов DMA MemCh.

DSP может выполнить запуск работы каналов DMA MemCh посредством записи 1 в соответствующие разряды регистра DCSR.

## 5. ИНТЕРВАЛЬНЫЙ ТАЙМЕР

### 5.1 Назначение

Интервальный таймер (ИТ), предназначен для выработки периодических прерываний на основе деления тактовой частоты CPU. Основные характеристики интервального таймера:

- Число разрядов основного делителя – 32;
- Число разрядов предделителя – 8;
- Программное управление стартом и остановкой таймера;
- Доступ ко всем регистрам обеспечивается в любой момент времени.

### 5.2 Структурная схема

Структурная схема интервального таймера приведена на Рисунок 5.1.

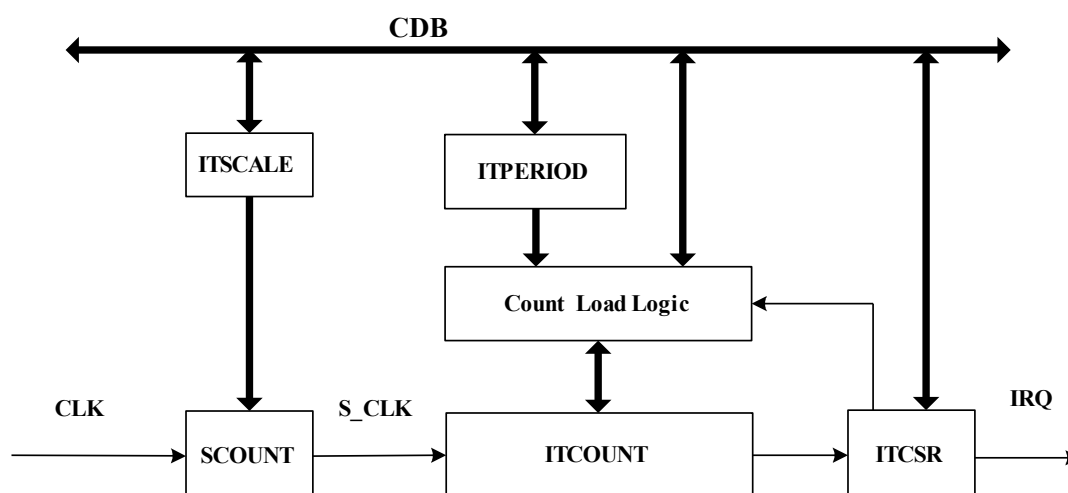


Рисунок 5.1. Структурная схема ИТ.

В состав интервального таймера входят следующие основные узлы:

- ITCSR - регистр управления и состояния;
- ITCOUNT - счетчик основного делителя;
- ITPERIOD - регистр периода основного делителя;
- ITSCALE - регистр предделителя;
- SCOUNT – счетчик предделителя;
- Count Load Logic - логика загрузки счетчика основного делителя.

На структурной схеме интервального таймера использованы следующие обозначения:

- CDB – шина данных CPU;

- CLK – тактовая частота работы CPU;
- S\_CLK – выходная частота предделителя;
- IRQ – запрос на прерывание от интервального таймера.

### 5.3 Регистры интервального таймера

Перечень программно-доступных регистров интервального таймера приведен в Таблица 5.1.

**Таблица 5.1. Перечень программно-доступных регистров интервального таймера.**

Условное обозначение регистра	Название регистра	Тип доступа	Исходное состояние
ITCSR[2:0]	Регистр управления и состояния	W/R	0
ITPERIOD[31:0]	Регистр периода	W/R	FFFF FFFF
ITCOUNT[31:0]	Регистр счетчика основного делителя частоты	W/R	0000_0000
ITSCALE[7:0]	Регистр предделителя частоты	W/R	0000

Формат регистра ITCSR приведен в Таблица 5.2.

**Таблица 5.2. Формат регистра ITCSR.**

Номер разряда	Условное обозначение	Описание
0	EN	Разрешение работы таймера: 0 – запрещение работы (неактивное состояние таймера); 1 – разрешение работы (активное состояние таймера).
1	INT	Признак срабатывания таймера. Состояние данного разряда транслируется в бит Timer регистра QSTR (на входе этого регистра он объединяется по логическому "или" с одноименными разрядами регистров управления и состояния таймеров WDT и RTT). Сбрасывается при записи нуля в этот разряд.
2	TICK	При записи 1 в этот разряд содержимое счетчика ITCOUNT и предделителя ITSCALE однократно декрементируются. Разряд предназначен только для целей тестирования. Разряд доступен только по записи

8-разрядный регистр ITSCALE используется для задания коэффициента предделения тактовой частоты CPU (CLK), которая поступает на вход счетчика SCOUNT.

32-разрядные регистр ITPERIOD используется для задания периода работы основного делителя.

32-разрядный счетчик основного делителя частоты ITCOUNT работает в режиме декремента. На вход этого счетчика поступает частота (S\_CLK) с выхода счетчика предделителя.

## 5.4 Программирование ИТ

Перед началом работы с интервальным таймером необходимо загрузить значение периода в регистр ITPERIOD и значение коэффициента предделения частоты в регистр ITSCALE.

Для активизации таймера необходимо в бит EN регистра ITCSR записать 1. В момент этой записи содержимое регистров ITSCALE и ITPERIOD переписывается в счетчики SCOUNT и ITCOUNT соответственно. После этого оба счетчика начинают работать в режиме декремента. При этом предделитель работает от частоты CLK, а счетчик ITCOUNT – от частоты S\_CLK, формируемой предделителем.

Когда оба счетчика SCOUNT и ITCOUNT достигают нулевого состояния, в регистре ITCSR устанавливается бит INT и формируется запрос на прерывание QSTR[29] (бит TIMER), а содержимое регистров ITSCALE и ITPERIOD опять переписывается в счетчики SCOUNT и ITCOUNT соответственно. Далее таймер работает аналогичным образом.

Запрос на прерывание формируется каждые  $\{(itperiod + 1) * (itscale + 1)\}$  тактов работы CPU, где *itperiod* и *itscale* – содержимое регистров ITPERIOD и ITSCALE соответственно.

При необходимости, в любой момент времени в ITCOUNT и ITPERIOD можно произвести запись новых данных и тем самым изменить значение обрабатываемого временного интервала.

## 6. ТАЙМЕР РЕАЛЬНОГО ВРЕМЕНИ

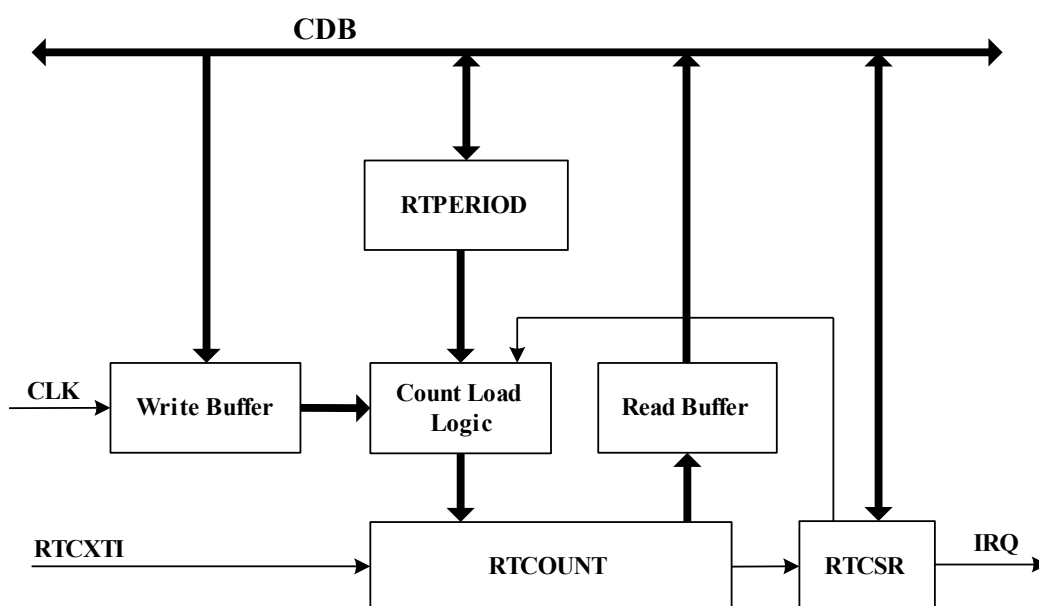
### 6.1 Назначение

Таймер реального времени (RTT) предназначен для выработки периодических прерываний на основе деления внешней тактовой частоты RTCXTI. Основные характеристики таймера реального времени:

- Число разрядов делителя – 32;
- Программное управление стартом и остановкой таймера;
- Доступ ко всем регистрам обеспечивается в любой момент времени.

### 6.2 Структурная схема RTT

Структурная схема RTT представлена на Рисунок 6.1.



**Рисунок 6.1. Структурная схема RTT.**

В состав таймера реального времени входят следующие основные узлы:

- RTCSR - регистр управления и состояния;
- RTCOUNT - счетчик основного делителя;
- RTPERIOD - регистр периода основного делителя;
- Count Load Logic - логика загрузки счетчика основного делителя;
- Write Buffer – буфер записи;
- Read Buffer – буфер чтения.

На структурной схеме интервального таймера использованы следующие обозначения:

- CDB – шина данных CPU;
- CLK – тактовая частота работы CPU;
- RTCXTI – внешняя тактовая частота;
- IRQ – запрос на прерывание от таймера реального времени.

На вход таймера реального времени поступает внешняя тактовая частота RTCXTI. Для правильной работы RTT должно выполняться соотношение:  $f_{RTCXTI} \leq \frac{f_{CLK}}{7}$ , где  $f_{RTCXTI}$  и  $f_{CLK}$  значения частот RTCXTI и CLK соответственно. Как правило, RTCXTI имеет частоту 32,768 кГц.

### 6.3 Описание регистров таймера реального времени

В Таблица 6.1. приведен перечень программно-доступных регистров RTT.

**Таблица 6.1. Перечень регистров RTT**

Условное обозначение регистра	Название регистра	Тип доступа	Исходное состояние
RTCSR[1:0]	Регистр управления и состояния	W/R	0
RTPERIOD[31:0]	Регистр периода	W/R	0000 7FFF
RTCOUNT[31:0]	Регистр счетчика делителя	W/R	0000 0000

Формат регистра RTCSR приведен в Таблица 6.2.

**Таблица 6.2. Формат регистра RTCSR.**

Номер разряда	Условное обозначение	Описание
0	EN	Разрешение работы таймера: 0 – запрещение работы (неактивное состояние таймера); 1 – разрешение работы (активное состояние таймера).
1	INT	Признак срабатывания таймера. Состояние данного разряда транслируется в бит Timer регистра QSTR (на входе этого регистра он объединяется по логическому «или» с одноименными разрядами регистров управления и состояния таймеров WDT и IT). Сбрасывается при записи нуля в этот разряд.
2	TICK	При записи 1 в этот разряд содержимое счетчика RTCOUNT однократно декрементируется. Разряд предназначен только для целей тестирования. Разряд доступен только по записи

32-разрядный регистр RTPERIOD используется для задания периода работы таймера. Если RTPERIOD = 0000\_7FFF, а частота RTCXTI = 32,768 кГц, то таймер реального времени формирует прерывание каждую секунду.

32-разрядный счетчик RTCOUNT работает в режиме декремента от частоты RTCXTI.



## 6.4 Программирование RTT

Перед началом работы с таймером необходимо загрузить данные в регистр RTPERIOD.

Для активизации таймера необходимо в бит EN регистра RTCSR записать 1. В момент этой записи содержимое регистра RTPERIOD переписывается в счетчик RTCOUNT, который начинает работать в режиме декремента. Когда счетчик RTCOUNT достигнет нулевого состояния, в регистре RTCSR устанавливается бит INT и формируется запрос на прерывание QSTR[29] (бит TIMER), а содержимое регистра RTPERIOD опять переписывается в счетчик RTCOUNT. Далее таймер работает аналогичным образом.

При необходимости, в любой момент времени в RTPERIOD и RTCOUNT можно произвести запись новых данных и тем самым изменить значение, обрабатываемого временного интервала.

Следует отметить, что при записи в RTCOUNT, обновление его содержимого происходит с задержкой, равной периоду RTCXTI.

## 7. СТОРОЖЕВОЙ ТАЙМЕР

### 7.1 Назначение

Сторожевой таймер (WDT) предназначен для:

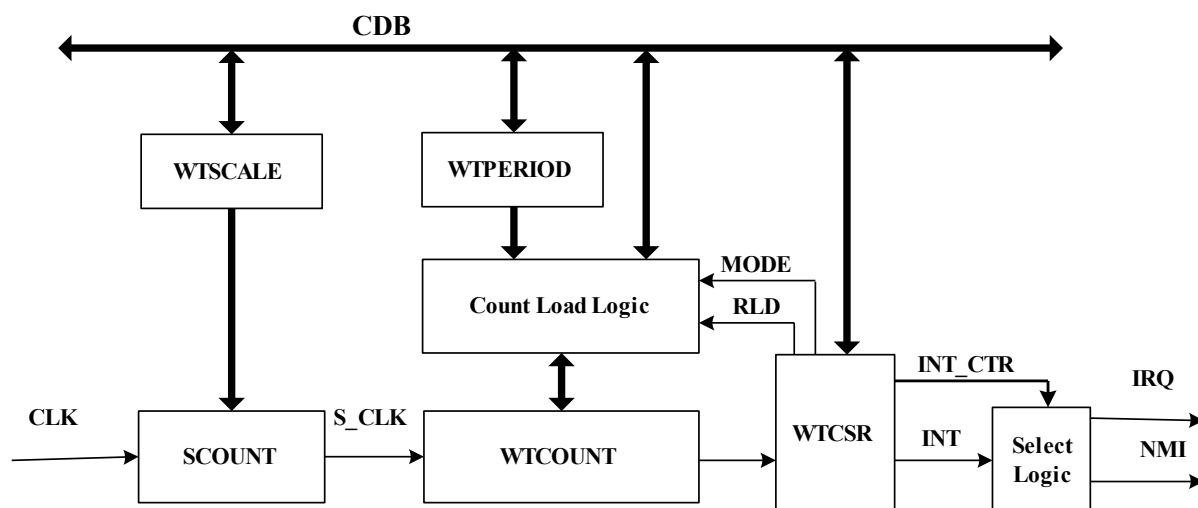
- вывода системы из зависания, если программное обеспечение заиклилось и не формирует соответствующих управляющих воздействий;
- выработки прерываний на основе деления тактовой частоты CPU.

Основные характеристики таймера:

- число разрядов основного делителя – 32;
- число разрядов предделителя – 8;
- программное управление стартом и остановкой таймера;
- два режима работы: режим сторожевого таймера (WDM) и режим интервального таймера (ITM);
- два режима отработки временных интервалов: однократный и периодический;
- доступ ко всем регистрам обеспечивается в любой момент времени.

### 7.2 Структурная схема

Структурная схема сторожевого таймера приведена на **Рисунок 7.1**.



**Рисунок 7.1. Структурная схема сторожевого таймера.**

В состав сторожевого таймера входят следующие основные узлы:

- WTCSR - регистр управления и состояния;
- WTCOUNT - счетчик основного делителя;
- WTPERIOD - регистр периода основного делителя;
- WTSCALE - регистр предделителя;

- SCOUNT – счетчик предделителя;
- Count Load Logic - логика загрузки счетчика основного делителя.

На структурной схеме интервального таймера использованы следующие обозначения:

- CDB – шина данных CPU;
- CLK – тактовая частота работы CPU;
- S\_CLK – выходная частота предделителя;
- IRQ – запрос на прерывание от интервального таймера;
- NMI – немаскируемое прерывание.

### 7.3 Описание регистров WDT

В таблице 7.1 приведен перечень программно-доступных регистров WDT.

**Таблица 7.1. Перечень программно-доступных регистров WDT.**

Условное обозначение регистра	Название регистра	Тип доступа	Исходное состояние
<b>WTCSR[14:0]</b>	Регистр управления и состояния	W/R	0000
<b>WTPERIOD[31:0]</b>	Регистр периода	W/R – в неактивном состоянии; R – в активном состоянии.	FFFF_FFFF
<b>WTCOUNT[31:0]</b>	Регистр счетчика основного делителя частоты	W/R – в неактивном состоянии; R – в активном состоянии.	0000_0000
<b>WTSCALE[15:0]</b>	Регистр предделителя частоты	W/R – в неактивном состоянии; R – в активном состоянии.	0000

8-разрядный регистр WTSCALE используется для задания коэффициента предделения тактовой частоты CPU (CLK), которая поступает на вход счетчика SCOUNT.

32-разрядные регистр WTPERIOD используется для задания периода работы основного делителя.

32-разрядный счетчик основного делителя частоты WTCOUNT работает в режиме декремента. На вход этого счетчика поступает частота S\_CLK с выхода счетчика предделителя.

Формат регистра WTCSR приведен в таблице 7.2.

Таблица 7.2. Формат регистра WTCSR

Номер разряда	Условное обозначение	Описание
7: 0	KEY	<p>Поле для записи ключей. Запись в это поле последовательности кодов A0 (ключ KEY1) и F5 (ключ KEY2) приводит к переключению таймера из режима сторожевого таймера (WDM) в режим интервального таймера (ITM). Поле доступно по чтению и записи. Поле доступно по записи только в режиме WDM: когда EN=1 или когда таймер находится в состоянии Timeout. Сбрасывается в ноль при переводе таймера из режима ITM в режим WDM. Значение в исходном состоянии – 0.</p>
8	EN	<p>Разрешение работы таймера: 0 – запрещение работы (неактивное состояние таймера); 1 – разрешение работы (активное состояние таймера). Доступен по чтению и записи. Запись нуля в этот бит при работе таймера в режиме WDM не имеет эффекта. Значение в исходном состоянии – 0.</p>
9	INT	<p>Признак срабатывания таймера. В зависимости от содержимого поля INT_CTR состояние данного разряда транслируется или в бит Timer регистра QSTR (на входе этого регистра он объединяется по логическому «или» с одноименными разрядами регистров управления и состояния таймеров RTT и IT), или в немаскируемое прерывание (NMI). Сбрасывается при записи нуля в этот разряд, а также при переводе таймера из режима ITM в режим WDM. Доступен по чтению и записи в режиме ITM и только по чтению в режиме WDM. Значение в исходном состоянии – 0.</p>
10	MODE	<p>Режим работы таймера: 0 – режим сторожевого таймера (WDM); 1 – режим обычного таймера (ITM). Доступен по чтению и записи при EN=0 и только по чтению при EN=1. Значение в исходном состоянии – 0.</p>
11	RLD	<p>Бит управления перезагрузкой SCOUNT и WTCOUNT при работе в режиме ITM: 0 – таймер однократно обрабатывает временной интервал и останавливается; 1 – таймер обрабатывает заданный временной интервал периодически. После обработки очередного временного интервала содержимое WTSCALE и WTPERIOD загружается в SCOUNT и WTCOUNT соответственно. Доступен по чтению и записи при EN=0 и только по чтению при EN=1. Значение в исходном состоянии – 0.</p>
13: 12	INT_CTR	Управления типом прерывания, которое формируется

Номер разряда	Условное обозначение	Описание
		таймером WDT: 00 – прерывание не формируется; 01 – обычное прерывание (QSTR[29]). Как правило, используется в режиме ITM; 10 – немаскируемое прерывание (NMI). Как правило, используется в режиме WDM. 11 – прерывание не формируется. Формируется внешний сигнал WDT (см. табл. 15.2). Поле доступно по чтению и записи при EN=0 и только по чтению при EN=1. Значение в исходном состоянии – 0.
14	TICK	При записи 1 в этот разряд содержимое счетчика WTCOUNT и предделителя WTSCALE однократно декрементируются. Разряд предназначен только для целей тестирования. Разряд доступен только по записи

## 7.4 Программирование WDT

Диаграмма состояний WDT приведена на рис 7.2.

В исходном состоянии WDT находится в режиме сторожевого таймера. Для перевода его в режим интервального таймера необходимо записать 1 в бит MODE регистра WTCSR. Следует отметить, что смена режима работы таймера посредством записи в бит MODE возможна, если таймер не активен (EN=0).

Перед началом работы с таймером WDT необходимо загрузить значение периода в регистр WTPERIOD и значение коэффициента предделения частоты в регистр WTSCALE.

Для активизации таймера необходимо в бит EN регистра WTCSR записать 1. В момент этой записи содержимое регистров WTSCALE и WTPERIOD переписывается в счетчики SCOUNT и WTCOUNT соответственно. После этого оба счетчика начинают работать в режиме декремента. При этом предделитель работает от частоты CLK, а счетчик WTCOUNT – от частоты S\_CLK, формируемой предделителем.

После активизации таймера, WTCOUNT, WTPERIOD, WTSCALE, а также поля INT\_CTR, MODE, RLD регистра WTCSR, становятся не доступными по записи.

Сторожевой таймер в режиме WDM необходимо периодически обслуживать. То есть, если он был активизирован в режиме WDM, то для того, чтобы не возникло состояния Timeout необходимо периодически выполнять следующую последовательность действий:

- переключить таймер из режима WDM в режим ITM посредством последовательной записи в поле KEY регистра WTCSR кодов A0 и F5;
- остановить таймер посредством записи 0 в бит EN регистра WTCSR;
- установить MODE=0;

Если вслед за значением A0 в поле KEY будет записано значение  $\neq$  F5, то таймер перейдет в состояние Timeout.

Если после активизации таймера в режиме WDM, он не будет переведен в режим ITM, то, когда оба счетчика SCOUNT и WTCOUNT достигнут нулевого значения, таймер перейдет в состояние Timeout.

В состоянии Timeout таймер формирует признак INT и останавливается, а запись в какой-либо из его регистров блокируется. Для вывода WDT из состояния Timeout необходимо его переключить в режим ITM посредством последовательной записи в поле KEY регистра WTCSR кодов A0 и F5.

При переключении таймера из неактивного состояния в режиме ITM в режим WDM путем записи 0 в поле MODE регистра WTCSR происходит обнуление полей KEY и INT.

При работе таймера в режиме ITM при RLD=0 он однократно обрабатывает заданный временной интервал, устанавливает INT=1 и останавливается (когда оба счетчика SCOUNT и WTCOUNT достигают нулевого состояния). Если RLD=1, то каждый раз после достижения счетчиками нулевого состояния и установки INT=1, происходит перезагрузка значений периода и коэффициента предделения частоты. То есть, таймер обрабатывает заданный временной интервал периодически до тех пор, пока он не будет остановлен.

Запрос на прерывание формируется каждые  $\{(wtperiod + 1) * (wt scale + 1)\}$  тактов работы CPU, где wtperiod и wt scale – содержимое регистров WTPERIOD и WTSCALE соответственно.

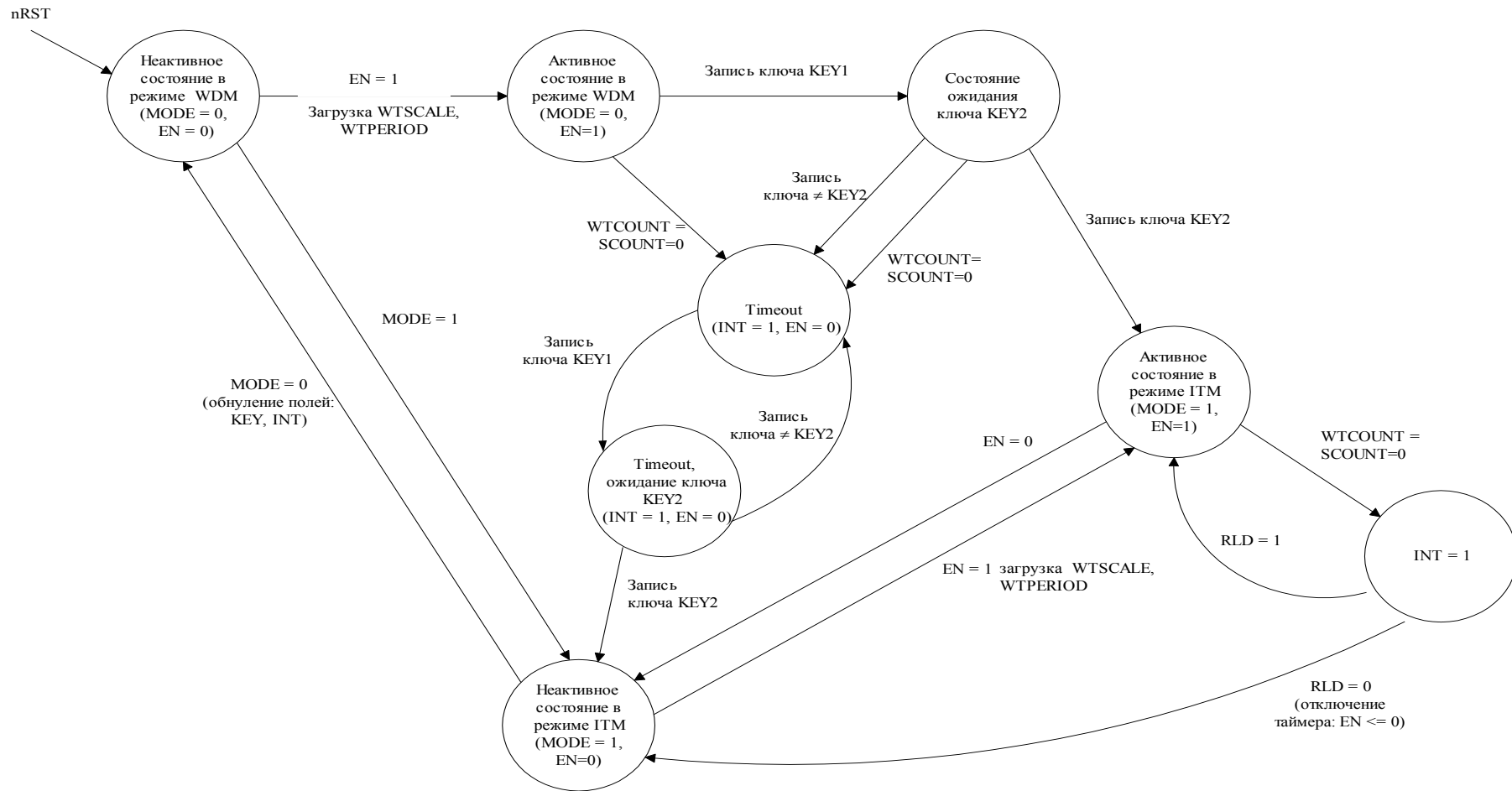


Рисунок 7.2. Диаграмма состояний WDT.

## 8. КОНТРОЛЛЕР ПРЯМОГО ДОСТУПА В ПАМЯТЬ (DMA)

### 8.1 Общие положения

#### 8.1.1 Типы каналов

Контроллер DMA имеет 12 каналов следующих типов:

- каналы обмена данными между последовательными портами и внутренней (CRAM, PMEM, XMEM, YRAM) или внешней памятью;
- каналы обмена данными между линковыми портами и внутренней (CRAM, PMEM, XMEM, YRAM) или внешней памятью;
- каналы обмена данными между внутренней памятью (CRAM, PMEM, XMEM, YRAM) и внешней памятью.

Перечень каналов DMA 1892BM3T приведен в Таблица 8.1.

**Таблица 8.1. Каналы DMA**

Условное обозначение канала	Назначение канала	Приоритет каналов DMA и CPU
SportRxCh0	Прием данных из буфера SRx порта SPORT0 во внутреннюю или внешнюю память	0
SportRxCh1	Прием данных из буфера SRx порта SPORT1 во внутреннюю или внешнюю память	1
SportTxCh0	Передача данных из внутренней или внешней памяти в буфер STx порта SPORT0	2
SportTxCh1	Передача данных из внутренней или внешней памяти в буфер STx порта SPORT1	3
CPU	-	4
LportCh3 – LportCh0	Обмен данными между буферами данных линковых портов и памятью (внешней или внутренней)	8-5
MemCh3 – MemCh0	Обмен данными между внешней памятью и внутренней памятью.	12-9 (изменяется циклически)

Если при работе DMA изменяется программный код в памяти, то когерентность кэш программ CPU (ICACHE) аппаратно не обеспечивается. В этом случае для обеспечения когерентности используется бит FLUSH в регистре CSR.

#### 8.1.2 Приоритет каналов DMA и CPU

CPU по шине CDB без конфликтов с DMA обменивается с памятью CRAM, с системными регистрами CSR, MASKR, QSTR, и с регистрами таймеров IT, WDT, RTT. CPU без конфликтов с DMA обменивается с регистрами MPORT и внешней памятью, если нет DMA передач.



При передаче данных каналы DMA конфликтуют между собой всегда. Каналы DMA конфликтуют с CPU, если CPU и DMA одновременно запрашивают шину DDB.

Приоритет каналов DMA указан в правой колонке Таблица 8.1 (0 – наивысший приоритет). Если несколько каналов DMA одновременно запрашивают шину DDB, то ее занимает канал, приоритет которого самый высокий.

Взаимный приоритет каналов MemCh изменяется циклически следующим образом. Исходное распределение приоритетов между каналами MemCh (в порядке их убывания): MemCh0, MemCh1, MemCh2, MemCh3. Далее, после каждой DMA передачи распределение приоритетов изменяется циклическим сдвигом влево, таким образом, что приоритет канала, который выполнил DMA передачу, становится самым низким. Например, если после исходного состояния передал канал MemCh0, то приоритеты распределятся следующим образом: MemCh1, MemCh2, MemCh3, MemCh0. Далее, если передал канал MemCh3, то приоритеты распределятся следующим образом: MemCh0, MemCh1, MemCh2, MemCh3 и т.д.

### 8.1.3 Темп передачи

DMA передача одного 32-разрядного слова данных между внутренней памятью и SPORT, LPORT выполняется за время  $TCLK$  (период частоты CLK).

Время DMA передачи одного 32-разрядного слова данных между внешней памятью и SPORT, LPORT или внутренней памятью, равно:

- для асинхронной внешней памяти –  $2 * TCLK + TCLK * N$ , где  $N$  – число тактов ожидания (код в поле WS регистров CCON, увеличенный на 1).
- для синхронной внешней памяти -  $TCLK$ .

Каналы последовательных и линковых портов за один цикл занятия шины DDB передают одно слово данных. После передачи этого слова шина DDB данным каналом освобождается.

Каналы MemCh за один цикл занятия шины DDB передают пачку данных. Размер пачки задается полем  $WN$  в регистре CSR соответствующего канала DMA и определяется системными требованиями по передаче данных. Если после передачи пачки данных нет запросов от других каналов DMA или CPU, то данный канал без перерыва начинает передавать следующую пачку данных и т.д.

CPU за один цикл занятия шины DDB выполняет одну из следующих операций (после этого шина освобождается):

- чтение одного слова данных по команде Load;
- запись одного слова данных по команде Store;
- выборка команды из внешней памяти;
- процедура Refill (загрузка из внешней памяти в ICACHE 4 команды), если адрес команды CACHED, а ее нет в ICACHE (ситуация MISS).

### 8.1.4 Регистры DMA

Для управления работой каждого канала DMA имеются следующие регистры:

- регистр управления и состояния (CSR);
- набор регистров индекса (адрес памяти) и смещения (IOR, IR, OR, Y);

- регистр начального адреса блока параметров DMA передачи (CP).

Следует отметить, что индексные регистры IR и IOR содержат физические адреса памяти.

Для эффективной передачи двумерных массивов (матриц  $W[m;n]$ ) все каналы DMA используют регистр Y, в котором хранятся смещение и число строк в направлении Y.

Разные типы каналов содержат разный набор регистров.

Исходное состояние регистров CSR: разряды 15:0 – нули, а состояние разрядов 31:16 не определено. Исходное состояние остальных регистров не определено.

Индексный регистр содержит адрес 32-разрядного слова в памяти (младшие два разряда адреса должны быть равны нулю).

Регистр смещения задает приращение адреса. Содержимое регистра смещения, аппаратно умноженное на 4, прибавляется к индексу после передачи каждого слова данных. Если по каналам MemCh выполняется обмен данными с SDRAM, то смещение прибавляется после передачи каждой пачки 32-разрядных слов, которая передается в режиме “Burst”. То есть, при обмене данными с SDRAM по каналам MemCh, величина смещения в регистре OR должна быть не меньше, чем размер пачки, указанный в поле WN регистра CSR (WN=0, OR>=1; WN=1, OR>=2 и т.д.).

### 8.1.5 Прерывания DMA

Канал DMA формирует прерывание (при условии, если установлен соответствующий бит в регистре MASKR и бит IM[7] в регистре STATUS RISC-ядра):

- при единичном состоянии бита DONE;
- при единичном состоянии битов END и IM.

Обнуление битов DONE и END (и снятие соответствующего прерывания) выполняется посредством чтения содержимого регистра CSR. Обнуление бита DONE может быть выполнено также записью нуля в него.

## 8.2 Процедура самоинициализации

Все каналы DMA могут выполнять процедуру самоинициализации (выполнение цепочки передач DMA).

Для выполнения самоинициализации в каналах имеется 16-разрядный регистр CP, в котором хранится начальный адрес блока параметров очередного DMA обмена. Эти параметры при самоинициализации аппаратно загружаются в соответствующие регистры канала DMA. Процедура этой загрузки ничем не отличается от обычного DMA обмена. Блок параметров может размещаться только во внутренней памяти MEM.

Блоки параметров, размещаемых в памяти, имеют следующую структуру (в порядке возрастания адресов):

- каналы последовательных портов и линковых портов – IR, OR, Y, CP, CSR;
- каналы MemCh – IOR, IR, OR, Y, CP, CSR.

Параметры, соответствующие 16-разрядным регистрам, размещаются в младших разрядах памяти. В слове памяти, соответствующем регистру CSR должно быть: RUN=1, DONE=0. Если необходимо продолжить цепочку команд, то необходимо указать CHEN=1.

Для запуска работы канала DMA в режиме с самоинициализацией необходимо в регистр CP записать адрес первого блока параметров DMA передачи. При этом 31 разряд записываемых данных должен содержать 1 (признак пуска самоинициализации). В результате этого, соответствующий канал загрузит в свои регистры параметры DMA передачи и начнет обмен данными.

После окончания передачи данного блока данных устанавливается в единичное состояние бит END в регистре CSR и выдается прерывание, если бит IM = 1. После этого канал проверяет состояние бита CHEN. Если он равен 1, то будет загружен следующий блок параметров DMA передачи и т.д. В противном случае цепочка DMA обменов закончится и в регистре CSR бит DONE установится в единичное состояние.

При необходимости каналы DMA могут инициализироваться программно. Для этого RISC должен загрузить все необходимые регистры индекса и смещения, а затем регистр CSR. При загрузке регистра CSR бит RUN необходимо установить в единичное состояние. Следует отметить, что бит RUN может быть использован для приостановки канала DMA. Для этого в любой момент времени в него необходимо записать 0.

Следует иметь в виду, что если биты END или DONE имеют единичное состояние, то после считывания содержимого регистра CSR эти биты автоматически обнуляются.

### 8.3 Каналы DMA последовательных портов

Для обслуживания последовательных портов имеется 4 канала DMA: SportTxCh0, SportRxCh0, SportTxCh1, SportRxCh1 (раздельно на прием и передачу).

Формат регистров управления и состояния CSR\_SpRx0, CSR\_SpTx0, CSR\_SpTx1, CSR\_SpRx1 каналов DMA последовательных портов приведен в Таблица 8.2.

**Таблица 8.2. Формат регистров управления и состояния DMA последовательных портов**

Номер разряда	Условное обозначение	Назначение
0	RUN	Состояние работы канала DMA: 0 – состояние останова; 1 – состояние обмена данными.
1-8	-	Резерв
9	2D	Режим модификации адреса памяти: 0 – одномерный режим; 1 – двухмерный режим.
11,10	-	Резерв
12	CHEN	Признак разрешения самоинициализации (выполнения цепочки DMA передач)
13	IM	Маска прерывания при окончании передачи блока данных: 0 – прерывание запрещено; 1 – прерывание разрешено.
14	END	Признак окончания передачи блока данных

Номер разряда	Условное обозначение	Назначение
15	DONE	Признак завершения передачи цепочки блоков данных. Аппаратно устанавливается в 1 после завершения передачи данных (при CHEN=0), при этом бит RUN сбрасывается. Доступен по записи и чтению. Состояние данного бита дублируется в соответствующий бит регистра QSTR
31:16	WCX	Счетчик слов при одномерной адресации. Счетчик числа слов в строке при двухмерной адресации.

Для задания адреса памяти (внутренней или внешней) каналы DMA последовательных портов содержат два регистра:

32-разрядный индексный регистр памяти IR;

16-разрядный регистр смещения памяти OR.

16-разрядный регистр OR содержит код смещения (приращения) памяти в 32-разрядных словах для перехода к следующему элементу массива. Он используется всегда. При адресации в двухмерном режиме он указывает приращение в направлении X. Приращение рассматривается как число со знаком в диапазоне от -32768 до +32767.

При работе каналов последовательных портов память (внутренняя или внешняя) может адресоваться в двухмерном режиме. Для этого имеется 32-разрядный регистр Y, формат которого приведен в Таблица 8.3.

**Таблица 8.3. Формат регистра Y**

Номер разряда	Условное Обозначение	Назначение
15:0	OY	Смещение (приращение) адреса памяти в 32-разрядных словах по направлению Y. Используется только при двухмерной адресации.
31:16	WCY	Число строк по Y направлению. Используется только при двухмерной адресации.

При двухмерном режиме адресации поле WCX регистра CSR содержит число слов в строке (X направление), а поле WCY регистра Y содержит число строк (Y направление). Пересылка каждого слова данных осуществляется по индексному регистру IR с его последующей инкрементацией на величину, соответствующую содержимому регистра смещения OR или поля OY регистра Y. Двухмерная адресация выполняется следующим образом:

1. Содержимое счетчика WCX сохраняется в буферном регистре;
2. 1 цикл. Индексный регистр внешней памяти модифицируется с использованием смещения OR. Счетчик WCX декрементируется. Если он равен 0, то переход ко второму циклу.
3. 2 цикл. Состояние счетчика WCX восстанавливается из буферного регистра. Индексный регистр внешней памяти модифицируется с использованием

смещения OY. Счетчик WCY декрементируется. Если он не равен 0, то переход к первому циклу. Если он равен 0, то работа канала завершается.

Функционально двухмерная адресация эквивалентна следующему двойному циклу, написанному на языке C:

```
for ( y = 0; y <= WCY; y++ ) {
    for ( x = 0; x < WCX; x++ ) { пересылка слова данных по адресу IR
        IR = IR + OR;           };
    пересылка слова данных по адресу IR
    IR = IR + OY;
};
```

Общее количество пересылок равно  $(WCX+1)*(WCY+1)$ .

## 8.4 Каналы DMA линковых портов

Для обслуживания линковых портов имеется 4 канала DMA: LportCh0, LportCh1, LportCh2, LportCh3.

Формат регистров управления и состояния CSR\_Lp0, CSR\_Lp1, CSR\_Lp2, CSR\_Lp3 каналов DMA линковых портов приведен в Таблица 8.4.

**Таблица 8.4. Формат регистров управления и состояния DMA линковых портов**

Номер разряда	Условное Обозначение	Назначение
0	RUN	Состояние работы канала DMA: 0 – состояние останова; 1 – состояние обмена данными.
8:1	-	Резерв
9	2D	Режим модификации адреса памяти: 0 – одномерный режим; 1 – двухмерный режим.
11:10	-	Резерв
12	CHEN	Признак разрешения самоинициализации (выполнения цепочки DMA передач)
13	IM	Маска прерывания при окончании передачи блока данных: 0 – прерывание запрещено; 1 – прерывание разрешено.
14	END	Признак окончания передачи блока данных

Номер разряда	Условное Обозначение	Назначение
15	DONE	Признак завершения передачи цепочки блоков данных. Аппаратно устанавливается в 1 после завершения передачи данных (при CHEN=0), при этом бит RUN сбрасывается. Доступен по записи и чтению. Состояние данного бита дублируется в соответствующий бит регистра QSTR
31:16	WCX	Счетчик слов при одномерной адресации. Счетчик числа слов в строке при двухмерной адресации.

Для задания адреса памяти (внутренней или внешней) каналы DMA линковых портов содержат два регистра:

32-разрядный индексный регистр памяти IR;

16-разрядный регистр смещения памяти OR.

16-разрядный регистр OR\_MEM содержит код смещения памяти в 32-разрядных словах. Он используется всегда. При адресации в двухмерном режиме он указывает смещение в направлении X. Смещение рассматривается как число со знаком в диапазоне от -32768 до +32767.

При работе каналов LportCh внешняя память может адресоваться в двухмерном режиме аналогично каналам последовательных портов.

## 8.5 Каналы обмена данными между внутренней и внешней памятью

Четыре канала MemCh0:MemCh3 обеспечивают обмен данными между внутренней памятью 1892BM3T (CRAM, PRAM, XRAM, YRAM) и внешней памятью.

Формат регистров состояния и управления этих каналов приведен в Таблица 8.5.

**Таблица 8.5. Формат регистра управления и состояния каналов MemCh**

Номер разряда	Условное обозначение	Назначение
0	RUN	Состояние работы канала DMA: 0 – состояние останова; 1 – состояние обмена данными.
1	DIR	Направление обмена данными: 0 – внутренняя память => внешняя память; 1 – внутренняя память <= внешняя память.
5:2	WN	Число слов данных (пачка), которое передается за одно предоставление прямого доступа: 0 – 1 слово, F – 16

Номер разряда	Условное обозначение	Назначение
		слов. Посредством этого параметра можно плавно изменять приоритет каналов DMA относительно RISC и относительно друг друга.
6	-	Резерв
7	START_DSP	Разрешение запуска работы DSP-ядра (перевод из состояния STOP в состояние RUN) после завершения передачи цепочки блоков данных в момент установки бита DONE: 0 – запуск запрещен; 1 – запуск разрешен.
8	MODE	Режим модификации адреса внутренней памяти: 0 – линейный режим; 1 – режим с реверсивным переносом.
9	2D	Режим модификации адреса внешней памяти: 0 – одномерный режим; 1 – двухмерный режим.
10	MASK	Маска внешнего запроса прямого доступа nDMAR: 0 – запрос запрещен; 1 – запрос разрешен. Если разряд равен нулю, то канал работает только под управлением бита RUN. Если разряд равен 1, то для инициализации канала необходимо также наличие запроса nDMAR (низкий уровень).
11	FLYBY	Признак выполнения обмена между внешней памятью и внешним устройством.
12	CHEN	Признак разрешения самоинициализации (выполнения цепочки DMA передач)
13	IM	Маска прерывания при окончании передачи блока данных: 0 – прерывание запрещено; 1 – прерывание разрешено.
14	END	Признак окончания передачи блока данных

Номер разряда	Условное обозначение	Назначение
15	DONE	Признак завершения передачи цепочки блоков данных. Аппаратно устанавливается в 1 после завершения передачи цепочки блоков данных (при CHEN=0), при этом бит RUN сбрасывается. Доступен по записи и чтению. Состояние данного бита дублируется в соответствующий бит регистра QSTR
31:16	WCX	Счетчик слов при одномерной адресации. Счетчик числа слов в строке при двухмерной адресации.

Следует иметь в виду, что при обмене с внешней памятью типа SDRAM в поле WN допускается указывать только числа 0, 1, 3, 7, 15. При этом начальный адрес массива, предназначенного для передачи при помощи DMA, должен быть кратен WN+1. В противном случае обмен данными будет произведен неправильно.

Состоянием разряда 0 регистра CSR можно управлять, используя адрес псевдорегистра Run. При этом остальные разряды этого регистра не изменяются. Эта процедура может быть использована для временной приостановки канала DMA.

Для задания адресов обмена данными каналы MemCh содержат три регистра:

- 32-разрядный регистр индекса и смещения адреса внутренней памяти IOR;
- 32-разрядный индексный регистр внешней памяти IR;
- 16-разрядный регистр смещения внешней памяти OR.

Формат регистра индекса и смещения IOR приведен в Таблица 8.6.

**Таблица 8.6. Формат регистра индекса и смещения каналов MemCh**

Номер разряда	Условное обозначение	Назначение
23:0	ADDR	Адрес внутренней памяти
31:24	OFFSET	Смещение (приращение) адреса внутренней памяти в 32-разрядных словах после передачи каждого слова данных

Смещение, задаваемое полем OFFSET, имеет диапазон от -128 до +127.

При инверсном режиме модификации адреса внутренней памяти смещение, задаваемое полем OFFSET, имеет диапазон от 0 до 255.

Поле ADDR в регистре IOR\_MEM указывает адрес внутренней памяти относительно базового адреса 1800\_0000.

16-разрядный регистр OR содержит код смещения внешней памяти в 32-разрядных словах. Он используется всегда. При адресации в двухмерном режиме он указывает смещение (приращение) в направлении X для перехода к следующему элементу строки. Смещение рассматривается как число со знаком в диапазоне от -32768 до +32767.



При работе каналов MemCh внешняя память может адресоваться в двухмерном режиме аналогично каналам последовательных портов.

Работа по внешним запросам.

Каждый канал MemCh[3:0] имеет внешний сигнал запроса передачи (nDMAR[3:0] соответственно), позволяющий организовывать эффективный обмен данными с внешними устройствами. Для работы по внешним запросам необходимо сначала настроить канал DMA (в том числе установить бит MASK регистра CSR\_MemCh в «1»), а затем активизировать внешнее устройство на формирование сигналов nDMAR.

По каждому переходу сигнала nDMAR из «1» в «0» DMA выполняет процедуру передачи одной пачки слов размером в соответствии с полем WN регистра CSR\_MemCh. Внешнее устройство может снять сигнал nDMAR в начале этой пачки или выдавать сигнал nDMAR в виде отрицательного импульса длительностью не менее 1,5 периодов системной тактовой частоты CLK (частота, на которой работает CPU).

Следует иметь в виду, что факт перехода сигнала nDMAR из «1» в «0» запоминается в DMA на триггере. Это триггер сбрасывается в момент представления данному каналу права на передачу в соответствии с его текущим приоритетом.

Необходимо также учитывать то, что факт перехода сигнала nDMAR из «1» в «0» запоминается в DMA при MASK=1 вне зависимости от состояния бита RUN. Если в процессе работы в DMA будет запомнен «лишний» факт перехода сигнала nDMAR из «1» в «0», то его можно сбросить, выполнив фиктивный DMA обмен.

Работа в режиме FLYBY.

Режим FLYBY используется для передачи данных между внешним устройством ввода-вывода и внешней памятью (как асинхронной, так и синхронной). Например, контроллер DMA может быть запрограммирован для передачи данных из аналого-цифрового преобразователя в SDRAM. Для выполнения передачи данных в этом режиме в соответствующем регистре CSR\_MemCh необходимо установить бит FLYBY.

При передаче данных в режиме FLYBY 1892BM3T отключается от шины данных, и активизирует внешнюю память и внешнее устройство ввода-вывода одновременно. Память управляется как обычно, а устройство ввода-вывода – при помощи сигналов nFLYBY (признак данного режима), nOE (активизация выходных формирователей устройства ввода-вывода) и nCSIO[3:0] (выбор устройства ввода-вывода).

Каждому каналу MemCh может соответствовать свое устройство ввода-вывода. Выбор устройства ввода-вывода осуществляется посредством сигналов nCSIO[3:0]. Каналу MemCh0 соответствует низкий уровень на выводе nCSIO[0], каналу MemCh1 соответствует низкий уровень на выводе nCSIO[1], и так далее.

В режиме FLYBY можно использовать сигналы nDMAR[3:0].

Временные диаграммы работы 1892BM3T в режиме FLYBY приведены в разделе 9.

## 9. ПОРТ ВНЕШНЕЙ ПАМЯТИ

### 9.1 Введение

Порт внешней памяти (MPORT) позволяет организовать интерфейс с широким набором устройств памяти и периферии, асинхронной и синхронной памятью. Внешний интерфейс порта обеспечивает подключение без дополнительной логики синхронной памяти типа SDRAM, а также асинхронной памяти, например EPROM и FLASH.

Порт памяти имеет следующие основные характеристики:

- Шина данных внешней памяти – 32 разряда;
- Шина адреса внешней памяти – 32 разряда;
- программное конфигурирование областей внешней памяти;
- интерфейс с синхронной памятью типа SDRAM;
- интерфейс с асинхронной памятью (SRAM, EPROM, FLASH, FIFO и т.д.);
- режим передачи данных Flyby;
- управление числом тактов ожидания при обмене с асинхронной памятью при помощи внешнего входного сигнала nACK и поля WS регистров CSCON.

### 9.2 Регистры порта внешней памяти

Перечень регистров порта внешней памяти приведен в Таблица 9.1.

**Таблица 9.1. Регистры порта внешней памяти**

Условное обозначение регистра	Название регистра
CSCON0	Регистр конфигурации 0.
CSCON1	Регистр конфигурации 1.
CSCON2	Регистр конфигурации 2.
CSCON3	Регистр конфигурации 3.
CSCON4	Регистр конфигурации 4.
SDRCON	Регистр конфигурации памяти типа SDRAM
CKE_CTR	Регистр управления состоянием вывода СКЕ

## 9.2.1 Регистр конфигурации CSCON0

Регистр CSCON0 предназначен для конфигурирования блока внешней памяти, подключаемого к выводу nCS[0].

Формат регистра приведен в Таблица 9.2.

**Таблица 9.2. Назначение разрядов регистра CSCON0**

Номер разряда	Условное обозначение	Описание
7-0	CSMASK	Разряды маски 31:24 при определении базового адреса блока. Младшие разряды маски равны нулю.
15-8	CSBA	Разряды 31:24 базового адреса блока. Младшие разряды базового адреса равны нулю.
19-16	WS	Число тактов ожидания при обращении к блоку памяти, если она является асинхронной
20	E	Разрешение формирования сигнала nCS[0]: 0 – запрещено; 1 – разрешено.
21	T	Тип памяти данного блока: 0 – асинхронная; 1 – синхронная.
22	AE	Разрешение ожидания сигнала nACK: 0 – запрещено; 1 – разрешено.
31-23	-	Резерв

Регистр CSCON0 доступен по записи и чтению. Исходное состояние регистра – 000F\_0000.

Сигнал nCS[0] формируется, если  $PHA \& CSMASK = CSBA$ , где PHA – 32-разрядный физический адрес. Минимальный размер блока – 16 Мбайт (при CSMASK = FF). Для увеличения размера блока в младшие разряды поля CSMASK необходимо записать соответствующее число нулей. Например, для блока размером в 128 Мбайт, разряды 2-0 CSMASK должны быть равны нулю.

Регистры CSCON должны быть сконфигурированы таким образом, чтобы определяемые ими области памяти занимали уникальные адресные пространства. Если эти области перекрываются, то результат обмена данными будет непредсказуем.

В поле WS этого регистра задается количество тактов ожидания в тактах частоты CLK, которое необходимо добавить в цикл шины при обращении к несинхронной внешней памяти. Во время аппаратного сброса процессора во все эти поля записывается значение F (15 тактов).

Управление длительностью циклов обмена с асинхронной памятью осуществляется сигналом nACK и полем тактов ожидания WS. Сигнал nACK позволяет вставлять такты ожидания непосредственно в начатый цикл обмена данными. Количество

вставленных тактов ожидания равно максимальному количеству дополнительных тактов, заданных полем WS и сигналом nACK.

## 9.2.2 Регистр конфигурации CSCON1

Регистр CSCON1 предназначен для конфигурирования блока памяти, подключаемого к выводу nCS[1]. Формат регистра приведен в Таблица 9.3.

**Таблица 9.3. Назначение разрядов регистра CSCON1**

Номер разряда	Условное обозначение	Описание
7-0	CSMASK	Разряды маски 31:24 при определении базового адреса блока. Младшие разряды маски равны нулю.
15-8	CSBA	Разряды 31:24 базового адреса блока. Младшие разряды базового адреса равны нулю.
19-16	WS	Число тактов ожидания при обращении к блоку памяти, если она является асинхронной
20	E	Разрешение формирования сигнала nCS[1]: 0 – запрещено; 1 – разрешено.
21	T	Тип памяти данного блока: 0 – асинхронная; 1 – синхронная.
22	AE	Разрешение ожидания сигнала nACK: 0 – запрещено; 1 – разрешено.
31-23	-	Резерв

Регистр CSCON1 доступен по записи и чтению. Исходное состояние регистра – 000F\_0000.

## 9.2.3 Регистр конфигурации CSCON2

Регистр CSCON2 предназначен для конфигурирования блока памяти, подключаемого к выводу nCS[2].

Формат регистра приведен в Таблица 9.4.

**Таблица 9.4. Назначение разрядов регистра CSCON2**

Номер разряда	Условное обозначение	Описание
7-0	CSMASK	Разряды маски 31:24 при определении базового адреса блока. Младшие разряды маски равны нулю.
15-8	CSBA	Разряды 31:24 базового адреса блока. Младшие разряды базового адреса равны нулю.

Номер разряда	Условное обозначение	Описание
19-16	WS	Число тактов ожидания при обращении к блоку памяти.
20	E	Разрешение формирования сигнала nCS[2]: 0 – запрещено; 1 – разрешено.
21	-	Резерв
22	AE	Разрешение ожидания сигнала nACK: 0 – запрещено; 1 – разрешено.
31-23	-	Резерв

Регистр CSCON2 доступен по записи и чтению. Исходное состояние регистра – 000F\_0000.

Память, подключаемая к выводу nCS[2], может быть только асинхронной.

## 9.2.4 Регистр конфигурации CSCON3

Регистр CSCON3 предназначен для конфигурирования блока памяти, подключаемого к выводу nCS[3].

Формат регистра приведен в Таблица 9.5.

**Таблица 9.5. Назначение разрядов регистра CSCON3**

Номер разряда	Условное обозначение	Описание
15-0	-	Резерв
19-16	WS	Число тактов ожидания при обращении к блоку памяти.
22-20	-	Резерв
23	BYTE	Разрядность памяти: 0 – 32 разряда; 1 – 8 разрядов. Исходное состояние данного разряда соответствует состоянию сигнала на входе BYTE микросхемы во время аппаратного сброса.
24	OVER	Признак того, что при обмене данными с асинхронной памятью блоков 0, 1, 2, 4 от нее не поступил сигнал nACK в течение 256 периодов частоты CLK.
31-25	-	Резерв

Регистр CSCON3 доступен по записи и чтению. Исходное состояние регистра – 000F\_0000, или 008F\_0000, в зависимости от состояния сигнала на выводе BYTE микросхемы.

Область памяти, определяемая регистром C5CON3, размещается в диапазоне физических адресов от 1C00\_0000 до 1FFF\_FFFF (64 Мбайт). Память данного блока может быть только асинхронной. Доступ к данному блоку памяти всегда разрешен. При обмене данными с этим блоком сигнал nACK безразличен.

Как правило, к выводу nCS[3] подключается блок памяти программ, реализованный на FLASH, PROM, EEPROM и т.д. Этот блок, в зависимости от состояния сигнала на выводе микросхемы BYTE может быть 8 – или 32 – разрядным.

8-разрядная память подключается к выводам D[7:0] микросхемы 1892BM3T. Шину адреса A[31:0] к этой памяти необходимо подключать, начиная с 0 разряда (к 32-разрядной памяти адрес подключается, начиная со 2 разряда). 32-разрядное слово из 8-разрядной памяти считывается байтами, причем сначала считывается младший байт. Запись данных в 8-разрядную память выполняется побайтно в соответствии с рекомендациями п. 9.4.2.

Признак OVER формируется, если в соответствующем регистре C5CON бит AE=1, а от памяти не поступил сигнал nACK в течение 256 тактов CLK. В этом случае операция обмена данными заканчивается обычным образом, за исключением того, что считываемые данные не определены, а записываемые данные теряются. Состояние бита OVER не влияет на выполнение последующих операций обмена данными.

## 9.2.5 Регистр конфигурации C5CON4

Регистр C5CON4 предназначен для конфигурирования внешней памяти, не вошедшей в области, определяемые регистрами C5CON3-C5CON0.

Формат регистра приведен в Таблица 9.6.

**Таблица 9.6. Назначение разрядов регистра C5CON4**

Номер разряда	Условное обозначение	Описание
15-0	-	Резерв
19-16	WS	Число тактов ожидания при обращении к памяти.
21:20	-	Резерв
22	AE	Разрешение ожидания сигнала nACK: 0 – запрещено; 1 – разрешено.
31-23	-	Резерв

Регистр C5CON4 доступен по записи и чтению. Исходное состояние регистра – 000F\_0000.

Данная область памяти может быть только асинхронной. Доступ к ней всегда разрешен.

## 9.2.6 Регистр управления работой с памятью SDRAM

Формат регистра приведен в Таблица 9.7. Исходное состояние – нули.

**Таблица 9.7. Формат регистра SDRCON**

Номер разряда	Условное обозначение	Описание
3:0	PS	Размер страницы микросхем SDRAM, подключенных к порту внешней памяти: 0 – 512; 1 – 1024; 2 – 2048; 3 – 4096. Число банков SDRAM – 4.
15:4	RFR	Период регенерации SDRAM в тактах частоты CLK
18:16	BL	Длина burst (двоичный код): 000 – 1; 001 – 2; 010 – 4; 011 – 8; 100:110 – резерв; 111 – Full Page.
19	WBM	Режим записи: 0 – Программируемая длина burst; 1 – Одиночная запись.
20	CL	Задержка чтения (CAS latency): 0 – 2; 1 – 3.
30:21	-	Резерв
31	INIT	При выполнении процедуры записи 1 в данный разряд выполняется процедура инициализации SDRAM. Время инициализации – не более 2 мкс. В SDRAM устанавливаются следующие режимы работы: Burst Length – поле BL; Burst Type – последовательный; CAS latency – бит CL; Режим записи – бит WBM.

Регистр SDRCON доступен по записи и чтению. Исходное состояние регистра – 0. 31 разряд регистра SDRCON доступен только по записи, при чтении всегда 0.

Для работы со SDRAM ее необходимо инициализировать со следующими параметрами:

- PS (размер страницы) - в соответствии с параметрами SDRAM;
- RFR (период регенерации) – в соответствии с параметрами SDRAM. Например, при тактовой частоте CLK 100 МГц для обеспечения 8 192 цикловой регенерации за 64 мс необходимо в поле RFR записать код 30D, что соответствует 7, 81 мкс на строку;
- BL = 111 (Full page). Остальные значения используются только при тестировании микросхемы;

- WBM = 0 (программируемая длина burst);
- CL (задержка чтения) - в соответствии с параметрами SDRAM;

Выполнение инициализации SDRAM осуществляется посредством записи в регистр SDRCON соответствующего кода с единицей в 31 разряде. Следует отметить, что перед выполнением процедуры инициализации SDRAM необходимо сконфигурировать регистры CSCON0, CSCON1.

Для прекращения burst Full Page и тем самым задания реального числа передаваемых слов данных, используется команда «BURST TERMINATE», которая формируется портом внешней памяти аппаратно.

## 9.2.7 Регистр СКЕ\_СТR

Регистр СКЕ\_СТR предназначен для управления состоянием вывода СКЕ микросхемы.

Формат регистра приведен в Таблица 9.8.

**Таблица 9.8. Назначение разрядов регистра СКЕ\_СТR**

Номер разряда	Условное обозначение	Описание
0	СКЕ	Состояние вывода СКЕ микросхемы: 0 – низкий уровень; 1 – высокий уровень.
1-7	-	Резерв.
8	INIT_DONE	Признак окончания выполнения процедуры инициализации SDRAM: 0 – инициализация завершена; 1 – инициализация не проводилась.
31-9	-	Резерв.

Регистр СКЕ\_СТR доступен по записи и чтению. Исходное состояние регистра – 0000\_0101.

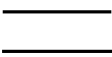


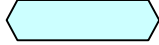

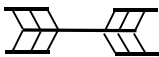
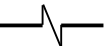


## 9.3 Временные диаграммы обмена данными

### 9.3.1 Общие положения

При описании временных диаграмм используются условные обозначения в соответствии с Таблица 9.9.

Таблица 9.9. Условные обозначения

Условное обозначение	Описание
	Стабильное значение
	Возможное значение
	область изменения из «0» в «1»
	область изменения из «1» в «0»
	Достоверное значение
	Для входов: Не воспринимается, допустимо любое переключение Для выходов: состояние не определено
	Переключение выхода из (в) высокоимпедансного (е) состояния (е) (центральная линия)
	Повторение сигнала в течение неопределенного времени
<b>T<sub>i</sub></b>	<i>i</i> = 1, 2, ... фаза обмена на временной диаграмме
<b>n</b>	Число дополнительных тактов ожидания, задаваемых полем WS регистров CCON
<b>w</b>	Число тактов ожидания поступления сигнала nACK
<b>nCS<sub>x</sub></b>	Один из четырёх сигналов nCS[3:0]
<b>CSIO<sub>x</sub></b>	Один из четырёх сигналов nCSIO[3:0]

### 9.3.2 Обмен данными с асинхронной памятью

Временные диаграммы записи данных в асинхронную память приведены на Рисунок 9.1 - Рисунок 9.3.

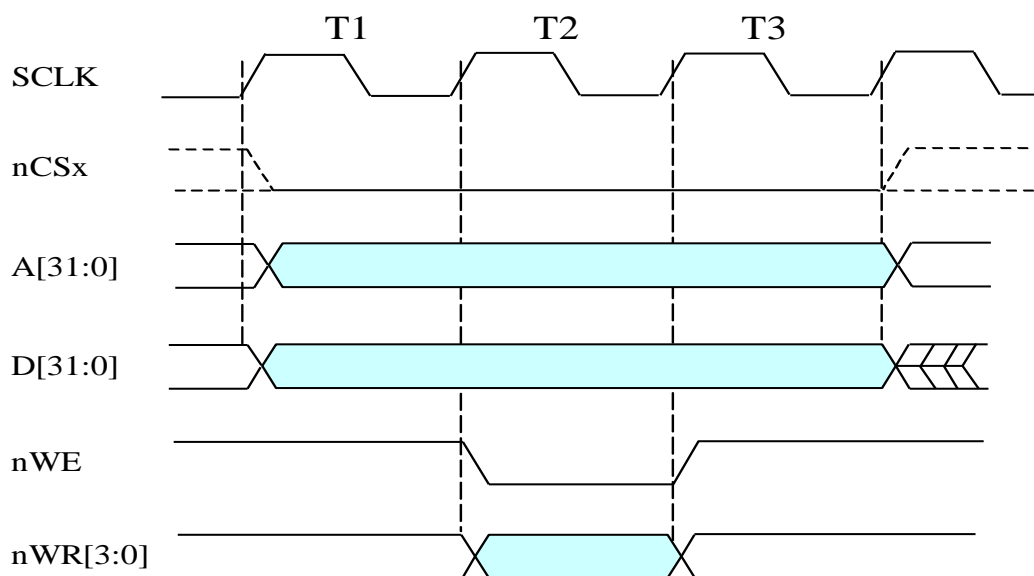


Рисунок 9.1. Запись в асинхронную память без дополнительных тактов ожидания.

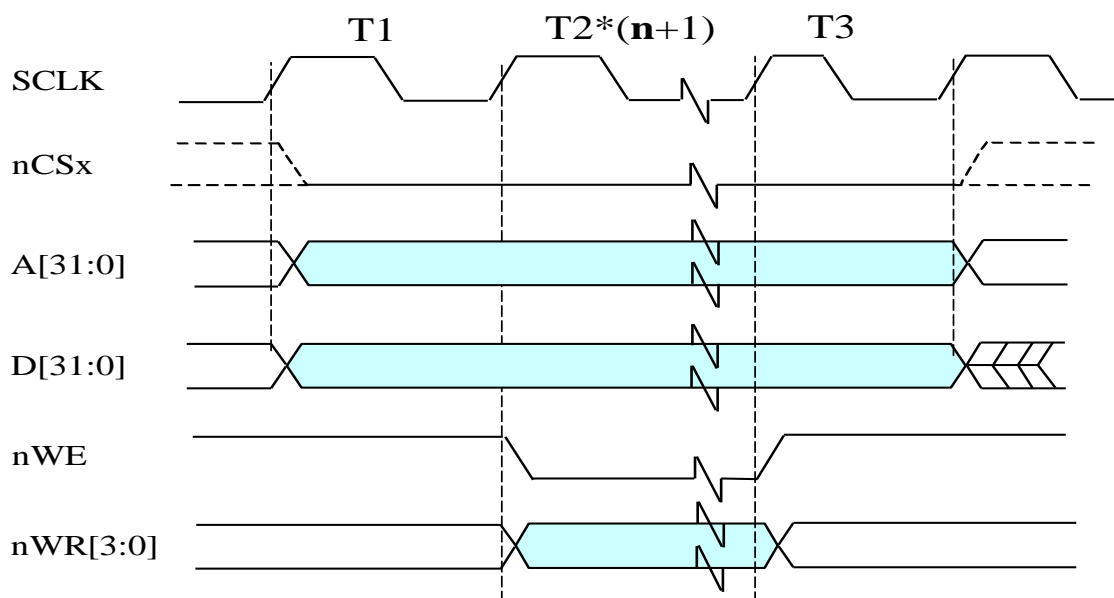
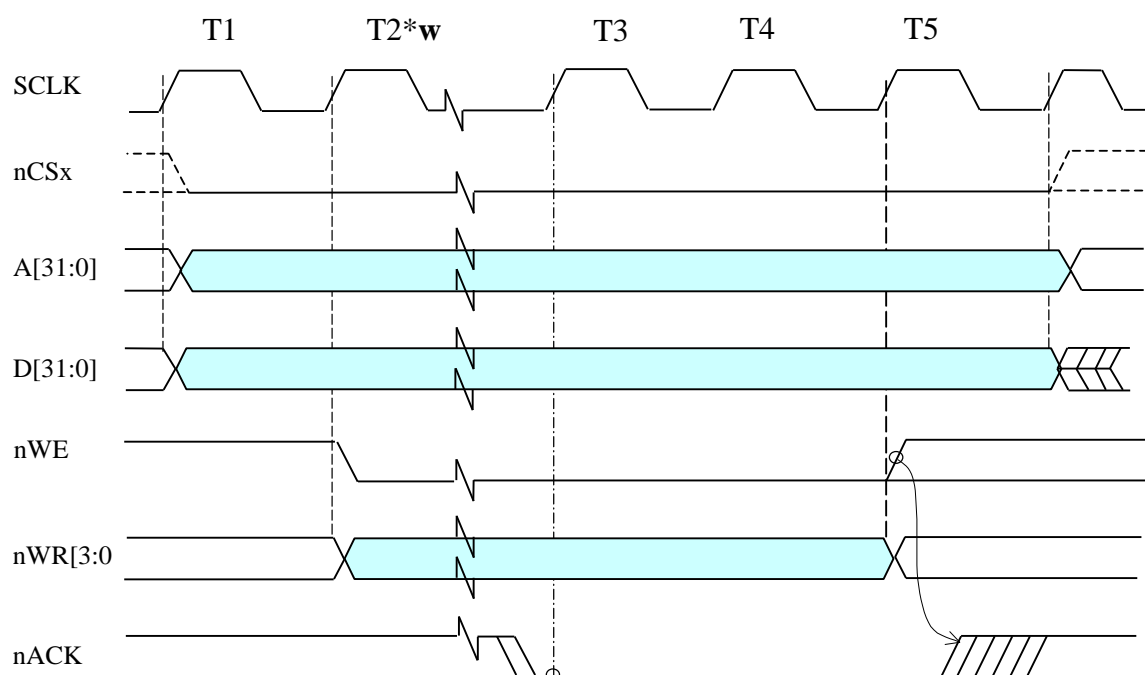
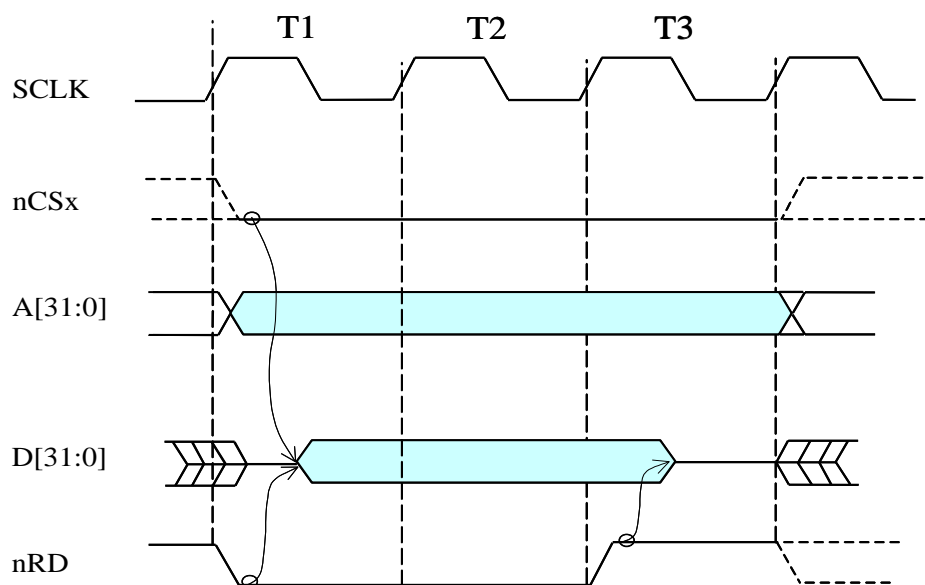


Рисунок 9.2. Запись в асинхронную память с  $n$  дополнительными тактами ожидания.

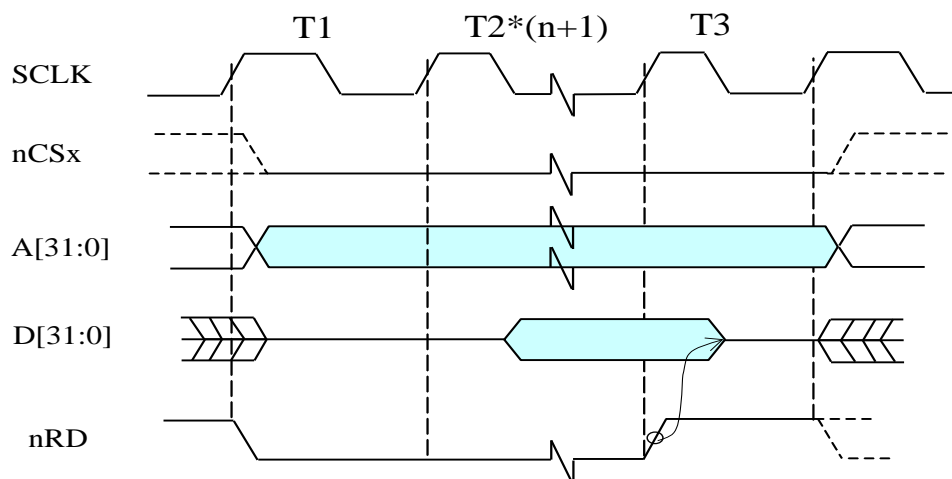


**Рисунок 9.3. Запись в асинхронную память с ожиданием сигнала nACK**

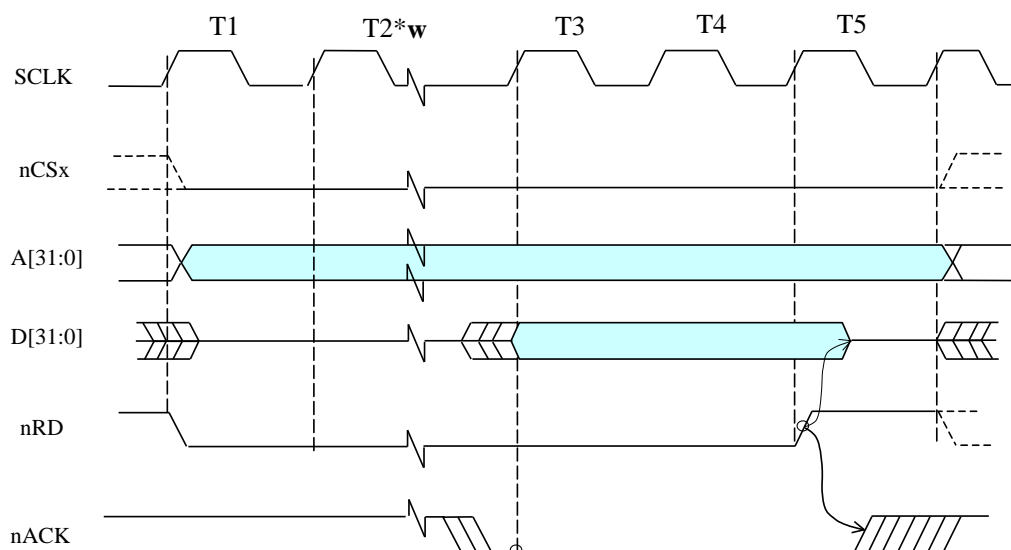
Временные диаграммы чтения данных из асинхронной памяти приведены на Рисунок 9.4 - Рисунок 9.7.



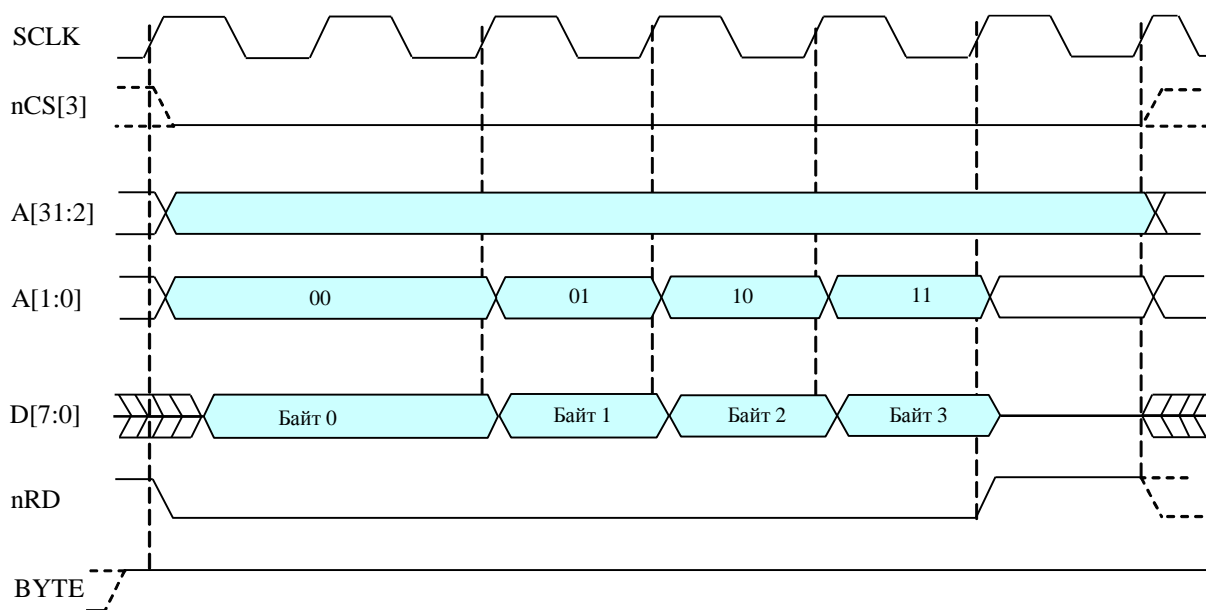
**Рисунок 9.4. Чтение асинхронной памяти без дополнительных тактов ожидания**



**Рисунок 9.5. Чтение асинхронной памяти с  $n$  дополнительными тактами ожидания.**

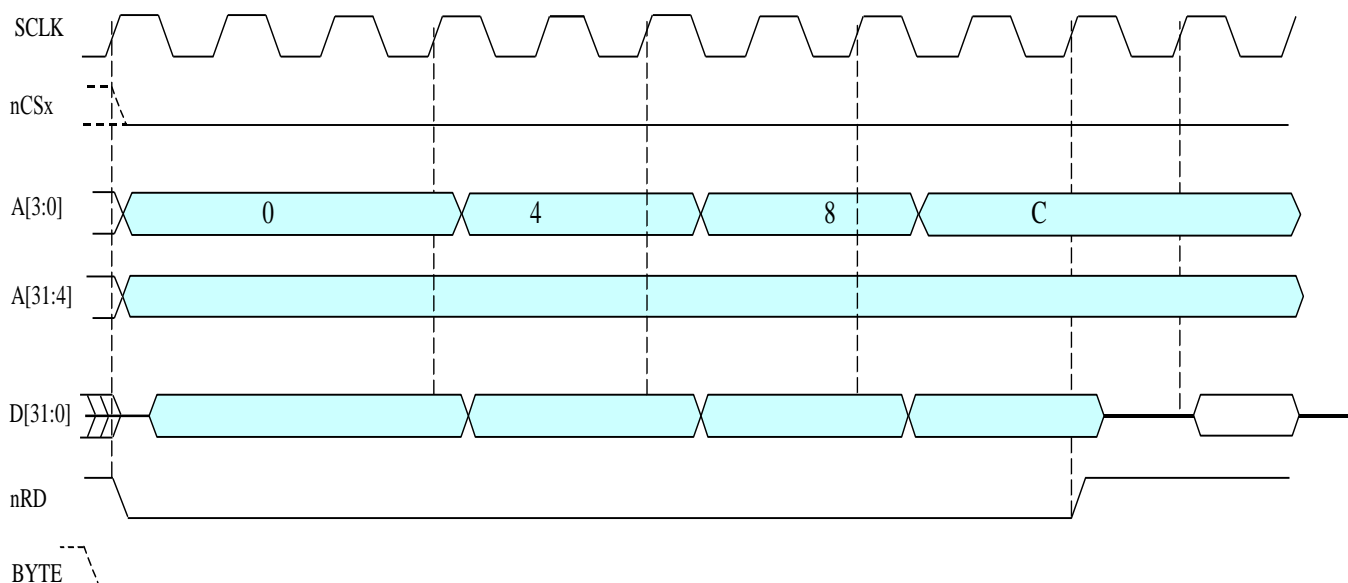


**Рисунок 9.6. Чтение данных из асинхронной памяти с ожиданием сигнала nACK.**

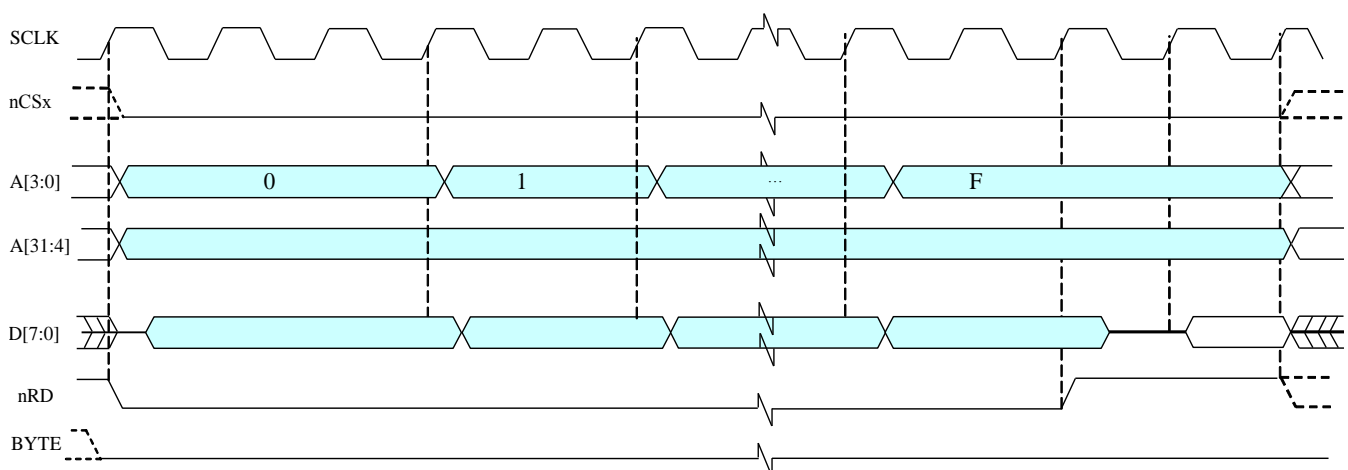


**Рисунок 9.7. Чтение 32-разрядного слова из 8-разрядной асинхронной памяти (BYTE = 1, n = 0).**

Если CPU выполняет программу из кэшируемой области внешней памяти, то загрузка строки кэш (процедура Refill) выполняются посредством чтения 4 слов в режиме burst. Адрес, по которому начинается burst, выровнен по 16-байтной границе. На Рисунок 9.8 приведена временная диаграмма выполнение процедуры Refill из 32-разрядной асинхронной памяти. На Рисунок 9.9 приведена временная диаграмма выполнение процедуры Refill из 8-разрядного ПЗУ.



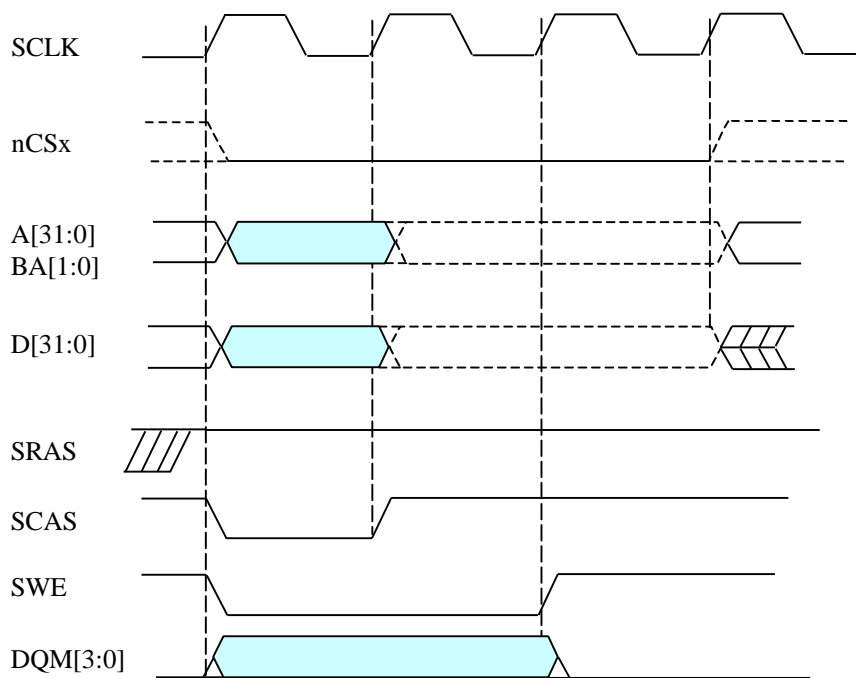
**Рисунок 9.8. Выполнение процедуры Refill из 32-разрядной асинхронной памяти (BYTE = 0, n = 0).**



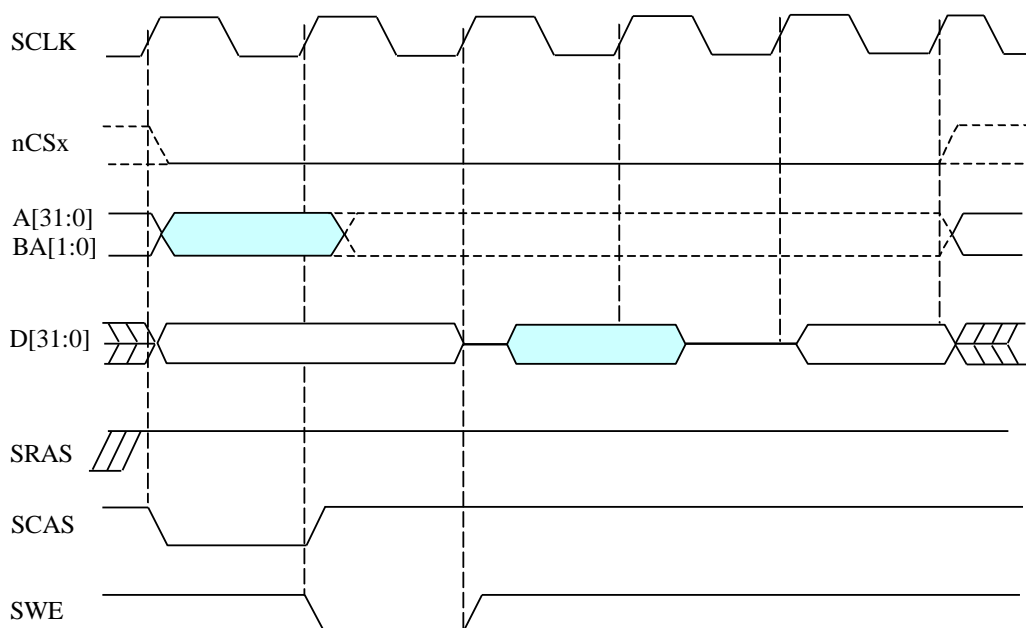
**Рисунок 9.9. Выполнение процедуры Refill из 8-разрядной асинхронной памяти (BYTE = 1, n = 0).**

### 9.3.3 Обмен данными с синхронной памятью

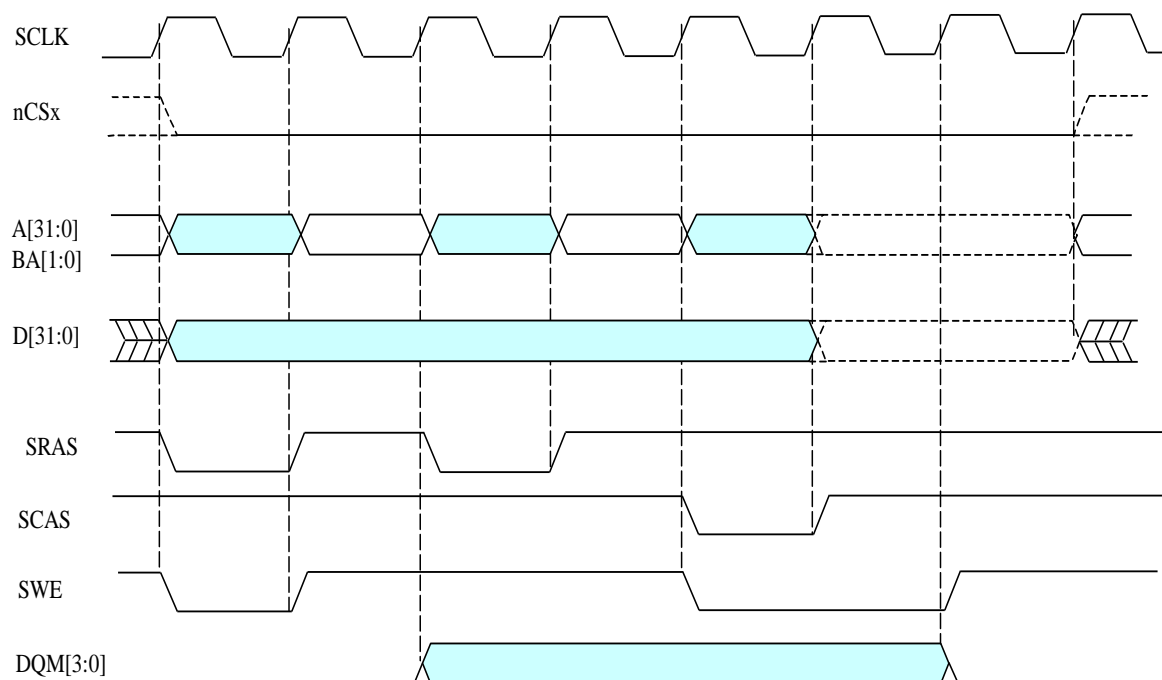
Временные диаграммы с синхронной памятью приведены на Рисунок 9.10 -Рисунок 9.16. Временные диаграммы инициализации и регенерации SDRAM приведены на Рисунок 9.17, Рисунок 9.18 соответственно.



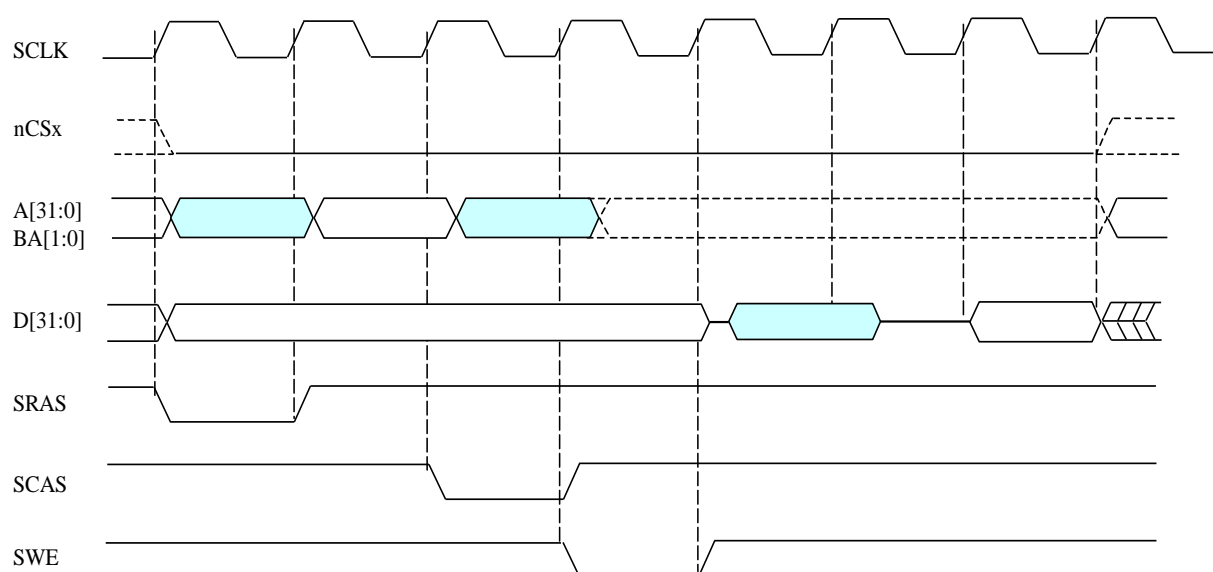
**Рисунок 9.10. Запись одного слова данных в синхронную память.**



**Рисунок 9.11. Чтение одного слова данных из синхронной памяти (здесь и далее CAS latency = 2)**

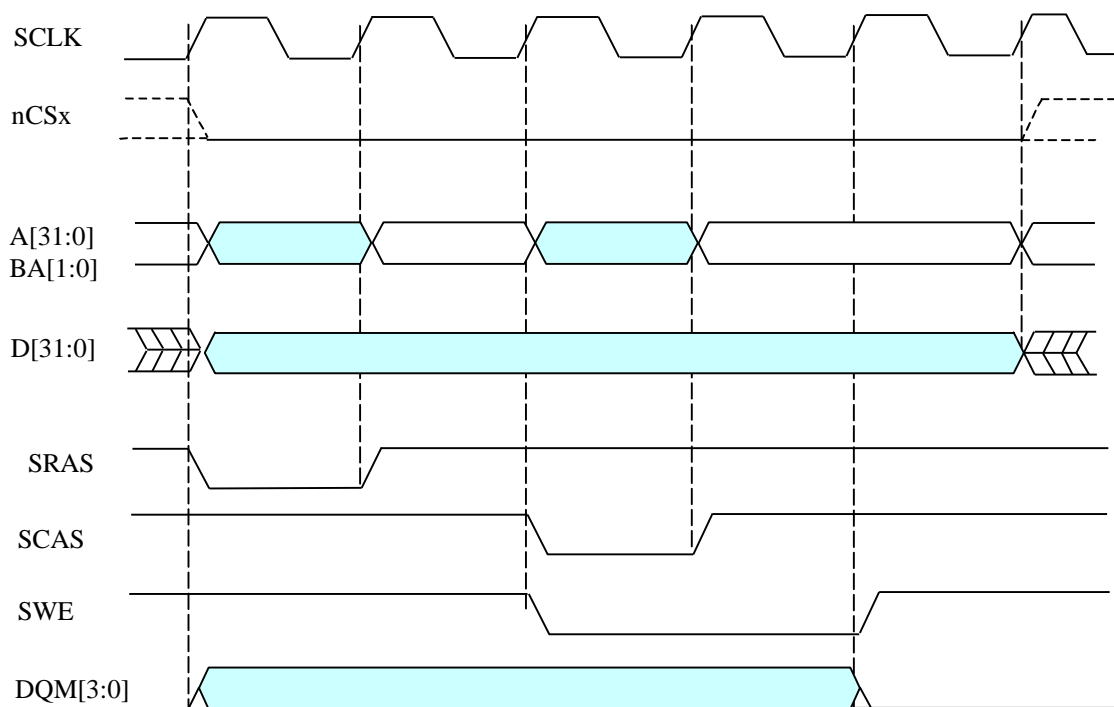


**Рисунок 9.12. Запись одного слова данных в синхронную память с деактивизацией строки**

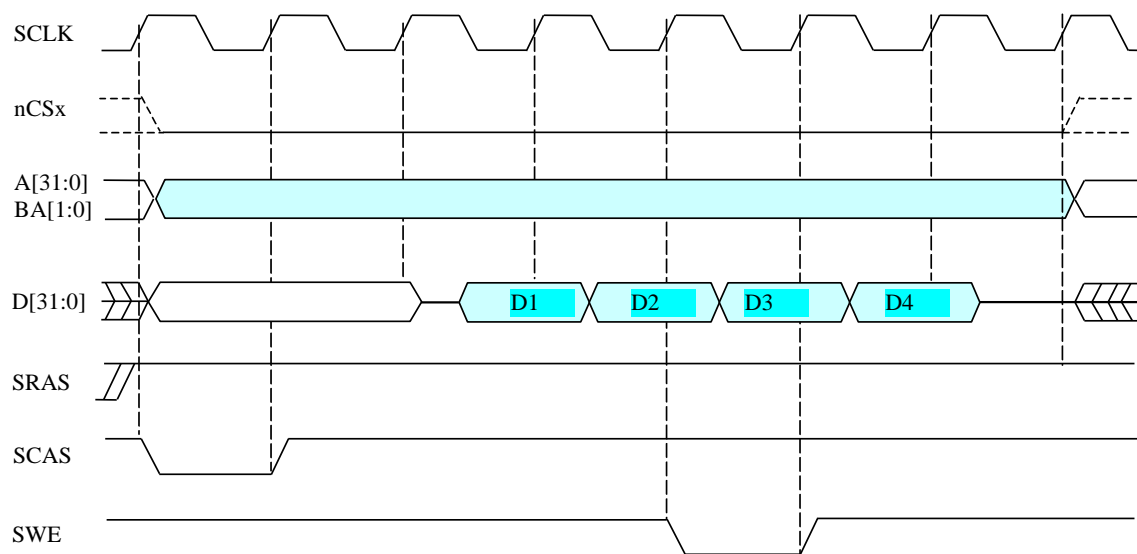


**Рисунок 9.13. Чтение одного слова данных из синхронной памяти с активизацией строки.**

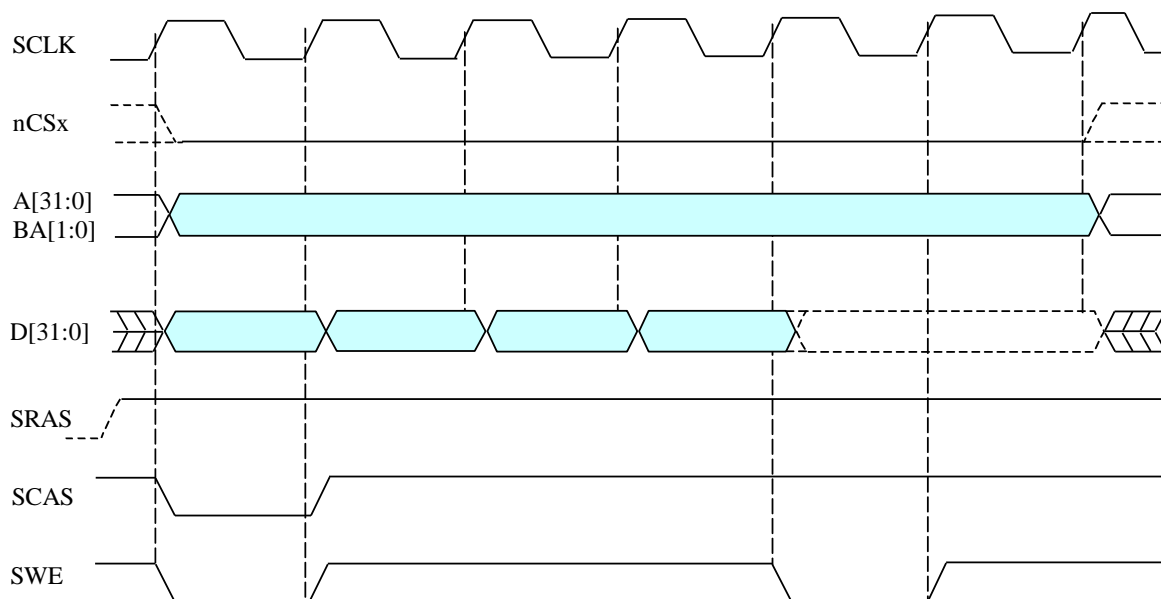




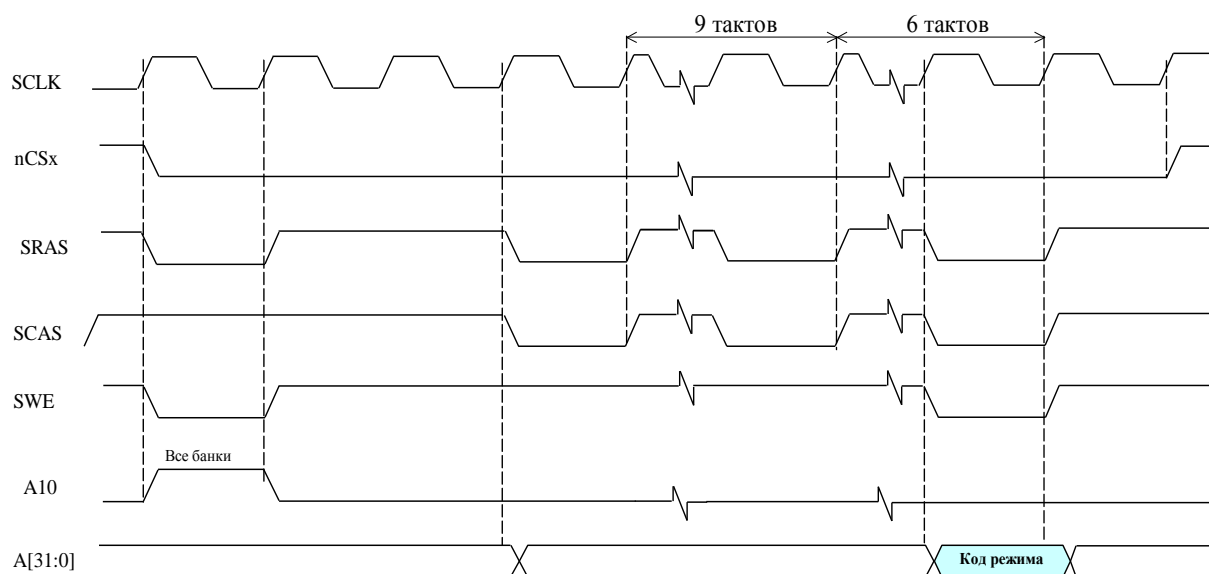
**Рисунок 9.14. Запись одного слова данных в синхронную память с активизацией строки.**



**Рисунок 9.15. Чтение 4-х слов данных из синхронной памяти в режиме “burst”.**



**Рисунок 9.16. Запись 4-х слов данных в синхронную память в режиме “burst”**



**Рисунок 9.17. Инициализация синхронной памяти**

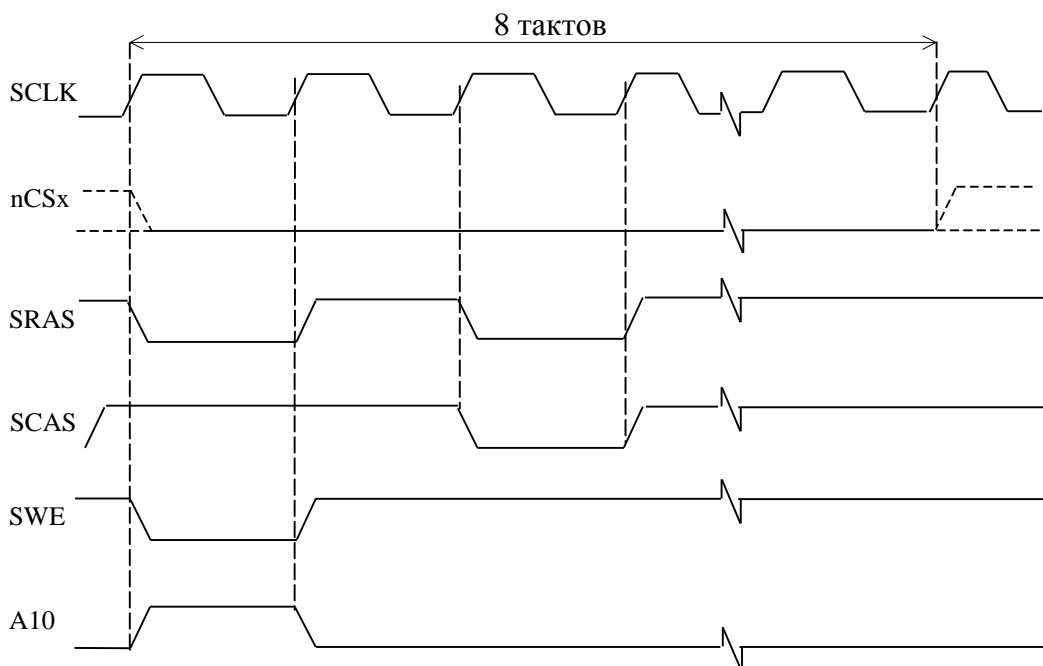


Рисунок 9.18. Временная диаграмма регенерация синхронной памяти.

### 9.3.4 Обмен данными в режиме Flyby

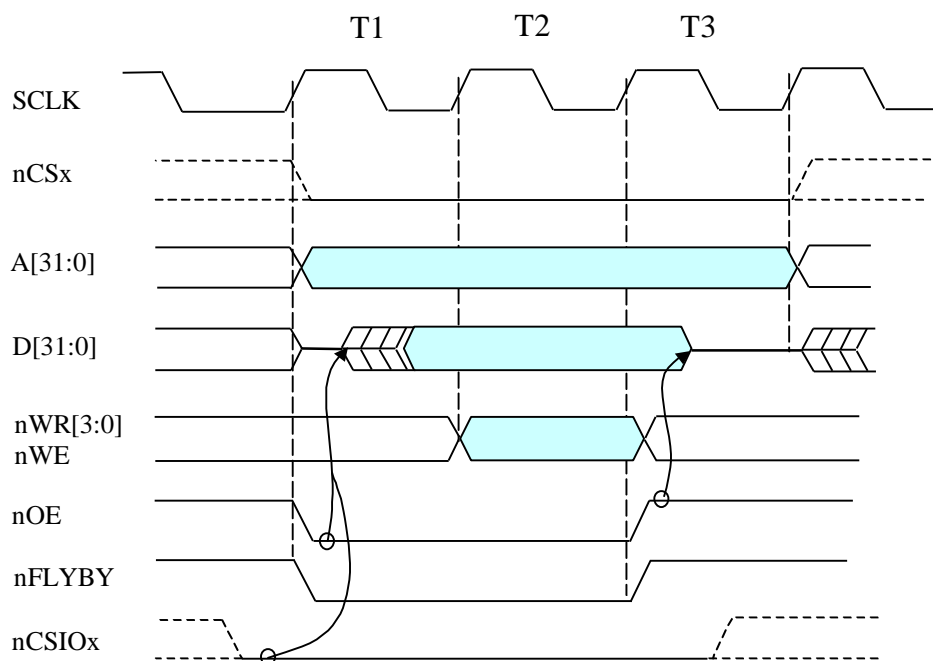
Режим Flyby используется контроллером DMA (каналы MemCh) для передачи данных между внешним устройством ввода-вывода и внешней памятью (как асинхронной, так и синхронной). Например, контроллер DMA может быть запрограммирован для передачи данных из аналого-цифрового преобразователя в SDRAM. Для выполнения передачи данных в режиме Flyby в соответствующем регистре CSR\_MemCh необходимо установить бит 11.

При передаче данных в режиме Flyby 1892BM3T отключается от шины данных, и активизирует внешнюю память и внешнее устройство ввода-вывода одновременно. Память управляется как обычно, а устройство ввода-вывода – при помощи сигналов nFLYBY (признак данного режима), nOE (активизация выходных формирователей устройства ввода-вывода) и nCSIO[3:0] (выбор устройства ввода-вывода).

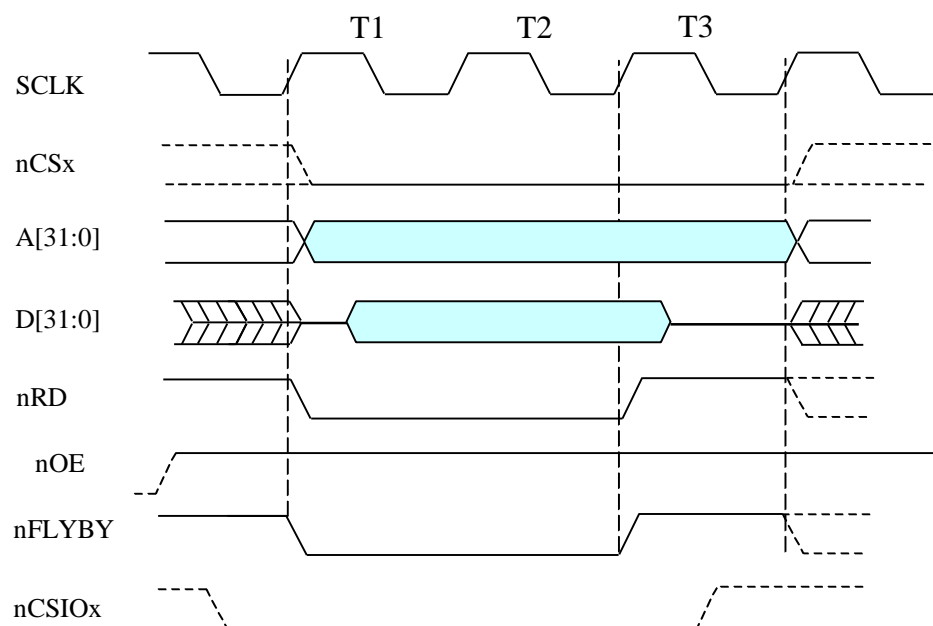
Каждому каналу MemCh может соответствовать свое устройство ввода-вывода. Выбор устройства ввода-вывода осуществляется посредством сигналов nCSIO[3:0]. Каналу MemCh0 соответствует низкий уровень на выводе nCSIO[0], каналу MemCh1 соответствует низкий уровень на выводе nCSIO[1], и так далее.

При работе с медленными внешними устройствами можно использовать сигнал nACK следующим образом. Если nFLYBY=1, то nACK=0. По сигналу nFLYBY=0 nACK переводится в «1» и удерживается в этом состоянии необходимое время. Для завершения обмена nACK переводится в состояние «0».

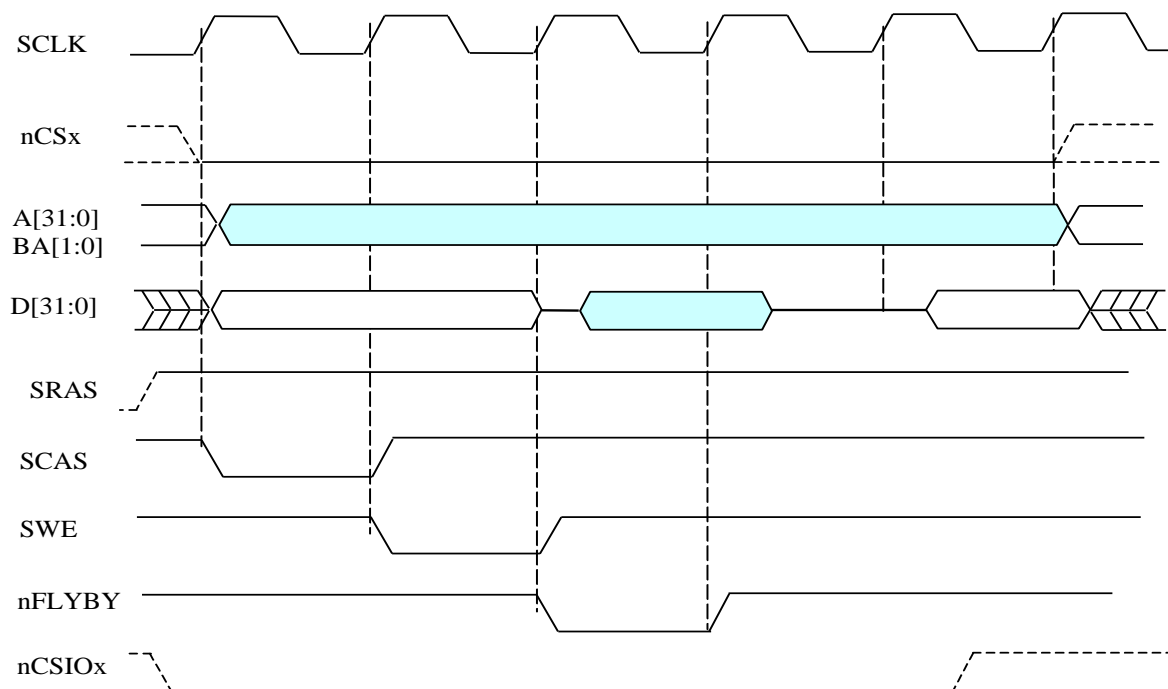
Временные диаграммы обмена данными в режиме Flyby приведены на Рисунок 9.19 - Рисунок 9.24 (WS=0, AE=0, CL=0).



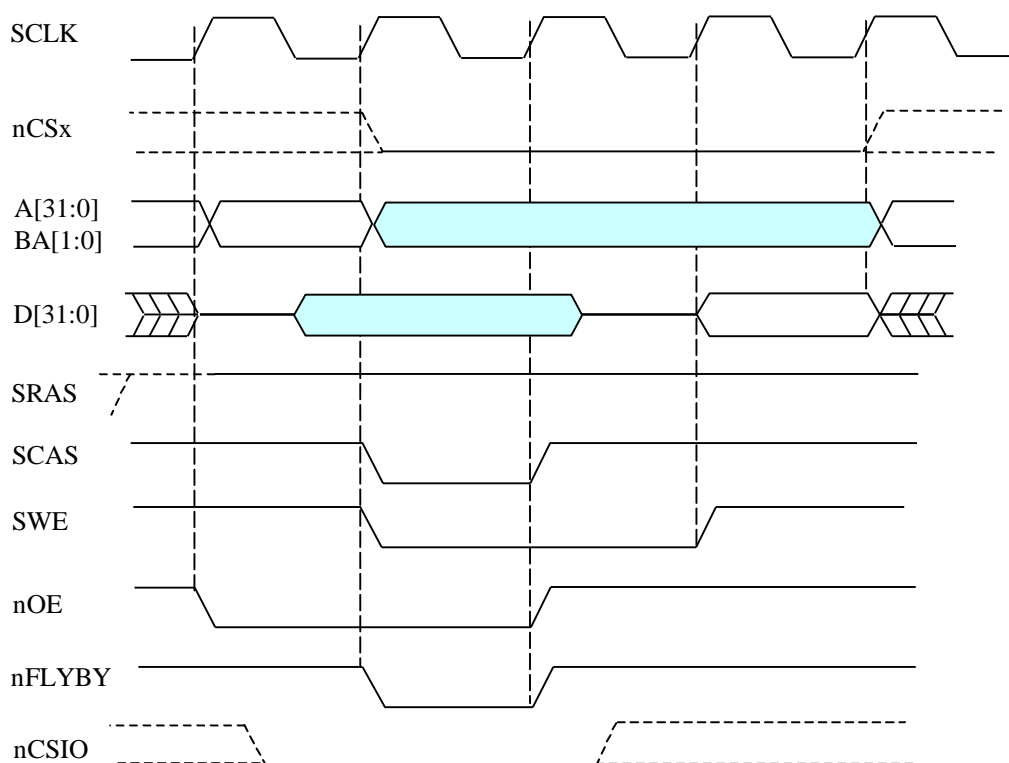
**Рисунок 9.19. Передача одного слова данных из устройства ввода-вывода в асинхронную память.**



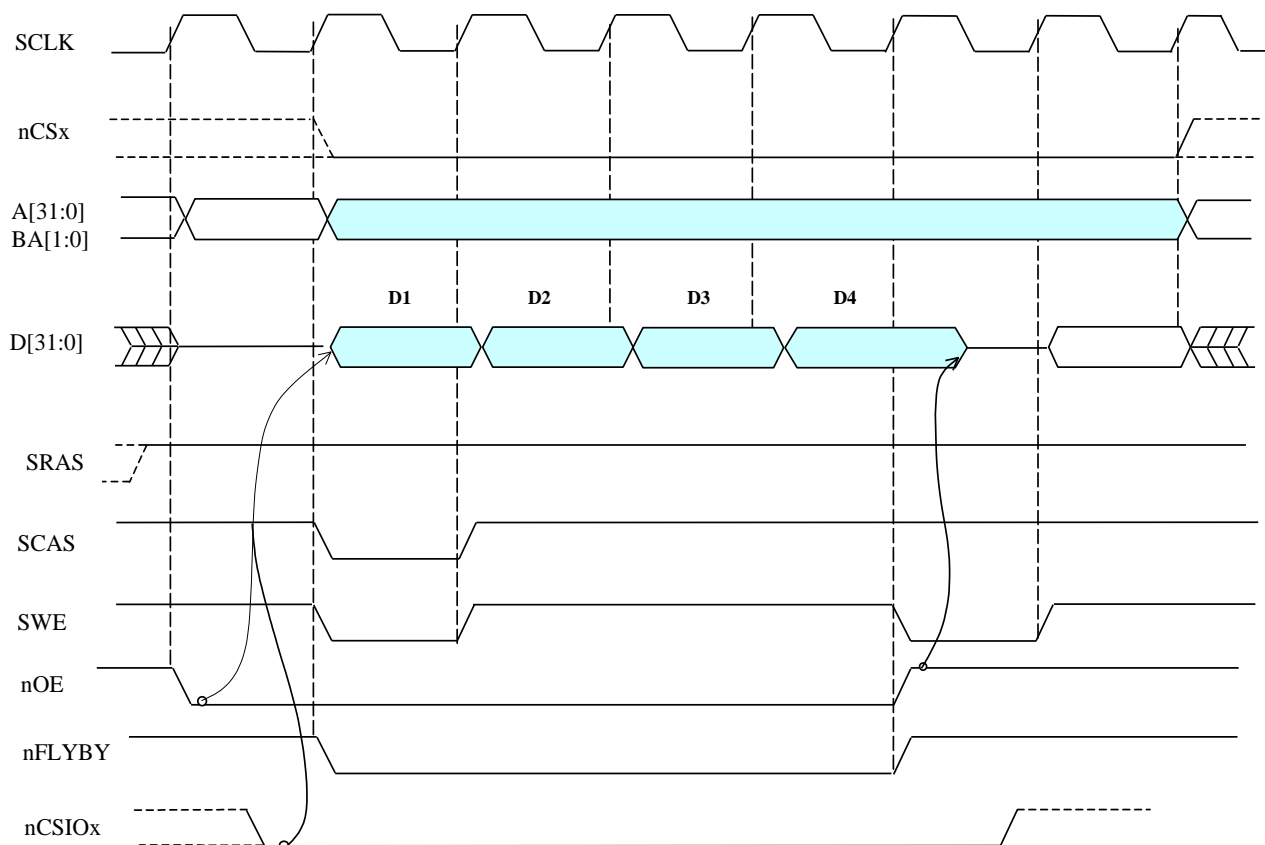
**Рисунок 9.20. Передача одного слова данных из асинхронной памяти в устройство ввода-вывода.**



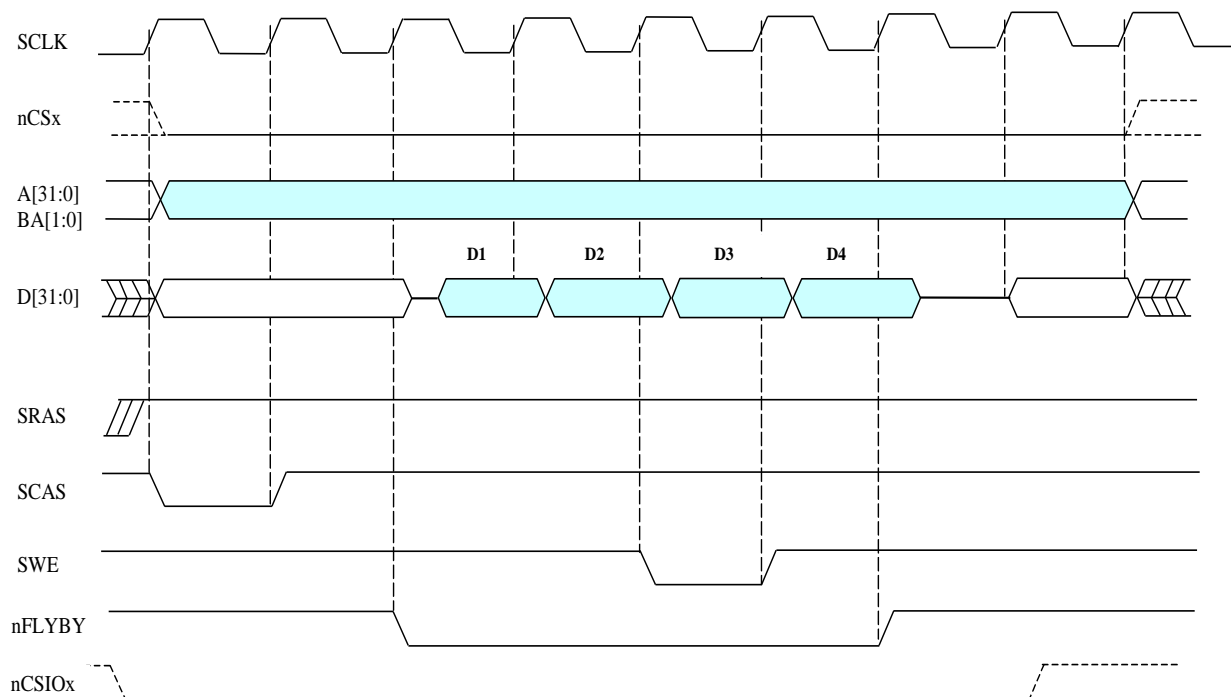
**Рисунок 9.21. Передача одного слова данных из синхронной памяти в устройство ввода-вывода**



**Рисунок 9.22. Передача одного слова данных из устройства ввода-вывода в синхронную память.**



**Рисунок 9.23. Передача 4-х слов данных из устройства ввода-вывода в синхронную память.**



**Рисунок 9.24. Передача 4-слов данных из синхронной памяти в устройство ввода-вывода.**

## 9.4 Рекомендации по подключению внешней памяти

### 9.4.1 Память типа SDRAM

Выводы адреса микросхем типа SDRAM подключаются к выводам шины адреса порта внешней памяти следующим образом:

- номер банка SDRAM – к выводам BA[1:0];
- адрес A[12:0] SDRAM – к выводам A[14:13], A10, A[11:2] соответственно.

### 9.4.2 Память типа Flash

К микросхеме 1892BM3T можно подключать 32-разрядную или 8-разрядную память типа Flash.

32-разрядная память Flash подключается к 1892BM3T аналогично статической памяти. Как правило, она подключается к сигналу выборки памяти nCS[3] и используется для старта 1892BM3T. Но при необходимости, 32-разрядная память Flash может быть подключена к любому из 4-х сигналов выборки памяти nCS[3:0].

8-разрядная память Flash подключается только к сигналу выборки памяти nCS[3], а на вход BYTE 1892BM3T необходимо подать высокий уровень. Выходную адресную шину 1892BM3T необходимо подключать к памяти Flash, начиная с 0 разряда (к 32-разрядной памяти адрес подключается, начиная со 2 разряда).

При использовании памяти типа Flash возможны два варианта ее программирования:

1. Микросхемы этой памяти программируются на программаторе и потом распаивается на плату или устанавливаются в контактирующее устройство.
2. Микросхемы этой памяти программируются на плате через порт JTAG микросхемы 1892BM3T. Для процесса программирования необходим специальный драйвер, который не входит в состав MC Studio.

Если используется 8-разрядная память Flash и требуется ее программирование в составе платы через порт JTAG 1892BM3T, то при ее проектировании необходимо иметь в виду следующую особенность микросхемы 1892BM3T. В этой микросхеме разряды адреса A[1:0] изменяются только при чтении из 8-разрядной памяти, а при записи в память (8- или 32-разрядную) они имеют постоянно нулевое состояние. Поэтому, для обеспечения записи в 8-разрядную память Flash через порт JTAG разряды адреса A[1:0] от 1892BM3T при помощи внешней логики необходимо объединить по логическому ИЛИ с двумя сигналами, при помощи которых можно перебрать все состояния шины адреса микросхемы памяти Flash.

## 10. УНИВЕРСАЛЬНЫЙ АСИНХРОННЫЙ ПОРТ (UART)

### 10.1 Общие положения

Универсальный асинхронный порт (далее UART) имеет следующие характеристики:

- по архитектуре совместим с UART 16550;
- частота приема и передачи данных – от 50 до 1 Мбод;
- FIFO для приема и передачи данных имеют объем по 16 байт;
- полностью программируемые параметры последовательного интерфейса: длина символа от 5 до 8 бит; генерация и обнаружение бита четности; генерация стопового бита длиной 1, 1.5 или 2 бита;
- диагностический режим внутренней петли;
- эмуляция символьных ошибок;
- функция управления модемом (CTS, RTS, DSR, DTR, RI, DCD).

Структурная схема порта UART приведена на Рисунок 10.1.

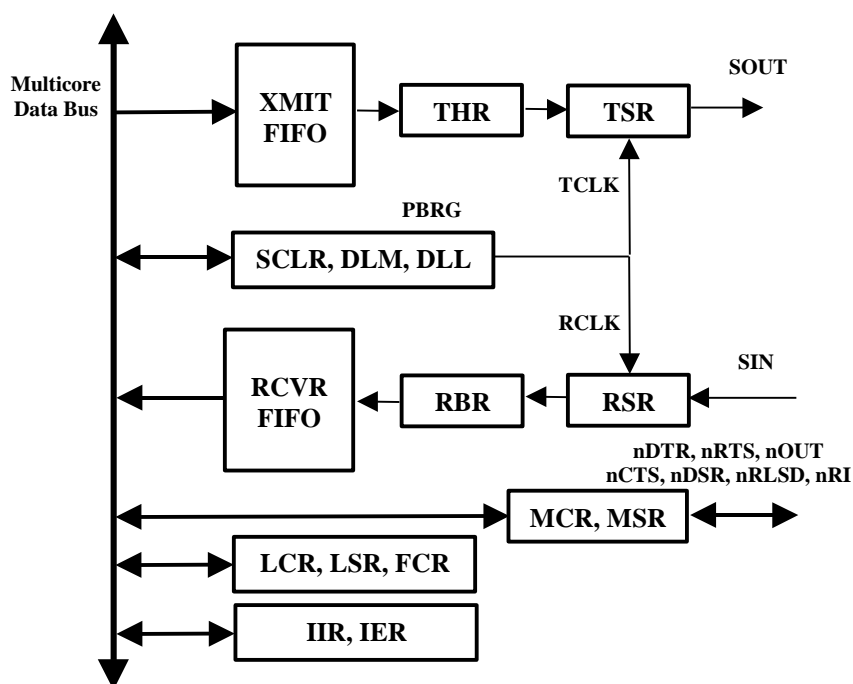


Рисунок 10.1. Структурная схема UART

Передаваемые данные записываются в регистр THR, а затем аппаратно переписываются в передающий сдвигающий регистр (TSR), если он пуст. После этого в регистр THR могут быть записаны следующие данные.

После приема данных в приемный сдвигающий регистр (RSR) данные переписываются в регистр RBR, если он не занят.



Назначение внешних выводов UART приведено в Таблица 10.1.

**Таблица 10.1. Внешние выводы UART.**

Название вывода	Тип вывода	Описание
SIN	I	Вход последовательных данных
SOUT	O	Выход последовательных данных
nDTR	O	Готовность UART к установлению связи (Data Terminal Ready)
nRTS	O	Готовность UART к обмену данными (Request To Send)
nOUT1	O	Выход общего назначения
nOUT2	O	Выход общего назначения
nCTS	I	Готовность модема к обмену данными (Clear To Send)
nDSR	I	Готовность модема к установлению связи (Data Set Ready)
nDCD	I	Признак обнаружения модемом несущей частоты (Receiver Line Signal Detect)
nRI	I	Признак обнаружения модемом телефонного звонка (Ring Indicator)

## 10.2 Регистры UART

### 10.2.1 Общие положения

Перечень регистров UART приведен в Таблица 10.2.

**Таблица 10.2. Перечень регистров UART**

Условное обозначение регистра	Название регистра	Смещение	Доступ (R-чтение, W-запись)
RBR	Приемный буферный регистр	0 (DLAB=0)	R
THR	Передающий буферный регистр	0 (DLAB=0)	W
IER	Регистр разрешения прерываний	1 (DLAB=0)	R/W
IIR	Регистр идентификации прерывания	2	R
FCR	Регистр управления FIFO	2	W
LCR	Регистр управления линией	3	R/W
MCR	Регистр управления модемом	4	R/W
LSR	Регистр состояния линии	5	R
MSR	Регистр состояния модема	6	R/W
SPR	Регистр Scratch Pad	7	R/W
DLL	Регистр делителя младший	0 (DLAB=1)	R/W
DLM	Регистр делителя старший	1 (DLAB=1)	R/W
SCLR	Регистр предделителя (scaler)	5	W

## 10.2.2 Регистр LCR

Формат регистра LCR приведен в Таблица 10.3.

**Таблица 10.3. Формат регистра LCR**

Номер бита	Условное Обозначение	Назначение
1:0	WLS (Word Length Select)	Количество бит данных в передаваемом символе: 00 - 5 бит, 01 - 6 бит, 10 - 7 бит, 11 - 8 бит.
2	STB (Number Stop Bits)	Количество стоп-бит: 0 - 1 стоп-бит, 1 - 2 стоп-бита (для 5-битного символа стоп-бит имеет длину 1,5 бита). Приемник анализирует только первый стоп бит.
3	PEN (Parity Enable)	Разрешение генерации (передатчик) или проверки (приемник) контрольного бита: 1 – контрольный бит (паритет или постоянный) разрешен, 0 – запрещен.
4	EPS (Even Parity Select)	Выбор типа контроля (при PEN=1): 0 – нечетность, 1 – четность.
5	STP (Stick Parity)	Принудительное формирование бита паритета: 0 – контрольный бит генерируется в соответствии с паритетом выводимого символа, 1 – постоянное значение контрольного бита: при EPS=1 - нулевое, при EPS=0 – единичное.
6	SBC (Set Break Control)	Формирование обрыва линии: 0 – нормальная работа; 1 – на выходе SOUT устанавливается низкий уровень (Spacing level). Это влияет только на выход SOUT, а не на логику передачи символа.
7	DLAB (Divisor Latch Access bit)	Управление доступом к регистрам: 0 – разрешен доступ к регистрам RBR, THR, IER; 1 – разрешен доступ к регистрам DLL, DLM

Исходное состояние регистра LCR – нули.

Бит SBC используется как признак «Внимание» для приемного терминала, подключенному к выходу UART. Для того чтобы не было передано ошибочного символа при использовании бита SBC, необходимо выполнять следующую последовательность действий:

- Загрузить в регистр THR все нули по признаку THRE=1;
- Установить SBC=1 по следующему THRE=1;
- Дождаться TEMT=1.

Для восстановления нормальной передачи необходимо установить SBC=0.

### 10.2.3 Регистр FCR

Формат регистра FCR приведен в Таблица 10.4.

**Таблица 10.4. Формат регистра FCR**

Номер бита	Условное обозначение	Назначение
0	FEWO (FIFO Enable)	Разрешение работы XMIT и RCVR FIFO: 0 – символьный режим; 1 – режим FIFO. При изменении состояния этого бита, данные из FIFO, не удаляются. Запись в биты RFR, TFR, RFTL выполняется, если FEWO=1.
1	RFR (Receiver FIFO Reset)	Установка RCVR FIFO в исходное состояние. Регистр RSR не обнуляется. После записи 1 в этот бит он автоматически сбрасывается.
2	TFR (Transmitter FIFO Reset)	Установка XMIT FIFO в исходное состояние. Регистр TSR не обнуляется. После записи 1 в этот бит он автоматически сбрасывается.
5:3	-	Резерв
7:6	RFTL (RCVR FIFO Trigger Level)	Порог заполнения RCVR FIFO (в байтах), при котором формируется прерывание: 00 – 1; 01 – 4; 10 – 8; 11 – 14.

Исходное состояние регистра FCR – нули.

### 10.2.4 Регистр LSR

Формат регистра LSR приведен в Таблица 10.5.

**Таблица 10.5. Формат регистра LSR**

Номер бита	Условное обозначение	Назначение
0	RDR (Receiver Data Ready)	Готовность данных. Устанавливается после приема символа данных и передачи его в регистр RBR или FIFO. Сбрасывается после чтения регистра RBR (в символьном режиме) или чтения всего содержимого RCVR FIFO (в режиме FIFO)
1	OE (Overrun Error)	Ошибка переполнения. Устанавливается, если содержимое регистра RBR не было прочитано, в сдвигающий регистр принят следующий символ и начат прием очередного символа. При этом новый символ записывается в сдвигающий регистр вместо старого. В режиме FIFO устанавливается, если после перехода порогового (trigger) уровня FIFO заполнено до конца, во входной сдвигающий регистр полностью принят следующий

Номер бита	Условное обозначение	Назначение
		символ и начат прием очередного символа. При этом в FIFO ничего не передается. Бит сбрасывается при чтении содержимого регистра LSR.
2	PE (Parity Error)	Ошибка контрольного бита (паритета или фиксированного). В режиме FIFO этот бит показывает на ошибку в символе, находящемся наверху FIFO. Бит сбрасывается при чтении содержимого регистра LSR.
3	FE (Framing Error)	Ошибка кадра. Устанавливается, если стоп-бит равен нулю (Spacing level). В режиме FIFO этот бит показывает на ошибку в символе, находящемся наверху FIFO. После этой ошибки UART пересинхронизируется. Бит сбрасывается при чтении содержимого регистра LSR.
4	BI (Break Interrupt)	Обрыв линии. Устанавливается, если вход приема данных находится в состоянии 0 (Spacing level) не менее чем время передачи всего символа. В режиме FIFO этот бит показывает на ошибку в символе, находящемся наверху FIFO. При возникновении этой ситуации, в FIFO загружается только один нулевой символ. Прием следующих символов разрешается после того, как вход приема данных перейдет в единичное состояние (Marking state) и будет принят действительный стартовый бит. Бит сбрасывается при чтении содержимого регистра LSR.
5	THRE (Transmitter Holding Register Empty)	Передающий буферный регистр пуст. Показывает, что UART готов принять следующий символ для передачи. Устанавливается, когда содержимое регистра THR передается в передающий сдвигающий регистр. Одновременно с этим генерируется прерывание THREI, если оно разрешено. Бит сбрасывается при записи символа в регистр THR. В режиме FIFO этот бит устанавливается, когда XMIT FIFO пусто, и сбрасывается, если в XMIT FIFO записывается хотя бы один символ.
6	TEMT (Transmitter Empty)	Передатчик пуст. Устанавливается, если регистры THR и TSR пусты. Имеет нулевое состояние, если хотя бы один из регистров THR и TSR не пуст. В режиме FIFO этот бит устанавливается, если нет символов ни в XMIT FIFO, ни в регистре TSR.
7	EIRF (Error in RCVR FIFO)	Наличие хотя бы одного признака ошибки в FIFO. В символьном режиме этот бит всегда равен нулю. Бит сбрасывается при чтении содержимого регистра LSR, если в FIFO нет больше признаков ошибок.

Исходное состояние бит THRE, TEMT – 1, остальных – 0.

Установка бит OE, PE, FE, BI приводит к формированию прерыванию по состоянию входа приема данных (Receiver Line Status Interrupt), если это прерывание разрешено.

### 10.2.5 Регистр IER

Формат регистра IER приведен в Таблица 10.6. Исходное состояние регистра IER – нули.

**Таблица 10.6. Формат регистра IER**

Номер бита	Условное обозначение	Назначение
0	ERBI	Разрешение прерывания по наличию принятых данных (RDAI), а также по таймауту (CTI)
1	ETBEI	Разрешение прерывания по отсутствию данных в регистре THR (THREI)
2	ERLSI	Разрешение прерывания по статусу приема данных (RLSI)
3	EMSI	Разрешение прерывания по статусу модема (MSI)
7:4	-	Резерв

### 10.2.6 Регистр IIR

Формат регистра IIR приведен в Таблица 10.7.

**Таблица 10.7. Формат регистра IIR**

Номер бита	Условное обозначение	Назначение
0	IP (Interrupt Pending)	Признак наличия прерывания: 0 – есть прерывание; 1 – нет прерывания.
3:1	IID[2:0]	Код идентификации прерывания в соответствии с Таблица 10.8.
5:4	-	Резерв
7:6	FE	Признак разрешения работы RCVR и XMIT FIFO

Исходное состояние бита IP – 1, остальных – 0.

**Таблица 10.8. Идентификация прерываний**

Код поля IID[2:0]	Уровень приоритета (1 – наивысший)	Тип прерывания	Причина прерывания	Условие сброса прерывания
011	1	Статус приема данных (RLSI – Receiver Line Status Interrupt)	OE - Overrun Error; PE - Parity Error; FE - Framing Error; BI - Break Interrupt.	Чтение содержимого регистра LSR. Чтение из FIFO символа, по которому сформировано это прерывание. Обнуление FIFO.

Код поля ID[2:0 ]	Уровень приоритета (1 – наивысший )	Тип прерывания	Причина прерывания	Условие сброса прерывания
010	2	Наличие принятых данных (RDAI – Received Data Available Interrupt)	Наличие данных в регистре RBR или достижение заданного порога FIFO	Чтение содержимого регистра RBR. Считывание данных из FIFO до уровня ниже порогового.
110	2	Таймаут (CTI – Character Timeout Interrupt)	С момента приема последнего символа в RCVR FIFO прошло время, равное длительности передачи 4-х символов и не было ни чтения FIFO, ни приема очередного символа.	Чтение содержимого регистра RBR. Прием очередного символа. Сброс FIFO.
001	3	Регистр THR пуст (THREI – Transmitter Holding Register Empty Interrupt)	Регистр THR пуст	Запись символа в регистр THR
000	4	Статус модема (MSI – Modem Status Interrupt)	Изменение состояния сигналов на входах порта nCTS, nDSR, nRI, nDCD	Чтение содержимого регистра MSR.

### 10.2.7 Регистр MCR

Формат регистра MCR приведен в Таблица 10.9.

**Таблица 10.9. Формат регистра MCR**

Номер бита	Условное обозначение	Назначение
0	DTR	Управление выходом nDTR: 0 – на выходе высокий уровень; 1 – на выходе низкий уровень.
1	RTS	Управление выходом nRTS: 0 – на выходе высокий уровень; 1 – на выходе низкий уровень.
2	Out 1	Управление выходом OUT1: 0 – на выходе высокий уровень; 1 - на выходе низкий уровень.
3	Out 2	Управление выходом OUT1: 0 – на выходе высокий уровень; 1 - на выходе низкий уровень.
4	LOOP	Режим петли. Используется для тестирования UART. При установке этого бита в 1 выполняется следующее: На выходе SOUT UART устанавливается высокий уровень; Вход SIN UART отключается от внешнего вывода; Выход регистра TSR подключается к входу регистра RSR; На выходах nDTR, nRTS, nOUT1, nOUT2 устанавливаются высокие уровни; Входы nCTS, nDSR, nDCD, nRI UART отключаются от внешних выводов; Выходы разрядов DTR, RTS, Out 1, Out 2 регистра MCR подключаются к входам разрядов DSR, CTS, RI, DCD регистра MSR соответственно. В режиме петли передаваемые данные немедленно принимаются. В режиме петли все прерывания формируются как обычно.
7:5	-	Резерв

Исходное состояние регистра MCR – нули.

### 10.2.8 Регистр MSR

Формат регистра MSR приведен в Таблица 10.10.

**Таблица 10.10. Формат регистра MCR**

Номер бита	Условное обозначение	Назначение
0	DCTS	Признаки любого изменения состояния входного сигнала CTS. Бит устанавливается в единичное состояние, если сигнал CTS изменил свое состояние после последнего считывания содержимого регистра MSR. Одновременно с этим

Номер бита	Условное обозначение	Назначение
		формируется прерывание MSI, если оно разрешено. Бит сбрасывается при чтении содержимого регистра MSR.
1	DDSR	Признаки любого изменения состояния входного сигнала DSR. Бит устанавливается в единичное состояние, если сигнал DSR изменил свое состояние после последнего считывания содержимого регистра MSR. Одновременно с этим формируется прерывание MSI, если оно разрешено. Бит сбрасывается при чтении содержимого регистра MSR.
2	TERI	Признаки перехода входного сигнала RI с низкого уровня на высокий уровень. Бит устанавливается в единичное состояние, если сигнал RI изменил свое состояние после последнего считывания содержимого регистра MSR. Одновременно с этим формируется прерывание MSI, если оно разрешено. Бит сбрасывается при чтении содержимого регистра MSR.
3	DDCD	Признаки любого изменения состояния входного сигнала nDCD. Бит устанавливается в единичное состояние, если сигнал nDCD изменил свое состояние после последнего считывания содержимого регистра MSR. Одновременно с этим формируется прерывание MSI, если оно разрешено. Бит сбрасывается при чтении содержимого регистра MSR.
4	CTS	Состояние сигнала на входе nCTS: 0 – на входе высокий уровень; 1 – на входе низкий уровень.
5	DSR	Состояние сигнала на входе nDSR: 0 – на входе высокий уровень; 1 – на входе низкий уровень.
6	RI	Состояние сигнала на входе nRI: 0 – на входе высокий уровень; 1 – на входе низкий уровень.
7	DCD	Состояние сигнала на входе nDCD: 0 – на входе высокий уровень; 1 – на входе низкий уровень.

Исходное состояние бит 3:0 регистра MSR – нули. Биты 7:4 следуют за инверсией состояния соответствующих входных сигналов.

### 10.2.9 Программируемый генератор скорости обмена

В UART имеется программируемый генератор скорости обмена данными (PBRG – Programmable Baud Rate Generator). Он состоит из 8-разрядного предделителя и 16-разрядного основного делителя частоты. На вход предделителя поступает системная тактовая частота CLK, на которой работает CPU, UART и другие устройства (см. рис. 4.1). Выходная частота предделителя поступает на вход основного делителя. Выходная частота генератора PBRG в 16 раз больше частоты обмена последовательными данными.



Значение частоты на выходе предделителя равно  $CLK/(SCLR + 1)$ . Коэффициент деления основного делителя задается 16-разрядным регистром, который является конкатенацией регистров DLM и DLL.

Период частот передачи и приема (TCLK и RCLK) UART вычисляется по формуле:

$CLK/(SCLR + 1) / ((\text{конкатенация содержимого регистров DLM и DLL}) * 16)$ . Минимальная величина, которая может быть записана в регистры {DLM, DLL}, равна 1.

Исходное состояние регистров DLL, DLM, SCLR – нули.

### 10.3 Работа с FIFO по прерыванию

Если установлен режим работы с FIFO (EFWO=1 в регистре FCR) и разрешены прерывания по приему (бит ERI=1 в регистре IER), то в процессе приема:

- формируется прерывание, если число символов в RCVR FIFO достигло запрограммируемого порога. Это прерывание сбрасывается, если при чтении из FIFO число символов оставшихся в нем, станет меньше запрограммируемого порога;
- одновременно с этим в регистре IIR устанавливается индикатор наличия принятых данных RDAI. Индикатор обнуляется, при чтении из FIFO до снижения запрограммируемого порога;
- может возникнуть прерывание по статусу приема данных (RLSI), приоритет которого выше, чем RDA;
- бит RDR в регистре LSR устанавливается в момент передачи символа из регистра RSR в RCVR FIFO. Этот бит обнуляется при считывании из FIFO всех символов данных.

Если установлен режим работы с FIFO (EFWO=1 в регистре FCR) и разрешены прерывания по приему (ERI=1 в регистре IER), то генерируется прерывание по таймауту, если с момента приема последнего символа в RCVR FIFO прошло время, равное длительности передачи 4-х символов и за это время не было:

- ни чтения RCVR FIFO;
- ни приема в RCVR FIFO очередного символа.

При 12-битном символе и скорости передачи 300 бод, прерывание по этой причине возникнет через 160 мс.

При возникновении прерывания по таймауту оно обнуляется при считывании символа из RCVR FIFO. При этом обнуляется и таймер, генерирующий данное прерывание. Если прерывание по таймауту не возникло, то таймер таймаута обнуляется при приеме нового символа или при считывании символа из RCVR FIFO.

Если установлен режим работы с FIFO (EFWO=1 в регистре FCR) и разрешены прерывания по передаче данных (бит ETI=1 в регистре IER), то генерируется прерывание по передаче следующим образом:

- формируется прерывание THREI, если XMIT FIFO пусто. Это прерывание обнуляется, как только выполняется запись символа в регистр THR (при приеме данного прерывания в XMIT FIFO можно записать от 1 до 16 символов);

- индикатор ТЕМТ в регистре LSR установится в единичное состояние через время равное длительности одного символа минус последний стоп бит, после установки THRE=1. Первое прерывание по передаче (если оно разрешено) формируется немедленно после установки FEWO=1.

## 10.4 Работа с FIFO по опросу

Если установлен режим работы с FIFO (EFWO=1 в регистре FCR) и запрещены прерывания, то обмен данными выполняется по опросу, а управление FIFO приема и передачи (RCVR, XMIT) выполняется раздельно.

В этом режиме опрос состояния RCVR и XMIT FIFO осуществляется программно, посредством считывания содержимого регистра LSR:

- бит RDR=1, пока есть данные в RCVR FIFO;
- биты OE, PE, FE, VI указывают на ошибки. Эти ошибки обрабатываются так же, как и при работе по прерыванию;
- бит THRE=1, если XMIT FIFO пусто;
- бит ТЕМТ=1, если в XMIT FIFO и TSR нет данных.

При работе по опросу нет индикации таймаута и факта достижения порога RCVR FIFO. Однако оба RCVR и XMIT FIFO могут хранить символы данных.

## 11. ПОРТ ОБМЕНА ПОСЛЕДОВАТЕЛЬНЫМ КОДОМ

### 11.1 Общие положения

Синхронный порт обмена последовательным кодом (SPORT) обеспечивают интерфейс ввода-вывода с широким набором периферийных устройств. Благодаря большому набору режимов тактовой и кадровой синхронизацией этот порт обеспечивает реализацию большого набора коммуникационных протоколов и простое аппаратное сопряжение со многими стандартными конверторами и кодеками.

Порт имеет следующие основные характеристики:

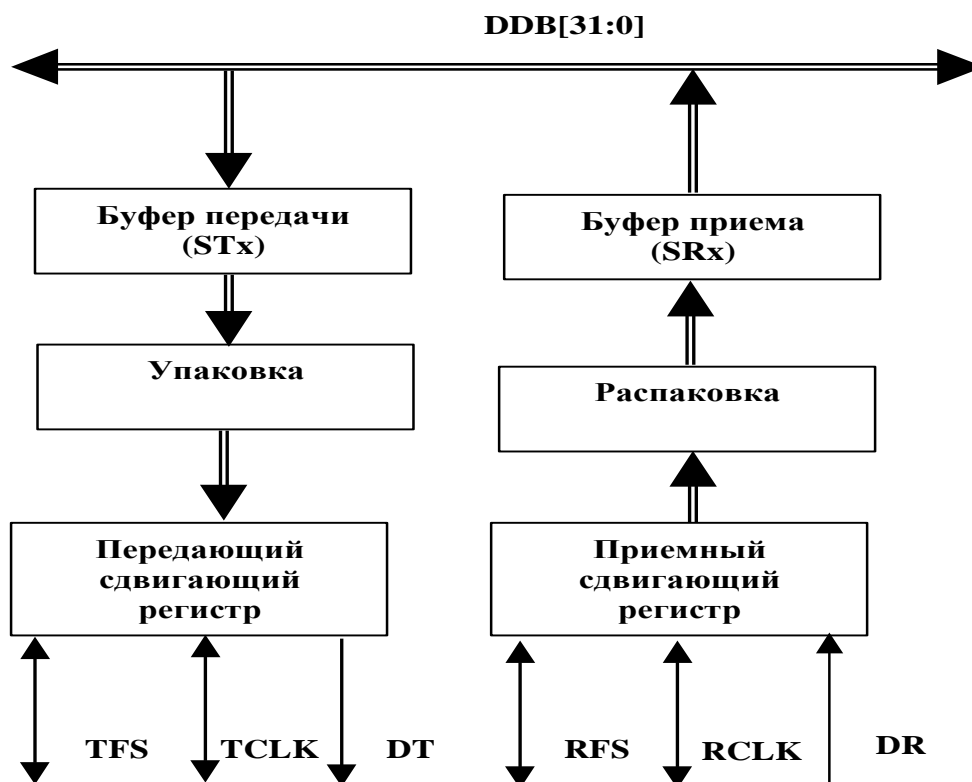
- обеспечивает независимые функции передачи и приема данных;
- передает слова данных длиной от 3 до 32 бит младшим или старшим битом вперед;
- используются двойная буферизация передаваемых данных и тройная буферизация принимаемых данных;
- частота последовательной передачи и приема и кадровая синхронизация может генерироваться самостоятельно или приниматься от внешних источников;
- выполняет однословный обмен данными с внутренней памятью по прерываниям под управлением CPU;
- выполняет обмен блоками данных при помощи DMA;
- имеет многоканальный режим работы для интерфейсов с временным разделением (TMD).

В Таблица 11.1. описаны внешние выводы порта обмена последовательным кодом.

**Таблица 11.1. Выводы порта обмена последовательным кодом**

Название вывода	Тип вывода	Описание
TCLK	IO	Частота передаваемых данных
DT	O	Передаваемые данные
TFS	IO	Кадровый синхроимпульс передаваемых данных
RCLK	IO	Частота принимаемых данных
DR	I	Принимаемые данные
RFS	IO	Кадровый синхроимпульс принимаемых данных

Структурная схема порта обмена последовательным кодом приведена на Рисунок 11.1.



**Рисунок 11.1. Структурная схема порта обмена последовательным кодом**

Порт обмена последовательным кодом состоит из передающей и приемной частей.

Данные для передачи записываются в буфер STx. Затем данные автоматически переписываются в передающий сдвигающий регистр и выдвигаются на выходной вывод DT порта синхронно с тактовой частотой TCLK. Если используется кадровая синхронизация, то сигнал TFS индицирует начало передачи последовательного кода. Вывод DT находится в активном состоянии, если порт активизирован для передачи данных (бит TEN=1 в регистре STCTL), или во время активного временного слота в многоканальном режиме.

При приеме данные вдвигаются в порт с вывода DR синхронно с частотой RCLK. Если используется кадровая синхронизация, то сигнал RFS сигнализирует о начале слова. Когда все слово вдвинуто, оно автоматически переписывается в буфер SRx.

## 11.2 Регистры

### 11.2.1 Общие положения

Перечень регистров порта обмена последовательным кодом приведен в Таблица 11.2.

**Таблица 11.2**

Условное обозначение регистра	Название регистра
STx	Буфер передачи данных
SRx	Буфер приема данных
STCTL	Регистр управления передачей данных
SRCTL	Регистр управления приемом данных
TDIV	Регистр коэффициентов деления при передаче данных
RDIV	Регистр коэффициентов деления при приеме данных
MTCS	Выбор канала передачи данным в многоканальном режиме
MRCS	Выбор канала приема данным в многоканальном режиме
KEYWD	Регистр кода сравнения
KEYMASK	Регистр маски сравнения
MRCE	Выбор канала для сравнения принимаемых данных

### 11.2.2 Буфер передачи STx

Буфер передачи STx является буфером FIFO на два 32-разрядных слова: выходной регистр данных и выходной сдвигающий регистр. Два 32-разрядных слова могут быть сразу записаны в буфер STx, если он был до этого пуст.

Буфер передачи STx генерирует прерывание (бит SportT в регистре QSTR) при следующих условиях:

- хотя бы один из битов TEN (STCTL[0]) или MCE (SRCTL[23]) имеют единичное состояние;
- выходной регистр данных пуст. Данный регистр пуст после начального включения или после передачи его содержимого в выходной сдвигающий регистр;
- соответствующий канал DMA не активизирован;
- данное прерывание не замаскировано.

Данное прерывание формируется в момент активизации последовательного порта на передачу при пустом буфере STx, или в момент переписи содержимого выходного регистра данных в выходной сдвигающий регистр. Прерывание, генерируемое буфером передачи, сигнализирует о том, что буфер STx готов принять следующее слово. Прерывание от буфера передачи сбрасывается в момент записи в него слова данных.

Бит состояния TUVF в регистре STCTL устанавливается, если сформирован сигнал кадровой синхронизации, а в буфер Tx не загружены новые данные. Этот бит может быть обнулен только посредством деактивизации данного порта (TEN=0). В многоканальном режиме бит TUVF всегда равен нулю.

### 11.2.3 Буфер приема SRx

Буфер приема SRx является буфером FIFO на три 32-разрядных слова: два входных регистра данных и входной сдвигающий регистр. Два принятых 32-разрядных слова могут храниться в буфере SRx, пока вдвигается третье слово. Третье слово затирает второе, если оно не было считано из буфера SRx (CPU или DMA). Если это произойдет, устанавливается бит состояния ROVF в регистре SRCTL. Этот бит может быть обнулен только посредством деактивизации данного порта. Почти три полных слова могут быть приняты до того, как бит ROVF может быть установлен. Бит ROVF используется в одноканальном и многоканальном режиме.

В момент окончания приема слова данных в буфер SRx генерируется прерывание, если оно разрешено, и соответствующий канал DMA не активизирован. Данное прерывание сбрасывается после чтения слова данных из буфера SRx.

### 11.2.4 Регистр управления передачей данных STCTL

Формат регистра STCTL приведен в Таблица 11.3.

**Таблица 11.3. Формат регистра STCTL**

Номер разряда	Условное обозначение	Назначение
0	TEN	Разрешение передачи: 0 – передача запрещена; 1 – передача разрешена.
2:1	-	Резерв
3	TENDN	Выбор порядка передаваемых бит данных (endian): 0 – передача осуществляется старшими разрядами вперед (little endian); 1- передача осуществляется младшими разрядами вперед (big endian).
8:4	TLEN	Длина передаваемого слова. Это поле определяет длину слова в битах (на единицу больше чем код TLEN). Длина слова может быть от 3 бит (TLEN = 2) до 32 бит (TLEN = 31).
9	TPACK	Разрешение распаковки передаваемых данных: 1 – разрешена распаковка 32 битного слова в буфере Tx перед его передачей, в два слова, разрядность которых 16 бит или меньше. Распаковка выполняется, если длина передаваемых слов данных меньше или равна 16 (определяется полем TLEN); 0 – запрещена распаковка.

Номер разряда	Условное обозначение	Назначение
10	TCLK	Разрешение выдачи внутренней частоты передачи на вывод TCLK: 0 – вывод TCLK является входом; 1 – вывод TCLK является выходом и на него выдается частота, период которой определяется полем TDIV[15:0].
11	-	Резерв
12	TCKRE	Выбор фронта частоты TCLK, по которому осуществляется опрос состояния передаваемых данных и импульса кадровой синхронизации: 0 – по отрицательному фронту; 1 – по положительному фронту.
13	TFSR	Требование приема кадровой синхронизации TFS для передачи каждого слова (при TLAFS = 0): 1 – кадровая синхронизация требуется при передаче каждого слова; 0 – кадровая синхронизация требуется при передаче только первого слова.
14	ITFS	Разрешение выдачи внутреннего сигнала кадровой синхронизации TFS: 0 – вывод TFS является входом; 1 – вывод TFS является выходом и на него выдается сигнал кадровой синхронизации, период которого определяется полем TDIV[31:16].
15	DITFS	Разрешение выдачи внутреннего кадрового синхроимпульса TFS вне зависимости от наличия данных в буфере STx (при TLAFS = 0): 1 – разрешение; 0 – запрещение. Действует только при TLAFS = 0. При TLAFS = 1 внутренний кадровый синхроимпульс TFS передается только при наличии данных в буфере STx.
16	LTFS	Выбор активного уровня импульса кадровой синхронизации при передаче данных: 1 – импульс кадровой синхронизации имеет активный низкий уровень; 0 – импульс кадровой синхронизации имеет активный высокий уровень.
17	TLAFS	Выбор режима кадровой синхронизации при передаче данных: 0 – режим ранней кадровой синхронизации; 1 – режим поздней кадровой синхронизации.
19:18	-	Резерв
23:20	MFD	Выбор задержки начала передачи данных от импульса кадровой синхронизации при многоканальном режиме работы. При MFD=0

Номер разряда	Условное обозначение	Назначение
		TFS и первый передаваемый бит совпадают. Максимальная величина MFD - 15.
28:24	CHNL	Номер текущего канала при многоканальном режиме работы. Это поле содержит инкрементирующий счетчик по модулю NCH. Доступен только по чтению
29	TUVF	Признак недозагрузки буфера STx. Устанавливается в «1» при DITFS = 1, если поступил TFS, а буфер STx пуст. В многоканальном режиме не устанавливается (всегда равен нулю). Доступен только по чтению. Обнуляется только при TEN = 0.
31:30	TXS	Состояние буфера STx: 00 – буфер пуст; 10 – буфер частично полон; 11 – буфер полон. Доступен только по чтению.

В многоканальном режиме работы биты TEN, TFSR, ITFS, TLAFS, DITFS должны иметь нулевое состояние.

Перед записью в регистр STCTL нового значения, в него предварительно необходимо записать все нули. Исходное состояние регистра STCTL - все нули. При TEN=0, биты CHNL, TUVF обнуляются. Признак TUVF устанавливается в одноканальном режиме работы, если сформирован сигнал TFS (самим портом или внешним источником), а буфер STx пуст. Если установлен режим генерации внутреннего TFS (ITFS=1), то при DITFS=0 TFS формируется только в том случае, если буфер STx не пуст. То есть формирование TFS синхронизируется посредством записи данных в буфер STx. При DITFS=1 TFS формируется вне зависимости от наличия данных в буфере STx.

### 11.2.5 Регистр управления приемом данных SRCTL

Формат регистра SRCTL приведен в Таблица 11.4.

**Таблица 11.4. Формат регистра SRCTL**

Номер разряда	Условное обозначение	Назначение
0	REN	Разрешение приема данных: 0 – прием запрещен; 1 – прием разрешен.
1	-	Резерв



Номер разряда	Условное обозначение	Назначение
2	DTYPE	Тип данных. Если длина принимаемых слов меньше 32 бит, то значащие биты размещаются в младших разрядах буфера Rx, а состояние старших разрядов определяется битом DTYPE следующим образом: 0 – старшие разряды имеют нулевое состояние (расширение нулями); 1 – старшие разряды имеют состояние старшего бита принятого слова (расширение знаком).
3	RENDN	Выбор порядка приема бит данных (endian): 0 – прием осуществляется старшими разрядами вперед (little endian); 1 – прием осуществляется младшими разрядами вперед (big endian).
8:4	RLEN	Длина принимаемого слова. Это поле определяет длину слова в битах (на единицу больше чем код RLEN). Длина слова может быть от 3 бит (RLEN = 2) до 32 бит (RLEN = 31).
9	RPACK	Разрешение упаковки принимаемых данных: 1 – разрешена упаковка каждой пары принимаемых слов данных, длина которых меньше или равна 16 бит, в 32-битное, перед записью в буфер SRx; 0 – запрещена упаковка.
10	RCLK	Разрешение выдачи внутренней частоты передачи на вывод RCLK: 0 – вывод RCLK является входом; 1 – вывод RCLK является выходом и на него выдается частота, период которой определяется полем RDIV[15:0].
11	-	Резерв
12	RCKRE	Выбор фронта частоты RCLK, по которому осуществляется опрос состояния передаваемых данных и импульса кадровой синхронизации: 0 – по отрицательному фронту; 1 – по положительному фронту.

Номер разряда	Условное обозначение	Назначение
13	RFSR	Требование кадровой синхронизации при приеме каждого слова: 1 – кадровая синхронизация требуется при приеме каждого слова; 0 – кадровая синхронизация требуется при приеме только первого слова.
14	IRFS	Разрешение выдачи внутреннего сигнала кадровой синхронизации RFS: 0 – вывод RFS является входом; 1 – вывод RFS является выходом и на него выдается сигнал кадровой синхронизации, период которого определяется полем RDIV[31:16].
15	IMODE	Разрешение сравнения кода принятых данных в многоканальном режиме работы порта: 0 – запрещение сравнения; 1 – разрешение сравнения.
16	LRFS	Выбор активного уровня импульса кадровой синхронизации при приеме данных: 1 – импульс кадровой синхронизации имеет активный низкий уровень; 0 – импульс кадровой синхронизации имеет активный высокий уровень.
17	RLAFS	Выбор режима кадровой синхронизации при приеме данных: 0 – режим ранней кадровой синхронизации; 1 – режим поздней кадровой синхронизации.
19:18	-	Резерв
20	IMAT	Выбор режима сравнения принятых данных в многоканальном режиме работы порта: 0 – принятые данные записываются в буфер Rx, если сравнение произошло не успешно (т.е. сравниваемые данные не совпали); 1 – принятые данные записываются в буфер Rx, если сравнение произошло успешно.
21	-	Резерв

Номер разряда	Условное обозначение	Назначение
22	SPL	Разрешение замыкания внутренней петли данных: 0 – обычный режим работы; 1 – сигналы приемной части порта DR, RCLK, RFS внутренне объединяются с сигналами передающей части порта DT, TCLK, TFS, которые становятся выходами.
23	MCE	Разрешения многоканального режима работы: 0 – режим запрещен; 1 – режим разрешен.
28:24	NCH	Число временных каналов при многоканальном режиме работы порта. Число каналов равно коду в этом поле, увеличенному на единицу. Число каналов может быть от 1 при NCH=0 до 32 при NCH=31
29	ROVF	Признак переполнения буфера Rx. Доступен только по чтению. Обнуляется только при REN = 0.
31:30	RXS	Состояние буфера SRx: 00 - буфер пуст; 10 - буфер частично полон; 11 - буфер полон. Доступен только по чтению.

При многоканальном режиме работы биты SPL, REN, RFSR, RLAFS должны иметь нулевое состояние.

Перед записью в регистр SRCTL нового значения, в него предварительно необходимо записать все нули.

Исходное состояние регистра SRCTL - все нули.

### 11.2.6 Регистр коэффициентов деления при передаче данных TDIV

Формат регистра TDIV приведен в Таблица 11.5.

**Таблица 11.5. Формат регистра TDIV**

Номер разряда	Условное обозначение	Назначение
15:0	TCLKDIV	Определяет период частоты TCLK.
31:16	TFSDIV	Определяет период частоты формирования кадрового синхрои импульса TFS.

Период частоты TCLK вычисляется по формуле:

$$\text{период частоты CLK} * 2 * ((\text{содержимое поля TCLKDIV}) + 1)$$

При выборе данной частоты необходимо учитывать системные ограничения.

Период формирования кадрового синхроимпульса вычисляется по формуле:

$$\text{период частоты TCLK} * ((\text{содержимое поля TFSDIV}) + 1)$$

При TFSDIV=0 кадровый синхроимпульс постоянно активен. Величина TFSDIV не должна быть меньше, чем длина слова минус 1.

Если порт SPORT не используется, то делитель TFSDIV может быть использован как делитель внешней частоты, или для генерации периодических импульсов или прерывания. Для выполнения этих функций SPORT должен быть активизирован.

### 11.2.7 Регистр коэффициентов деления при приеме данных RDIV

Формат регистра RDIV приведен в Таблица 11.6.

**Таблица 11.6. Формат регистра TDIV**

Номер разряда	Условное обозначение	Назначение
15:0	RCLKDIV	Определяет период частоты RCLK.
31:16	RFSDIV	Определяет период частоты формирования кадрового синхроимпульса RFS.

Период частоты RCLK вычисляется по формуле:

$$\text{период частоты CLK} * 2 * ((\text{содержимое поля RCLKDIV}) + 1)$$

При выборе данной частоты необходимо учитывать системные ограничения.

Период формирования кадрового синхроимпульса вычисляется по формуле:

$$\text{период частоты RCLK} * ((\text{содержимое поля RFSDIV}) + 1)$$

При RFSDIV=0 кадровый синхроимпульс постоянно активен. Величина RFSDIV не должна быть меньше, чем длина слова минус 1.

Если порт SPORT не используется, то делитель RFSDIV может быть использован как делитель внешней частоты, или для генерации периодических импульсов или прерывания. Для выполнения этих функций SPORT должен быть активизирован.

### 11.2.8 Регистры выбора канала в многоканальном режиме

Перечень регистров выбора канала в многоканальном режиме приведен в Таблица 11.7.

**Таблица 11.7. Регистры выбора канала в многоканальном режиме**

Условное обозначение регистра	Название регистра
MTCS	Выбор канала для передачи данных
MRCS	Выбор канала для приема данных
MRCE	Выбор канала для сравнения принимаемых данных

Все регистры выбора канала в многоканальном режиме являются 32-разрядными, каждый бит соответствует своему каналу. Исходное состояние регистров – нули.

При единичном состоянии бита в регистре MTCS последовательному порту разрешается передавать слово в соответствующем временном канале. При нулевом

состоянии бита в регистре MTCS последовательному порту запрещается передавать слово в соответствующем временном канале. В этом временном канале вывод DT порта находится в третьем состоянии. В регистре MTCS может быть установлено любое число единиц.

При единичном состоянии бита в регистре MRCS последовательному порту разрешается принимать слово в соответствующем временном канале. Принятое слово загружается в буфер Rx. При нулевом состоянии бита в регистре MRCS последовательному порту запрещается принимать слово в соответствующем временном канале. То есть слово игнорируется. В регистре MRCS может быть установлено любое число единиц.

Работа регистра MRCE разрешается, если разрешено сравнение принимаемых слов данных в соответствии с содержимым регистров KEYWD и KEYMASK, то есть, если бит IMODE в регистре SRCTL имеет единичное состояние. При единичном состоянии бита в регистре MRCE последовательному порту разрешается сравнивать принимаемое слово в соответствующем разрешенном временном канале. Принятое слово загружается в буфер Rx. При нулевом состоянии бита в регистре MRCE последовательный порт в соответствующем временном интервале принимает все слова данных. То есть, сравнения не производится. В регистре MRCE может быть установлено любое число единиц.

### 11.2.9 Регистры сравнения принимаемых данных в многоканальном режиме

Перечень регистров сравнения принимаемых данных в многоканальном режиме приведен в Таблица 11.8.

**Таблица 11.8. Регистры выбора канала в многоканальном режиме**

Условное обозначение регистра	Название регистра
KEYWD	Регистр сравнения
KEYMASK	Регистр маски

Регистры являются 32-разрядными. Исходное состояние регистров неопределено.

Регистр KEYWD содержит образец для сравнения с принятым словом данных.

Регистр KEYMASK указывает, сравнение каких бит в принятом слове разрешено. При нулевом состоянии бита в регистре KEYMASK разрешается сравнение соответствующего бита в принятом слове данных и регистре KEYWD. При единичном состоянии бита в регистре KEYMASK запрещается (маскируется) сравнение соответствующего бита в принятом слове данных и регистре KEYWD, то есть состояние бита не анализируется.

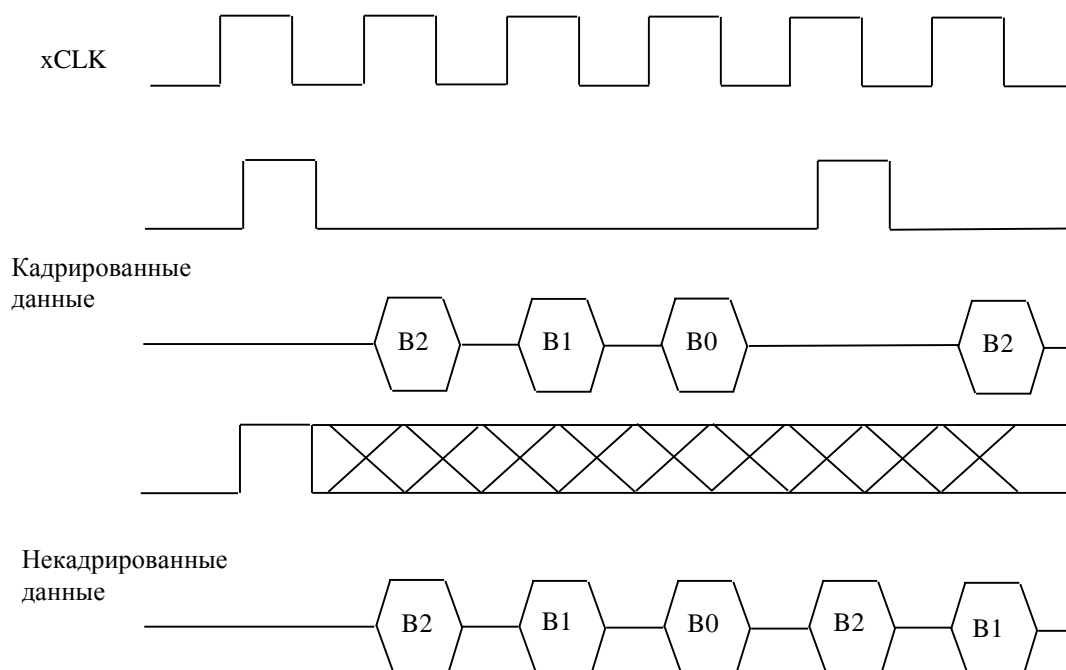
## 11.3 Одноканальный режим работы

В одноканальном режиме работы передающая и приемная части последовательного порта работают отдельно и независимо. Режимы передачи и приема слов данных могут быть различны.

Для синхронизации передачи данных формируются кадровые синхроимпульсы TFS. При TFSR=1 (кадрированные данные) каждое слово сопровождается кадровым

синхроимпульсом. При  $TFSR=0$  (некадрированные данные) кадровый синхроимпульс используется для инициализации всего процесса передачи данных и формируется только один раз перед передачей первого бита информации. В этом случае, данные по каналу связи идут одним потоком.

На Рисунок 11.2 приведены временные диаграммы передачи кадрированных и некадрированных данных.



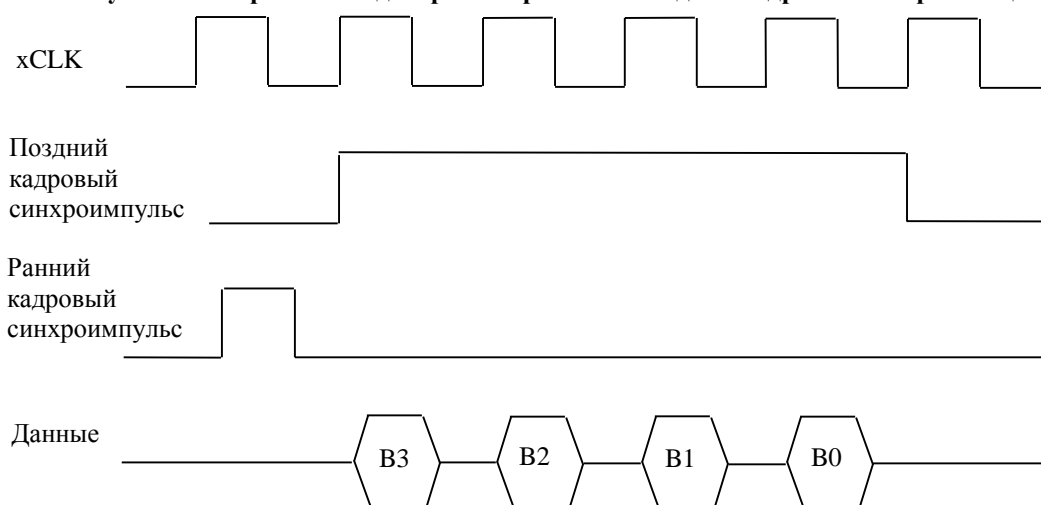
**Рисунок 11.2. Временные диаграммы передачи кадрированных и некадрированных данных**

Аналогично, для синхронизации приема данных формируются кадровые синхроимпульсы RFS. При  $RFSR=1$  каждое слово сопровождается кадровым синхроимпульсом. При  $RFSR=0$  кадровый синхроимпульс используется для инициализации всего процесса приема данных и формируется только один раз перед приемом первого бита информации. В этом случае, данные по каналу связи идут одним потоком.

Кадровые синхроимпульсы TFS и RFS могут формироваться самим портом или поступать от внешнего источника.

При работе последовательного порта может использоваться ранняя или поздняя кадровая синхронизация. Временные диаграммы ранней и поздней кадровой синхронизации приведены на Рисунок 11.3.

**Рисунок 11.3. Временные диаграммы ранней и поздней кадровой синхронизации**



Для настройки передающей части порта в одноканальном режиме необходимо в регистре STCTL выбрать необходимые параметры передачи и установить в единичное состояние бит TEN.

Для обеспечения приема данных в одноканальном режиме необходимо выбрать параметры приема и установить в единичное состояние бит REN.

SPORT формирует ранние кадровые синхроимпульсы TFS при ITFS = 1, если:

- DITFS = 1;
- TLAFS = 0 и буфер STx не пуст.

SPORT формирует поздние кадровые синхроимпульсы TFS при ITFS = 1, если TLAFS = 1 и буфер STx не пуст.

В одноканальном режиме передача очередного слова данных не зависит от наличия данных в буфере STx и выполняется следующим образом:

- если TLAFS = 1, то передача осуществляется по каждому входному сигналу TFS и не зависит от TFSR;
- если TLAFS = 0 и TFSR = 1, то передача осуществляется по каждому входному сигналу TFS;
- если TLAFS = 0 и TFSR = 0, то передача первого слова данных осуществляется по входному сигналу TFS. Последующие слова данных передаются вслед за первым непрерывно. Для того, чтобы прекратить передачу слов данных необходимо установить TEN = 0.

SPORT формирует ранние кадровые синхроимпульсы RFS при IRFS = 1, RLAFS = 0.

SPORT формирует поздние кадровые синхроимпульсы RFS при IRFS = 1, RLAFS = 1.

В одноканальном режиме прием очередного слова данных выполняется следующим образом:

- если RLAFS = 1, то прием осуществляется по каждому входному сигналу RFS и не зависит от RFSR;

- если  $RLAFS = 0$  и  $RFSR = 1$ , то прием осуществляется по каждому входному сигналу RFS;
- если  $RLAFS = 0$  и  $RFSR = 0$ , то прием первого слова данных осуществляется по входному сигналу RFS. Последующие слова данных принимаются вслед за первым непрерывно. Для того, чтобы прекратить прием слов данных необходимо установить  $REN = 0$ .

При использовании DMA SPORT не обеспечивает передачу данных по внешним сигналам TCLK, если они асинхронны по отношению к системной тактовой частоте данной микросхемы. Это относится как к одноканальному, так и к многоканальному режимам работы.

Если требуется передавать данные по внешним асинхронным сигналам TCLK, то эти данные необходимо записывать в буфер STx программно. Запись необходимо выполнять тогда, когда буфер STx пуст (в регистре STCTL поле TXS = 00). Данный алгоритм обмена может быть использован в случае, если CPU успеет зафиксировать TXS = 00 и записать данные в буфер STx за время меньше, чем период TCLK.

Прием данных по внешним асинхронным сигналам RCLK обеспечивается как с DMA, так и без него.

## 11.4 Режим петли

Режим петли используется для тестирования работы последовательного порта.

В этом режиме сигналы приемной части порта DR, RCLK, RFS внутренне соединяются с сигналами передающей части порта DT, TCLK, TFS. При этом выводы DT, TCLK, TFS переходят в активное состояние.

В режиме петли должен быть разрешены режимы генерации внутренней частоты передачи и внутреннего кадрового синхроимпульса передачи.

Проверка многоканального режима работы в режиме петли не обеспечивается.

Для включения последовательного порта в режим петли необходимо:

- в регистрах STCTL и SRCTL установить параметры передачи: биты TENDN, TLEN, TFSR, RENDN, RLEN, RFSR, TCKRE. Эти параметры должны быть одинаковы для передающей и приемной частей порта;
- в регистре SRCTL установить в единичное состояние биты REN, SPL.
- в регистре STCTL установить в единичное состояние биты TICLK, ITFS, TEN, а биты IRFS, RICLK – в нулевое состояние.

Сначала определяется состояние регистра SRCTL, а затем – регистра STCTL.

## 11.5 Многоканальный режим работы

Последовательный порт обеспечивает многоканальный режим работы, который позволяет обмениваться данными в системах с временным мультиплексированием (TDM time-division-multiplexed). В многоканальной системе каждое слово данных передается в своем временном канале (слоте). Многоканальный режим работы включается при MCE=1.



В многоканальной системе данные передаются кадрами. Кадр содержит число слов, равное числу временных каналов. Признаком начала процесса передачи кадров данных является первый сигнал кадровой синхронизации RFS, принятый после установки MCE = 1. RFS может генерироваться одним из последовательных портов многоканальной системы, или формироваться внешним источником кадровой синхронизации.

В многоканальном режиме приемная и передающая части последовательного порта работают одновременно и используют общее оборудование.

В многоканальном режиме сигнал TFS является признаком того, что данный последовательный порт находится в режиме передачи информации и вывод DT имеет активное состояние.

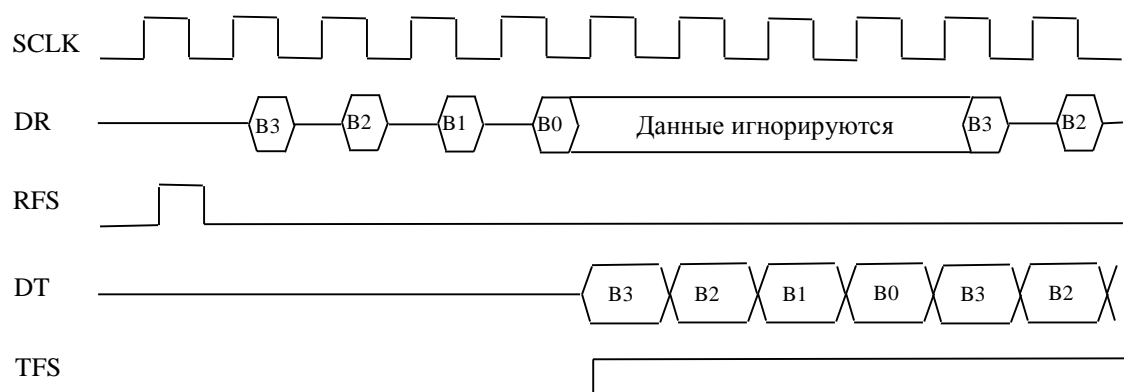
Последовательный порт автоматически выбирает временной канал. Имеется 32 канала для передачи или приема данных. Другими словами, последовательный порт в каждом временном канале может выполнять следующие действия:

- передавать данные;
- принимать данные;
- передавать и принимать данные;
- не принимать и не передавать данные.

В многоканальном режиме работы:

- выводы RFS всех портов многоканальной системы объединяются. Источником RFS может быть один из портов. Для этого в его регистре SRCTL необходимо установить IRFS = 1. В остальных портах - IRFS = 0;
- сигналы DT, DR всех портов объединяются в один узел;
- сигналы TCLK, RCLK всех портов объединяются в один узел. Вывод TCLK является входом. Источником RCLK может быть один из портов. Для этого в его регистре SRCTL необходимо установить RICK = 1. В остальных портах RICK = 0.
- вывод DT переводится в активное состояние (из высокоимпедансного состояния) только в разрешенном временном канале;
- вывод TFS обычно остается не подсоединенным;

На Рисунок 11.4. приведена временная диаграмма приема и передачи данных в многоканальном режиме. В данном примере порт выполняет прием данных во временном канале 0 и передает данные во временные каналы 1 и 2.



**Рисунок 11.4. Временная диаграмма приема и передачи данных в многоканальном режиме**

Для обеспечения работы данного последовательного порта в многоканальном режиме необходимо:

- в поле NCH регистра SRCTL установить число каналов, которое используется в данной системе;
- в поле MFD регистра STCTL установить величину задержки между импульсом кадровой синхронизации и началом передачи первого бита данных. Задержка измеряется в периодах частоты передачи данных. При MFD=0 кадровой синхроимпульс по времени совпадает с первым битом. Максимальная величина MFD равна 15. Программирование этой задержки позволяет работать по разным протоколам передачи данных. При работе на максимальной частоте передачи данных (CLK/2) в MFD должен быть установлен код не менее 1;
- в регистре MTCS установить в единичное состояние биты временных каналов, в которых требуется передавать данные;
- в регистре MRCS установить в единичное состояние биты временных каналов, в которых требуется принимать данные;
- в регистре SRCTL определить состояние бит IMODE и IMAT, то есть установить режим сравнения принимаемых данных (при необходимости);
- в регистрах STCTL и SRCTL установить параметры передачи и приема слов (биты TENDN, TLEN, RENDN, RLEN, TCKRE, RCKRE, LTFS, LRFS). Следует отметить, что для последовательного порта параметры передачи и приема в многоканальном режиме должны быть одинаковы;
- в регистры KEYWD, KEYMASK MRCE записать необходимые коды, если данные необходимо принимать в режиме сравнения;
- в регистре SRCTL установить в единичное состояние бит IRFS, если данный последовательный порт должен формировать кадровой синхроимпульс RFS;
- биты TEN, TFSR, ITFS, TLAFS, DITFS, REN, RFSR, RLAFS в регистрах STCTL и SRCTL должны иметь нулевое состояние;
- в регистре SRCTL установить в единичное состояние бит MCE.

Номер временного канала, который в данный момент времени активен, содержится в доступном только по чтению поле CHNL регистра STCTL. Это поле содержит инкрементирующий счетчик по модулю NCH.

Если в многоканальном режиме для данного порта наступил активный временной канал для передачи, то она выполнится вне зависимости от наличия необходимых данных в буфере STx. Признак недозагрузки буфера STx (TUVF) в многоканальном режиме не устанавливается.

В многоканальном режиме признак переполнения буфера Rx (ROVF) функционирует.

В многоканальном режиме работы прием данных можно выполнять со сравнением, используя регистры KEYWD, KEYMASK и MRCE. При этом каждое принятое слово данных сравнивается с содержимым регистра KEYWD с использованием маски в регистре KEYMASK. Режим сравнения определяется состоянием бит IMODE и IMAT в регистре SRCTL. Если сравнение произошло неуспешно, то принятое слово данных в буфер SRx не записывается при бите IMAT, установленном в единицу. Если бит IMAT установлен в ноль и сравниваемые данные не совпали (сравнение произошло неуспешно), то принятое слово данных в буфер SRx записывается.

В многоканальном режиме работы при использовании DMA SPORT не обеспечивает передачу данных по внешним сигналам TCLK, если они асинхронны по отношению к системной тактовой частоте данной микросхемы. То есть, чтобы использовать DMA для передачи данных, для синхронизации всех микропроцессоров многоканальной системы необходимо использовать одну частоту.

Если требуется передавать данные по внешним асинхронным сигналам TCLK, то эти данные необходимо записывать в буфер STx программно. Запись необходимо выполнять тогда, когда буфер STx пуст (в регистре STCTL поле TXS = 00). Данный алгоритм обмена может быть использован в случае, если CPU успеет зафиксировать TXS = 00 и записать данные в буфер STx за время меньше, чем период TCLK.

Программную передачу данных можно организовать и по прерываниям. Для формирования прерывания при пустом буфере STx (в регистре STCTL поле TXS = 00) можно использовать приемную частоту второго SPORT. Для этого его тоже необходимо включить в многоканальную сеть (подключить выводы RFS, RCLK, TCLK) и настроить на прием в тех же временных каналах, в которых передает первый. При этом, как только будет произведена передача слова первым SPORT, от второго SPORT сформируется прерывание по приему данных, по которому необходимо сделать фиктивное чтение их буфера SRx, а в буфер STx первого SPORT записать очередное слово данных. Первый раз запись данных в буфер STx выполняется сразу после инициализации SPORT на передачу. Необходимо учитывать, что при таком способе формирования прерываний первый SPORT не должен передавать данные в двух последовательных временных каналах.

Прием данных по внешним асинхронным сигналам RCLK обеспечивается как с DMA, так и без него.

## 11.6 DMA последовательного порта

С последовательным портом могут быть связаны два канала DMA:

- SportTxCh – передача данных в последовательный канал;
- SportRxCh – прием данных из последовательного канала.

## 11.7 Прерывания от последовательного порта

Последовательный порт формирует прерывания по приему и передаче данных.

Если соответствующий канал DMA активизирован, то прерывания формируются по завершению передачи или приема всего блока данных.

Если соответствующий канал DMA не активизирован, то прерывания формируются по завершению передачи или приема каждого слова данных.

## 12. ЛИНКОВЫЙ ПОРТ

### 12.1 Архитектура линкового порта

Линковый порт имеет следующие основные характеристики:

- частота передачи данных –  $CLK/4$ ,  $CLK/2$  ( $CLK$  – тактовая частота 1892ВМ3Т);
- использована двойная буферизация передаваемых и принимаемых данных;
- выполняет однословный обмен данными по прерываниям под управлением CPU-ядра;
- выполняет обмен блоками данных при помощи DMA;
- по внешнему интерфейсу линковый порт совместим с ADSP-21160.

Структурная схема линкового порта приведена на Рисунок 12.1.

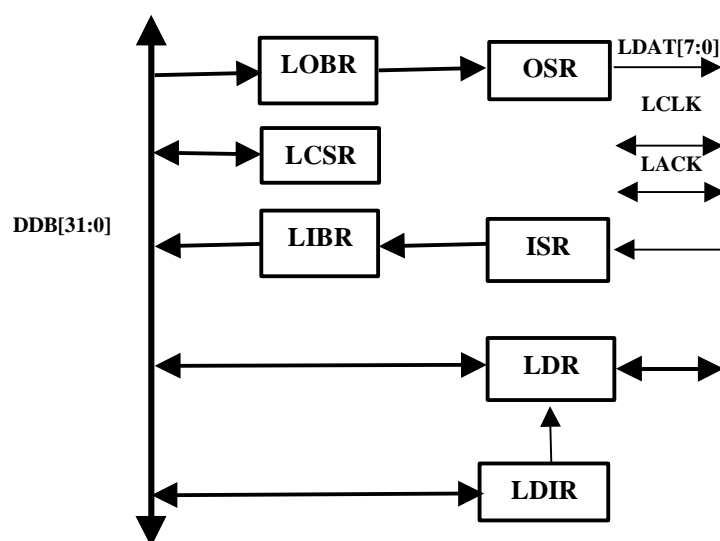


Рисунок 12.1. Структурная схема линкового порта

Передаваемые 32-разрядные данные записываются в выходной буферный регистр (LOBR), а затем аппаратно переписываются в передающий сдвигающий регистр (OSR), если он пуст. После этого, в выходной буферный регистр могут быть записаны очередные данные. Из передающего сдвигающего регистра данные выдаются во внешнюю шину данных тетрадами или байтами.

Из внешней шины данные поступают в приемный сдвигающий регистр (ISR) тетрадами или байтами. После набора 32-разрядного слова он переписывается во входной буферный регистр (LIBR).

Данные передаются, начиная со старшей тетрады или старшего байта.

Если LPORT неактивизирован ( $LEN=0$ ), внешние линии  $LDAT[7:0]$ ,  $LCLK$ ,  $LACK$  можно использовать как 10-разрядный двунаправленный порт ввода-вывода.

В Таблица 12.1. описаны внешние выходы линкового порта.

**Таблица 12.1. Выводы линковых портов**

Название вывода	Тип вывода	Описание
LDAT[3:0]/[7:0]	IO	Внешняя шина данных. Данные по этой шине передаются по положительному фронту сигнала LCLK.
LCLK	IO	Частота передачи данных
LACK	IO	Подтверждение приема

## 12.2 Регистры

### 12.2.1 Общие положения

Перечень регистров порта приведен в Таблица 12.2.

**Таблица 12.2**

Условное обозначение регистра	Название регистра
LTx	Буфер передачи данных
LRx	Буфер приема данных
LCSR	Регистр управления и состояния
LDIR	Регистр управления направлением выводов порта ввода-вывода
LDR	Регистр данных порта ввода-вывода

### 12.2.2 Буфер передачи LTx

Буфер передачи LTx является буфером FIFO на два 32-разрядных слова и состоит из выходного буферного регистра и передающего сдвигающего регистра. Два 32-разрядных слова могут быть сразу записаны в буфер LTx, если он был до этого пуст.

Буфер передачи LTx генерирует прерывание (бит LportTx в регистре QSTR) при следующих условиях:

- бит LTRAN=1;
- выходной регистр данных пуст;
- соответствующий канал DMA не активизирован;
- данное прерывание не замаскировано.

Данное прерывание формируется в момент активизации линкового порта на передачу при пустом буфере LTx, или в момент переписи содержимого выходного регистра данных в выходной сдвигающий регистр. Прерывание, генерируемое буфером

передачи, сигнализирует о том, что буфер LTx готов принять следующее слово. Прерывание от буфера передачи сбрасывается в момент записи в него данных.

Загрузка данных в порт возможна только при активизации порта на передачу.

### 12.2.3 Буфер приема LRx

Буфер приема LRx является буфером FIFO на два 32-разрядных слова и состоит из входного регистра данных и входного буферного регистра. Одно принятое 32-разрядное слово может храниться в буфере LRx, пока вдвигается второе слово.

В момент окончания приема в буфер LRx 32-разрядного слова данных, генерируется прерывание, если оно разрешено, а соответствующий канал DMA не активизирован. Данное прерывание сбрасывается при чтении данных из буфера приема.

Считывание данных из буфера приема возможно только при активизации порта на прием.

### 12.2.4 Регистр управления и состояния LCSR

Формат регистра LCSR приведен в Таблица 12.3.

**Таблица 12.3. Формат регистра LCSR**

Номер разряда	Условное обозначение	Назначение
0	LEN	Разрешение работы порта: 0 – все выходы порта находятся в высокоимпедансном состоянии; 1 – порт работает в соответствии с состоянием бита LTRAN.
1	LTRAN	Режим работы порта: 0 – приемник; 1 – передатчик.
2	LCLK	Управление частотой работы порта: 0 – CLK/4; 1 – CLK/2.
4:3	LSTAT	Состояние буферов Tx или Rx: 00 – буфер пуст; 10 – буфер содержит одно слово данных; 11 – буфер полон.
5	LERR	Ошибка приема или передачи данных: 0 – 32-разрядное слово принято (передано) полностью; 1 – 32-разрядное слово принято (передано) не полностью
6	LDW	Разрядность внешней шины данных: 0 - 4-разряда (32-разрядное слово передается за 8 посылок); 1 - 8-разряда (32-разрядное слово передается за 4 посылки).
7	SRQ_TX	Признак запроса обслуживания на передачу данных

8	SRQ_RX	Признак запроса обслуживания на прием данных
31:9	-	Резерв

Исходное состояние регистра LCSR – нули. Биты LEN, LTRAN, LCLK доступны по записи и чтению, а LSTAT, LERR – только по чтению.

Биты LSTAT, LERR сбрасываются при LEN=0.

## 12.2.5 Регистры порта ввода-вывода

10-разрядный регистр данных порта ввода-вывода (LDR) предназначен для реализации гибкого интерфейса с внешними устройствами. Внешние выходы порта ввода-вывода совмещены с внешними выводами линкового порта.

Соответствие разрядов регистра LDR и внешних линий линкового порта приведено в Таблица 12.4.

Таблица 12.4

Номер разряда Регистра LDR	Внешние выходы LPORT
0	LACK
1	LCLK
9:2	LDAT[7:0]

Настройка направления выводов порта ввода-вывода осуществляется программно при помощи 10-разрядного регистра LDIR. Если разряд этого регистра имеет нулевое состояние, то соответствующий разряд порта ввода-вывода является входом и наоборот. Линии порта ввода-вывода могут быть выходами, если LEN=0.

Исходное состояние регистров LDR, LDIR – нули.

## 12.3 DMA линковых портов

С каждым линковым портом связан канал DMA LportCh. Направление передачи DMA определяется битом LTRAN.

## 12.4 Прерывания от линковых портов

### 12.4.1 Прерывания при приеме и передаче данных

Линковый порт формирует прерывания по приему и передаче данных.

Если обмен данными по линковому порту выполняется программно без использования DMA, то прерывания формируются по завершению передачи или приема каждого 32-разрядного слова данных. При этом, биты RUN, DONE и END регистра CSR соответствующего канала DMA должны иметь нулевое состояние.



Если обмен данными по линковому порту выполняется с использованием DMA, то прерывания формируются в соответствии с п. 8.1.5.

### **12.4.2 Прерывания по запросу обслуживания**

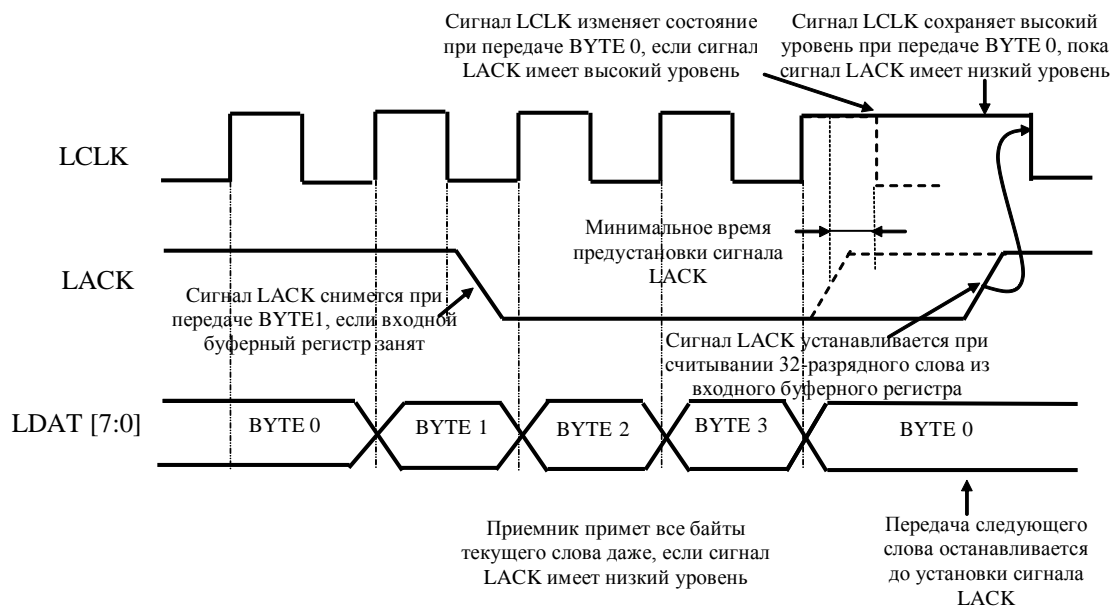
Если линковый порт не активизирован ( $LEN=0$ ), он формирует прерывание по запросу обслуживания, если:

- на внешней шине выставлены данные на прием (активное состояние сигнала LCLK);
- из внешней шины поступил запрос на выдачу данных (активное состояние сигнала LACK).

Данное прерывание сбрасывается после установки  $LEN=1$ .

## 12.5 Временная диаграмма работы линкового порта

Временная диаграмма работы линкового порта приведена на Рисунок 12.2.



**Рисунок 12.2. Временная диаграмма работы линкового порта (LDW=1)**

При LDW=0 передача 32-разрядного слова выполняется за 8 посылок, а при LDW=1 - за 4 посылки. Передатчик изменяет данные LDAT по положительному фронту LCLK, а приемник защелкивает данные по отрицательному фронту.

Исходное состояние сигнала LACK – высокий уровень. Сигнал LACK снимается приемником по заднему фронту LCLK при передаче BYTE1, если его входной буферный регистр занят. При этом приемник примет все байты текущего 32-разрядного слова даже, если сигнал LACK имеет низкий уровень. Сигнал LACK устанавливается при считывании 32-разрядного слова из входного буферного регистра.

Передатчик после выставления BYTE0 анализирует состояние сигнала LACK. Если LACK=1, то LCLK продолжает изменять свое состояние и после BYTE 0 передается BYTE 1 и так далее. Если LACK=0, то LCLK сохраняет высокий уровень при передаче BYTE 0, пока сигнал LACK имеет низкий уровень.

Если линковый порт деактивизирован (LEN=0) сигналы LDAT, LCLK LACK являются входами. Поэтому эти сигналы необходимо привязывать к земле через резисторы 10 кОм. Если порт настроен как передатчик, LDAT и LCLK становятся выходами, а LACK – входом. Если порт настроен как приемник, LDAT и LCLK становятся входами, а LACK – выходом.

### **13. ПОРТ JTAG И ВСТРОЕННЫЕ СРЕДСТВА ОТЛАДКИ ПРОГРАММ**

В данную микросхему встроен порт JTAG, реализованный в соответствии со стандартом IEEE 1149.1 (IEEE Standard Test Access Port and Boundary-Scan Architecture). Этот порт предназначен только для доступа к встроенным средствам отладки программ (OnCD) и не реализует Boundary Scan.

Модуль OnCD обеспечивает:

- выполнение остановки программы CPU по контрольным точкам (Breakpoint);
- выполнение заданного числа команд CPU (трассы) в реальном масштабе времени или пошаговое выполнение команд;
- доступ к адресуемым регистрам и памяти микросхемы.

Для подключения микросхемы к персональному компьютеру через порт JTAG необходимо использовать эмулятор JTAG, предназначенный для работы с данным микропроцессором.

## 14. ЭЛЕКТРИЧЕСКИЕ И ВРЕМЕННЫЕ ПАРАМЕТРЫ

### 14.1 Электропитание

Номинальное значение напряжения электропитания микросхемы:

- $U_{CC1}=3,3$  В (периферия);
- $U_{CC2}=2,5$  В (ядро).

Допустимые отклонения напряжения электропитания микросхемы от номинального значения - не более  $\pm 5\%$ .

При включении на микросхему сначала необходимо подать напряжение электропитания ядра  $U_{CC2}$ , а затем напряжение электропитания периферии  $U_{CC1}$ . Задержка между подачей напряжений электропитания должна быть не более 10 мс. Входные сигналы подаются после подачи напряжений электропитания или одновременно с напряжением электропитания периферии  $U_{CC1}$ . Фронт нарастания напряжений электропитания должен быть не более 5 мс;

При выключении микросхемы необходимо сначала снять входные сигналы, затем напряжение электропитания периферии  $U_{CC1}$ , затем, с задержкой не более 10 мс, напряжение электропитания ядра  $U_{CC2}$ .

Для фильтрации напряжений электропитания микросхемы, необходимо подключить к каждому источнику ( $U_{CC1}$  и  $U_{CC2}$ ) не менее шести высокочастотных конденсаторов номиналом 0,1 мкФ типа СС 0603 Y5V 0,1  $\mu$ F Z 25V. Конденсаторы необходимо разместить по возможности равномерно по периметру корпуса микросхемы между выводами PVDD и GND, а так же CVDD и GND. При этом расстояние между контактами микросхемы и площадками подсоединения конденсаторов должно быть не более 3 мм.

### 14.2 Электрические параметры

Электрические параметры микросхемы приведены в Таблица 14.1.

**Таблица 14.1. Электрические параметры микросхемы**

Наименование параметров, единица измерения, режим измерения	Буквенное обозначение	Норма		Температура °C
		не менее	не более	
Ток потребления статический по цепи PVDD, мА при $U_{CC1}=3,47$ В, $U_{CC2}=2,63$ В, ХТИ=0	$I_{CC1}$	-	3	от -60 до +85
Ток потребления статический по цепи CVDD, мА при $U_{CC1}=3,47$ В, $U_{CC2}=2,63$ В, ХТИ=0	$I_{CC2}$	-	10	от -60 до +85
Ток потребления динамический по цепи CVDD, мА, при $U_{CC1}=3,47$ В, $U_{CC2}=2,63$ В и рабочей частоте 80 МГц	$I_{OCC2}$	-	300	от -60 до +85
Ток утечки высокого и низкого уровня на входе, мкА при $U_{CC1}=3,47$ В и $U_{CC2}=2,63$ В	$I_{IL}$	-	2	от -60 до +85
Выходное напряжение низкого уровня, В при $I_{OL}= 4$ мА, $U_{CC1}=3,47$ В	$U_{OL}$	-	0,4	от -60 до +85
Выходное напряжение высокого уровня, В при $I_{OH} =-$ 2,8 мА, $U_{CC1} =3,13$ В	$U_{OH}$	2,4		от -60 до +85
Входная емкость, пФ	$C_I$	-	10	$25 \pm 10$
Емкость входа/выхода, пФ	$C_{I/O}$	-	10	$25 \pm 10$
Выходная емкость, пФ	$C_O$	-	15	$25 \pm 10$

### 14.3 Динамическая потребляемая мощность

Динамическая потребляемая мощность микросхемы имеет две составляющие: потребление ядра (по цепи CVDD) и потребление выходных драйверов (по цепи PVDD).

Мощность, потребляемая ядром микросхемы по цепи CVDD, зависит от последовательности выполняемых процессорными ядрами команд, от операндов, а также от активности DMA и периферийных устройств. Максимальный ток, потребляемый ядром микросхемы, не превышает 300 мА при внутренней частоте синхронизации 80 МГц.

Мощность, потребляемая выходными драйверами по цепи PVDD, зависит от следующих параметров:

- Число выходных драйверов (O);
- Максимальная частота, на которой выходные драйверы переключаются (F);
- Емкости нагрузки выходных драйверов (C);
- Величина напряжения электропитания выходных драйверов ( $U_{CC1}$ ).

Мощность, потребляемая выходными драйверами по цепи PVDD, определяется следующим уравнением:

$$P_{ext} = O * C * U_{CC1}^2 * F.$$

Рассмотрим для примера расчет мощности, потребляемой выходными драйверами при непрерывной записи данных в память типа SRAM (при  $U_{CC1} = 3,3$  В). Максимальная частота обмена данными со SRAM = CLK/4, где CLK – внутренняя тактовая частота микросхемы (80 МГц). При обращении по произвольным адресам можно предположить, что с частотой CLK/4 изменяются 50% разрядов адреса. Также можно допустить, что каждый цикл изменяются 50% разрядов шины данных. Данные для расчета потребляемой мощности приведены в Таблица 14.2.

**Таблица 14.2**

Название драйвера	Число драйверов	Емкость нагрузки	F, МГц	$U_{CC1}^2$	$P_{ext}$ , мВт
A[31:0]	16	30	20	10,9	100
nWR[3:0]	4	30	20	10,9	25
D[31:0]	16	30	20	10,9	100
SCLK	1	30	80	10,9	25
Итого:					250

То есть, при тактовой частоте 80 МГц и  $C=30$  пФ при непрерывной записи данных в SRAM потребление составляет 250 мВт. При чтении данных из SRAM выходные драйверы не активизируются. Поэтому, если запись данных в SRAM чередуется с чтением, то реальное энергопотребление микросхемы будет существенно меньше.

Оценим мощность, потребляемую драйверами линкового порта при передаче данных. Максимальная частота передачи данных по линковому порту равна 40 МГц. Потребление по LCLK составляет 12 мВт, а потребление по данным (изменяется 50% 8-разрядных данных с частотой 20 МГц) - 24 мВт. Суммарно – 36 мВт.

## 14.4 Предельно-допустимые и предельные электрические режимы эксплуатации

Значения предельно-допустимых и предельных электрических режимов эксплуатации микросхемы приведены в Таблица 14.3.

**Таблица 14.3. Значения предельно-допустимых и предельных электрических режимов эксплуатации**

Наименование параметра, единица измерения	Буквенное обозначение	Норма			
		Предельно допустимый режим		Предельный режим	
		не менее	не более	не менее	не более
Напряжение питания периферии, В	$U_{CC1}$	3,13	3,47	-	3,9
Напряжение питания ядра, В	$U_{CC2}$	2,37	2,63	-	3,0
Входное напряжение высокого уровня, В	$U_{IH}$	2,0	$U_{CC1}+0,2$	-	$U_{CC1}+0,3$
Входное напряжение низкого уровня, В	$U_{IL}$	0,0	0,7	-0,3	-
Напряжение, прикладываемое к выходу микросхемы в состоянии «выключено», В	$U_{OZ}$	0,0	$U_{CC1}+0,1$	-0,3	$U_{CC1}+0,3$
Емкость нагрузки каждого выхода, пФ	$C_L$	-	30	-	50

## 14.5 Временные параметры

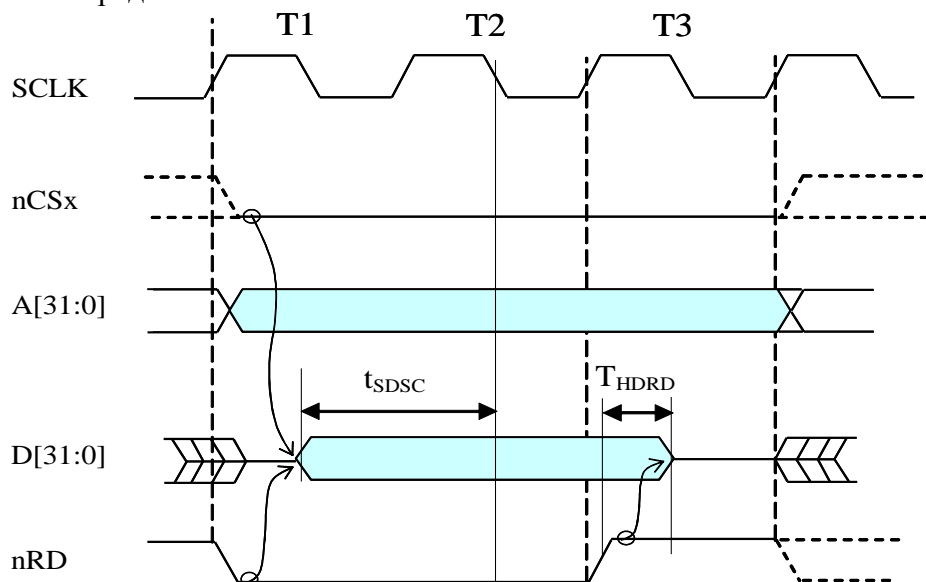
### 14.5.1 Обмен данными с внешней памятью и устройствами

Временные параметры при обмене данными с внешней памятью и устройствами приведены в Таблица 14.4.

**Таблица 14.4. Временные параметры при обмене данными с внешней памятью и устройствами**

Наименование параметра, единица измерения	Буквенное обозначение	Норма		Температура °C
		не менее	не более	
Время задержки выходных сигналов A, D, nWR, nWE, nRD, nCS, SRAS, SCAS, SWE, DQM, CKE, A10, BA, nFLYBY, nOE после переднего фронта частоты SCLK, нс	$t_{DOSC}$	2	5	от -60 до +85
Время предустановки считываемых данных из асинхронной памяти перед задним фронтом частоты SCLK, нс	$t_{SDSC}$	6	-	от -60 до +85
Время удержания считываемых данных из асинхронной памяти после фронта снятия сигнала nRD, нс ( $t_{CLK}$ – период частоты CLK)	$t_{HDRD}$	0	$0,5 t_{CLK}$	от -60 до +85
Время предустановки считываемых данных из синхронной памяти перед передним фронтом частоты SCLK, нс	$t_{SDSC}$	5	-	от -60 до +85
Время удержания считываемых данных из синхронной памяти после переднего фронта частоты SCLK, нс	$t_{HDSC}$	0	$0,5 t_{CLK}$	от -60 до +85

Временная диаграмма при чтении данных из асинхронной памяти приведена на Рисунок 14.1. Считываемые данные фиксируются в микросхеме по заднему фронту частоты SCLK перед снятием сигнала nRD.



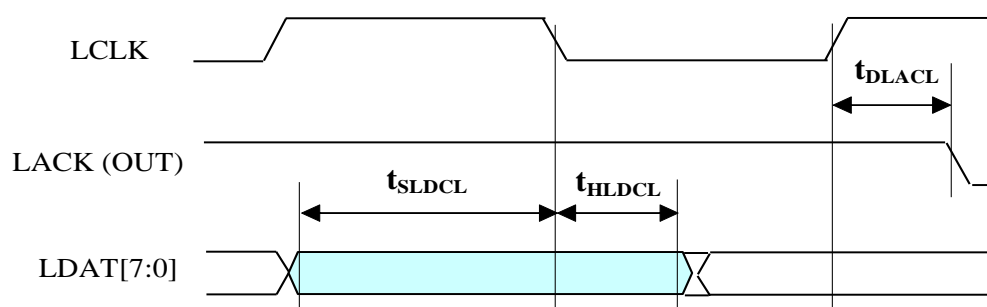
**Рисунок 14.1. Чтение асинхронной памяти без дополнительных тактов ожидания.**

### 14.5.2 Прием и передача данных по линковому порту

Временные параметры при приеме данных по линковому порту приведены в Таблица 14.5 и Рисунок 14.2.

**Таблица 14.5. Временные параметры при приеме данных по линковому порту**

Наименование параметра, единица измерения	Буквенное обозначение	Норма		Температура °С
		не менее	не более	
Время предустановки данных перед задним фронтом частоты LCLK, нс	$t_{SLDCL}$	5	-	от -60 до +85
Время удержания данных после заднего фронта частоты LCLK, нс	$t_{HLDCL}$	3	-	от -60 до +85
Время задержки переключения сигнала LACK с высокого на низкий уровень после заднего фронта частоты LCLK, нс	$t_{DLALC}$	5	15	от -60 до +85
Период частоты LCLK	$t_{CLK}$	$2,05 * t_{CLK}$	-	от -60 до +85


**Рисунок 14.2. Прием данных по линковому порту**

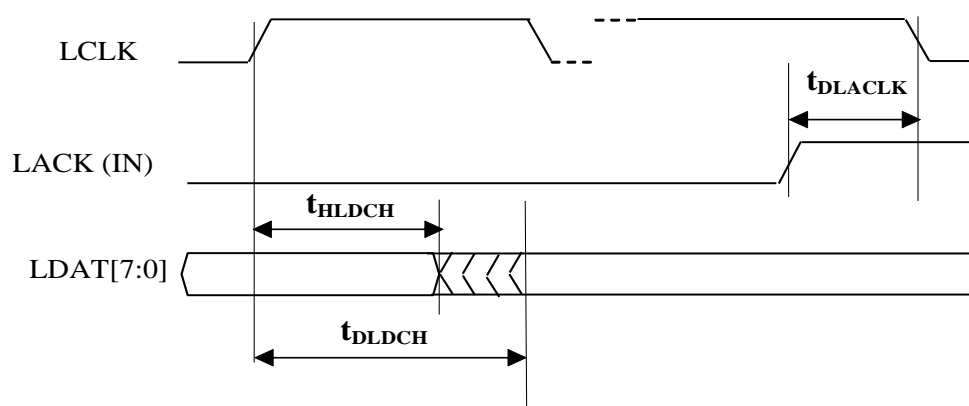
Временные параметры при передаче данных по линковому порту приведены в Таблица 14.6 и

Рисунок 14.3..

**Таблица 14.6. Временные параметры при передаче данных по линковому порту**

Наименование параметра, единица измерения	Буквенное обозначение	Норма		Температура °С
		не менее	не более	
Время задержки данных после переднего фронта частоты LCLK, нс	$t_{DLDC}$	-	10	от -60 до +85
Время удержания данных после переднего фронта частоты LCLK, нс	$t_{HLDCH}$	0	-	от -60 до +85
Время задержки переключения частоты LCLK в низкий уровень, после переключения сигнала LACK с низкого уровня на высокий, нс	$t_{DLACLK}$	5	$t_{CLK} + 5$	от -60 до +85

LCLK неактивна  
(высокий уровень)


**Рисунок 14.3. Передача данных по линковому порту**



### 14.5.3 Прием и передача данных по последовательному порту

Временные параметры при приеме данных по последовательному порту приведены в Таблица 14.7 и Таблица 14.8.

**Таблица 14.7. Временные параметры при приеме данных по последовательному порту (внешняя частота)**

Наименование параметра, единица измерения	Буквенное обозначение	Норма		Температура °С
		не менее	не более	
Время предустановки данных и TFS/RFS перед активным фронтом RCLK, нс	$t_{SRE}$	5	-	от -60 до +85
Время удержания данных и TFS/RFS после активного фронта RCLK, нс	$t_{HRE}$	5	-	от -60 до +85
Период частоты TCLK/RCLK, нс	$t_{SCLK}$	$2t_{CLK}$	-	от -60 до +85

**Таблица 14.8. Временные параметры при приеме данных по последовательному порту (внутренняя частота)**

Наименование параметра, единица измерения	Буквенное обозначение	Норма		Температура °С
		не менее	не более	
Время предустановки данных и TFS/RFS перед активным фронтом RCLK, нс	$t_{SRI}$	9	-	от -60 до +85
Время удержания данных и TFS/RFS после активного фронта RCLK, нс	$t_{HRI}$	3	-	от -60 до +85

Временные параметры при передаче данных по последовательному порту приведены в Таблица 14.9 и Таблица 14.10.

**Таблица 14.9. Временные параметры при передаче данных по последовательному порту (внешняя частота)**

Наименование параметра, единица измерения	Буквенное обозначение	Норма		Температура °С
		не менее	не более	
Время задержки выдачи данных и сигнала TFS после активного фронта TCLK, нс	$t_{DTE}$	-	15	от -60 до +85
Время удержания данных и сигнала TFS после активного фронта TCLK, нс	$t_{HTE}$	0	-	от -60 до +85

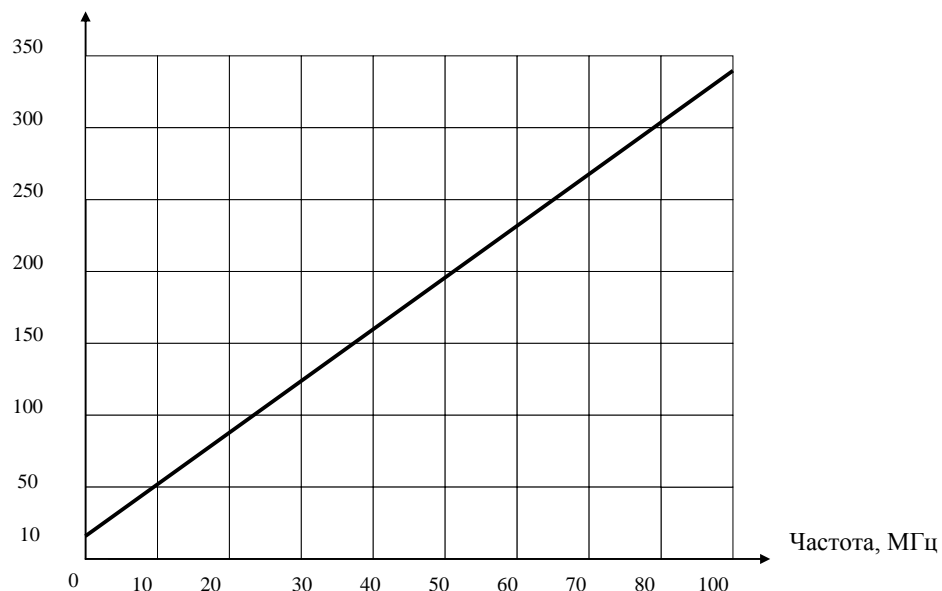
**Таблица 14.10. Временные параметры при передаче данных по последовательному порту (внутренняя частота)**

Наименование параметра, единица измерения	Буквенное обозначение	Норма		Температура °С
		не менее	не более	
Время задержки выдачи данных и сигнала TFS после активного фронта TCLK, нс	$t_{DPI}$	-	5	от -60 до +85
Время удержания данных и сигнала TFS после активного фронта TCLK, нс	$t_{HPI}$	0	-	от -60 до +85

## 14.6 Зависимости основных параметров от режимов и условий эксплуатации

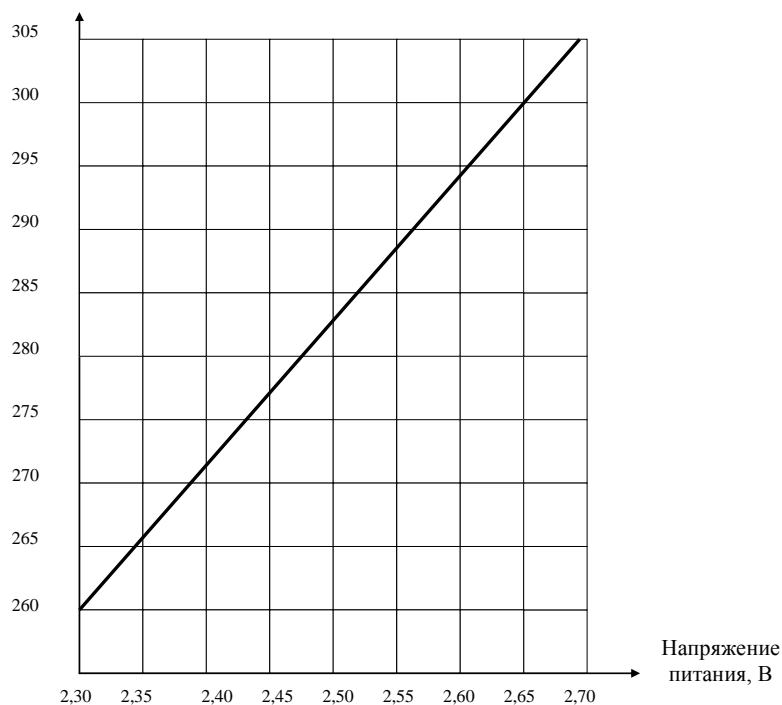
Зависимости основных электрических параметров микросхемы от режимов и условий эксплуатации приведены на Рисунок 14.4-Рисунок 14.7.

Динамический ток потребления, мА



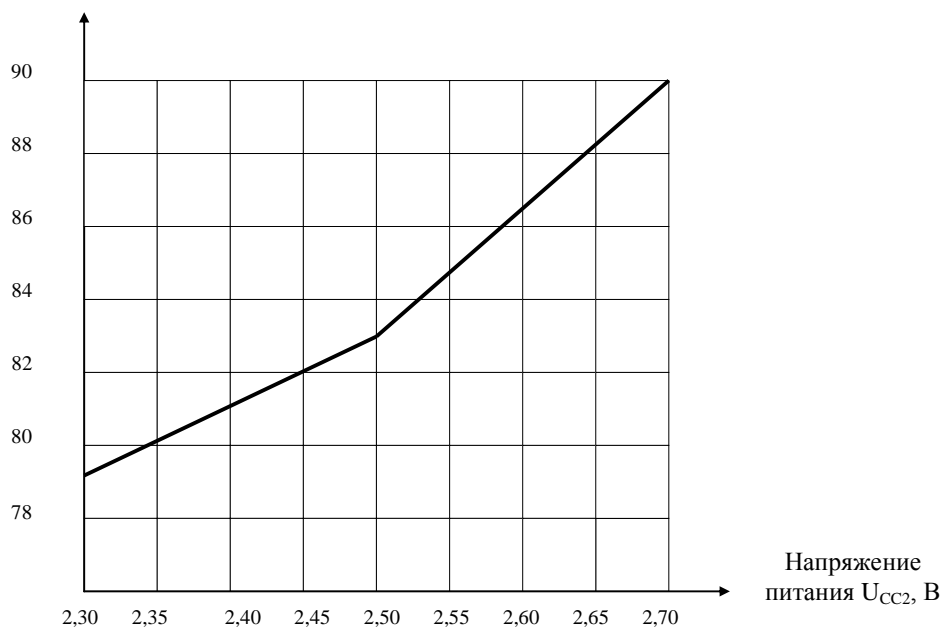
**Рисунок 14.4. Зависимость динамического тока потребления микросхемы по цепи CVDD от рабочей частоты при температуре окружающей среды от минус 60°C до +85°C,  $U_{CC2} = 2,63$  В и  $U_{CC1} = 3,13-3,47$  В**

Динамический ток потребления, мА



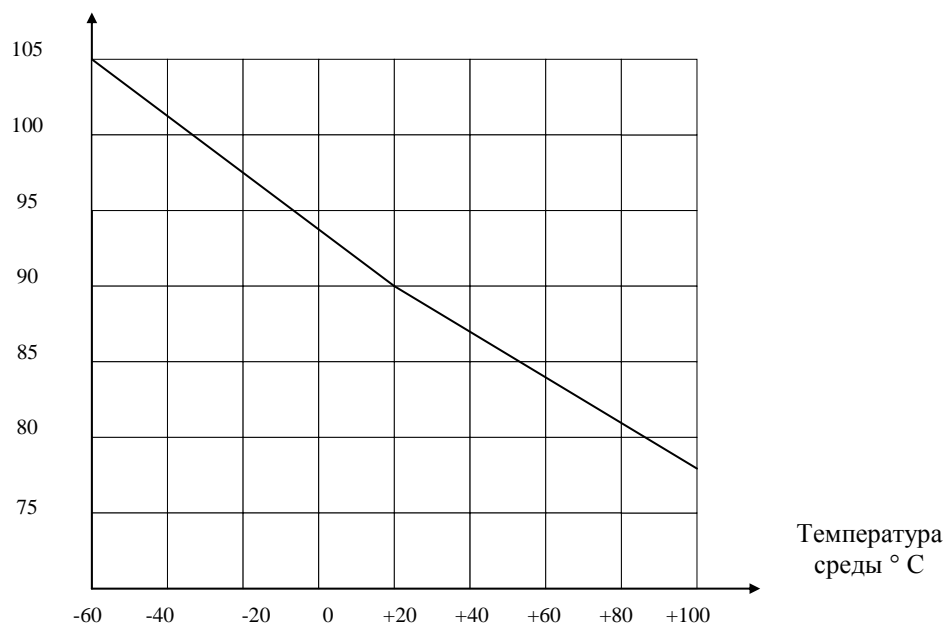
**Рисунок 14.5.** Зависимость динамического тока потребления микросхемы по цепи CVDD от напряжения питания при температуре окружающей среды от минус 60°C до +85°C, рабочей частоте 80 МГц и  $U_{CC1} = 3,13-3,47$  В.

Рабочая частота, МГц



**Рисунок 14.6.** Зависимость рабочей частоты микросхемы от напряжения питания  $U_{CC2}$  при температуре окружающей среды +85°C и  $U_{CC1} = 3,13-3,47$  В.

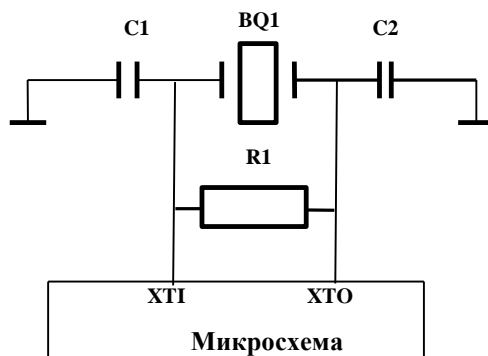
Рабочая частота, МГц



**Рисунок 14.7.** Зависимость рабочей частоты микросхемы от температуры окружающей среды при  $U_{CC1}=3,13$  В и  $U_{CC2}=2,37$  В.

## 14.7 Рекомендации по подключению кварцевого резонатора.

Схема подключения кварцевого резонатора к микросхеме приведена на Рисунок 14.8.



**Рисунок 14.8.** Схема подключения кварцевого резонатора к микросхеме

Частота кварцевого резонатора должна быть от 10 до 12 МГц. Ориентировочные величины:  $R1=1$  мОм,  $C1=C2=7$  пФ. Конкретная величина конденсаторов и резистора указывается в документации на резонатор.

## 15. ОПИСАНИЕ ВНЕШНИХ ВЫВОДОВ

Микросхема 1892ВМ3Т имеет следующие выводы:

- порт внешней памяти – 93;
- управление – 24;
- 2 последовательных порта – 12;
- 4 линковых порта - 40;
- UART – 10;
- электропитание – 52 (в корпусе QFP-240).

Описание выводов микросхемы 1892ВМ3Т приведено в табл. 15.1-15.7.

**Таблица 15.1. Порт внешней памяти**

Название вывода	Количество	Тип	Назначение
A[31:0]	32	О	Шина адреса.
D[31:0]	32	Ю	Шина данных
nWR[3:0]	4	О	Запись байтов асинхронной памяти
nWE	1	О	Запись асинхронной памяти
nRD	1	О	Чтение асинхронной памяти
nACK	1	И	Готовность асинхронной памяти
nCS[3:0]	4	О	Разрешение выборки блоков внешней памяти
SRAS	1	О	Строб адреса строки
SCAS	1	О	Строб адреса колонки
SWE	1	О	Разрешение записи
DQM[3:0]	4	О	Маска выборки байта при обмене данными с памятью типа SRAM, SDRAM. При подключении памяти типа SDRAM эти сигналы DQM подключаются к соответствующим выводам DQM микросхем памяти. При подключении памяти типа SRAM сигналы DQM подключаются к соответствующим выводам BLE, ВНЕ микросхем памяти (например, микросхемы типа CY7C1011CV33 фирмы CYPRESS).
SCLK	1	О	Тактовая частота работы
SKE	1	О	Разрешение частоты
A10	1	О	10 разряд адреса
BA[1:0]	2	О	Номер банка
nFLYBY	1	О	Признак режима передачи DMA “Flyby”
nOE	1	О	Разрешение чтения внешнего устройства в режиме “Flyby”
nCSIO[3:0]	4	О	Номер канала DMA MemCh, выполняющий передачу в режиме “Flyby”
Всего 93 вывода			

Таблица 15.2. Управление

Название вывода	Количество	Тип	Назначение
nDMAR[3:0]	4	I	Запрос канала DMA. Формируется по отрицательному фронту. Минимальная длительность – не менее 1,5 периодов системной тактовой частоты CLK (частота, на которой работает CPU).
NMI	1	I	Немаскируемое прерывание. Формируется по положительному фронту сигнала
nIRQ[3:0]	4	I	Запросы прерывания. Потенциальные сигналы, активный низкий уровень
BYTE	1	I	Разрядность блока внешней памяти, подключенного к выводу nCS[3] микросхемы: 0 – 32 разряда; 1 – 8 разрядов.
WDT	1	O	Признак срабатывания сторожевого таймера. Этот сигнал формируется, если в программе произошел сбой. Его можно подать на системный контроллер, который будет принимать решение, что делать в данной ситуации.
Ch_PLL	1	I	Технологический вход. Необходимо подключить к электропитанию PVDD
PLL_EN	1	I	Разрешение работы PLL: 0 – системная тактовая частота микроконтроллера равна входной частоте ХТИ (см. рис. 4.1); 1 - системная тактовая частота микроконтроллера поступает из PLL и равна входной частоте ХТИ, умноженной на коэффициент умножения/деления. (поле CLK_SEL регистра CSR).
PLL_OUT	1	O	Технологический выход. Должен быть незадействованным.

Название вывода	Количество	Тип	Назначение
XTI, XTO	2	I, O	Для тактирования микропроцессора можно использовать внешний кварцевый резонатор или внешний генератор импульсов. Внешний кварцевый резонатор должен подключаться к выводам XTI и XTO. Внешний генератор импульсов должен подключаться к выводу XTI, а вывод XTO должен быть незадействованным. Если используется встроенный умножитель частоты (PLL_EN = 1), то допускается: на вход XTI подавать частоту от 9 до 12 МГц. Длительность фронта и спада – не более 10 нс; к выводам XTI, XTO подключать кварцевый резонатор частотой от 10 до 12 МГц. Если не используется встроенный умножитель частоты (PLL_EN = 0), то допускается: на вход XTI подавать частоту от 1 до 80 МГц. Длительность фронта и спада – не более 10 нс; к выводам XTI, XTO подключать кварцевый резонатор частотой от 10 до 12 МГц.
RTC_XTI	1	I	Частота реального времени от 1 кГц до 10 МГц. Как правило - 32,768 кГц. Длительность фронта и спада – не более 10 нс
nRST	1	I	Сигнал установки исходного состояния
TCK	1	I	Тестовый тактовый сигнал (JTAG)
TRST	1	I	Установка исходного состояния (JTAG) При использовании микросхемы без возможности подключения эмулятора JTAG вывод TRST должен быть подключен к шине GND. Если микросхема используется с возможностью подключения эмулятора JTAG, то при включении электропитания микросхемы вывод TRST должен иметь низкий уровень и переключаться на высокий уровень через время не менее 1 мс после установки стабильного электропитания и стабильной тактовой частоты на входе XTI
TMS	1	I	Выбор режима теста (JTAG)
TDI	1	I	Вход данных теста (JTAG)
TDO	1	O	Выход данных теста (JTAG)

Название вывода	Количество	Тип	Назначение
nDE	1	IO	Состояние DEBUG. Сигнал предназначен для отладки программного обеспечения нескольких 1892BM3T (до 8), работающих одновременно. Для этого выводы nDE у этих микросхем необходимо объединить в проводное ИЛИ. Если совместная отладка не используется, то вывод nDE должен быть незадействованным.
Всего 24 вывода			

**Таблица 15.3. Последовательные порты (2 штуки)**

Название вывода	Количество	Тип	Назначение
DT	1	O	Передаваемые данные
DR	1	I	Принимаемые данные
TCLK	1	IO	Частота передачи
RCLK	1	IO	Частота приема
TFS	1	IO	Синхронизация передачи
RFS	1	IO	Синхронизация приема
Всего 6*2=12 выводов			

**Таблица 15.4. Линковые порты (4 штуки)**

Наименование Сигнала	Количество	Тип	Назначение
LDAT[7:0]	8	IO	Шина данных.
LCLK	1	IO	Синхронизация
LACK	1	IO	Подтверждение
Всего 10*4=40 выводов			

**Таблица 15.5. UART**

Наименование сигнала	Количество	Тип	Назначение
SIN	1	I	Вход последовательных данных
SOUT	1	O	Выход последовательных данных
nOUT1	1	O	Выход общего назначения
nOUT2	1	O	Выход общего назначения
nDCD	1	I	Признак обнаружения модемом несущей частоты (Receiver Line Signal Detect)
nRI	1	I	Признак обнаружения модемом телефонного звонка (Ring Indicator)
nDTR	1	O	Готовность UART к установлению связи (Data Terminal Ready)
nRTS	1	O	Готовность UART к обмену данными (Request To Send)
nCTS	1	I	Готовность модема к обмену данными (Clear To Send)
nDSR	1	I	Готовность модема к установлению связи (Data Set Ready)
Всего 10 выводов			



**Таблица 15.6. Электропитание**

Название вывода	Количество	Назначение
CVDD	14	Напряжение электропитания ядра ( $U_{CC2}$ )
PVDD	12	Напряжение электропитания входных и выходных драйверов ( $U_{CC1}$ )
CGND	14	Земля ядра
PGND	12	Земля входных и выходных драйверов
Всего 52 вывода		

Примечание. Цепи CGND и PGND можно объединять на плате.

Нумерация выводов микросхемы 1892ВМ3Т в корпусе QFP-240 приведена в таблице 15.7.

**Таблица 15.7. Нумерация выводов микросхемы 1892ВМ3Т в корпусе QFP-240**

Номер вывода	Тип вывода	Условное обозначение вывода	Номер вывода	Тип вывода	Условное обозначение вывода
1	-	-	61	-	-
2	I	nDCD	62		
3	I	nRI	63	IO	D[26]
4	IO	LDAT3[7]	64	IO	D[25]
5	IO	LDAT3[6]	65	IO	D[24]
6	IO	LDAT3[5]	66	IO	D[23]
7	IO	LDAT3[4]	67	IO	D[22]
8		CVDD	68		CVDD
9		CGND	69		CGND
10	IO	LDAT3[3]	70	IO	D[21]
11	IO	LDAT3[2]	71	IO	D[20]
12	IO	LDAT3[1]	72	IO	D[19]
13	IO	LDAT3[0]	73		PVDD
14	IO	LACK3	74		PGND
15	IO	LCLK3	75	IO	D[18]
16	O	A[31]	76	IO	D[17]
17	O	A[30]	77	IO	D[16]
18	O	A[29]	78	IO	D[15]
19	O	A[28]	79		CVDD
20	O	A[27]	80		CGND
21		PVDD	81	IO	D[14]
22		PGND	82	IO	D[13]
23	O	A[26]	83		PVDD
24	O	A[25]	84		PGND
25	O	A[24]	85	IO	D[12]
26	O	A[23]	86	IO	D[11]
27	O	A[22]	87	IO	D[10]
28	O	A[21]	88	IO	D[9]
29	O	A[20]	89		CVDD
30	O	A[19]	90		CGND
31	O	A[18]	91	IO	D[8]
32		CVDD	92	IO	D[7]
33		CGND	93	IO	D[6]
34	O	A[17]	94		PVDD
35	O	A[16]	95		PGND
36	O	A[15]	96	IO	D[5]
37	O	A[14]	97	IO	D[4]
38	O	A[13]	98		CVDD
39	O	A[12]	99		CGND
40	O	A[11]	100	IO	CVDD
41	O	A[10]	101	IO	CGND
42	O	A[9]	102	IO	D[3]
43		PVDD	103	IO	D[2]
44		PGND	104	IO	D[1]
45	O	A[8]	105	IO	D[0]
46	O	A[7]	106	O	nWR[3]
47	O	A[6]	107	O	nWR[2]
48	O	A[5]	108	O	nWR[1]
49	O	A[4]	109		nWR[0]
50	O	A[3]	110		nWE
51	O	A[2]	111		PVDD
52	O	A[1]	112		PGND
53	O	A[0]	113	O	nRD
54		PVDD	114	I	nACK
55		PGND	115	O	nCS[3]
56	IO	D[31]	116	O	nCS[2]
57	IO	D[30]	117	O	nCS[1]
58	IO	D[29]	118		
59	IO	D[28]	119		
60	IO	D[27]	120	-	-

## Продолжение Таблица 15.7

Номер вывода	Тип вывода	Условное обозначение вывода	Номер вывода	Тип вывода	Условное обозначение вывода
121	-	-	181		
122	O	nCS[0]	182	I	DR0
123	O	SRAS	183	IO	TCLK0
124	O	SCAS	184	IO	RCLK0
125	O	SWE	185	IO	TFS0
126	O	DQM[3]	186	IO	RFS0
127		CVDD	187		CVDD
128		CGND	188		CGND
129	O	DQM[2]	189	O	DT1
130	O	DQM[1]	190	I	DR1
131	O	DQM[0]	191	IO	TCLK1
132	O	SCLK	192	IO	RCLK1
133	O	CKE	193	IO	TFS1
134	O	A10	194	IO	RFS1
135	O	BA[1]	195	IO	LDAT0[7]
136	O	BA[0]	196	IO	LDAT0[6]
137	O	nFLYBY	197	IO	LDAT0[5]
138		PVDD	198		CVDD
139		PGND	199		CGND
140	O	nCSIO[3]	200	IO	LDAT0[4]
141	O	nCSIO[2]	201	IO	LDAT0[3]
142	O	nCSIO[1]	202	IO	LDAT0[2]
143	O	nCSIO[0]	203	IO	LDAT0[1]
144		CVDD	204	IO	LDAT0[0]
145		CGND	205	IO	LACK0
146	O	nOE	206	IO	LCLK0
147	I	nDMAR[3]	207	IO	LDAT1[7]
148	I	nDMAR[2]	208	IO	LDAT1[6]
149	I	nDMAR[1]	209		PVDD
150	I	nDMAR[0]	210		PGND
151	I	NMI	211	IO	LDAT1[5]
152		PVDD	212	IO	LDAT1[4]
153		PGND	213	IO	LDAT1[3]
154	I	nIRQ[3]	214	IO	LDAT1[2]
155	I	nIRQ[2]	215	IO	LDAT1[1]
156	I	nIRQ[1]	216	IO	LDAT1[0]
157		CVDD	217	IO	LCLK1
158		CGND	218	IO	LACK1
159	I	nIRQ[0]	219	IO	LDAT2[7]
160	I	BYTE	220	IO	LDAT2[6]
161	I	PLL_EN	221	IO	LDAT2[5]
162	I	Ch PLL	222		PVDD
163	I	XTI	223		PGND
164	O	XTO	224	IO	LDAT2[4]
165	I	RTC_XTI	225	IO	LDAT2[3]
166		-	226	IO	LDAT2[2]
167	O	PLL_OUT	227	IO	LDAT2[1]
168		PVDD	228	IO	LDAT2[0]
169		PGND	229	IO	LCLK2
170	I	nRST	230	IO	LACK2
171	I	TCK	231	I	SIN
172		CVDD	232		CVDD
173		CGND	233		CGND
174	IO	nDE	234	O	SOUT
175	I	TRST	235	O	nDTR
176	I	TMS	236	O	nRTS
177	I	TDI	237	I	nCTS
178	O	TDO	238	I	nDSR
179	O	WDT	239	O	nOUT2
180	O	DT0	240	O	nOUT1

Чертеж корпуса QFP-240 приведен на Рисунок 15.1

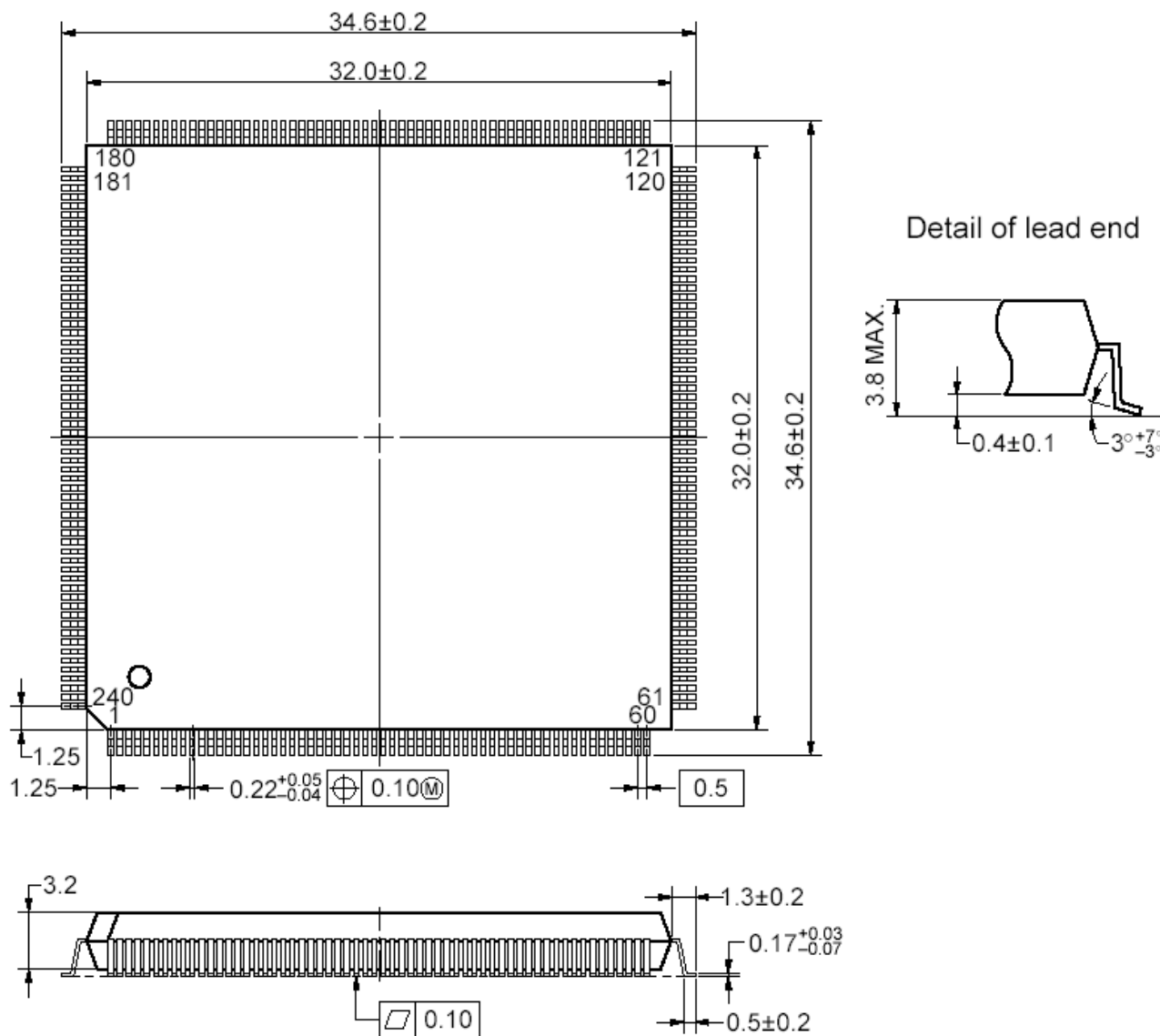


Рисунок 15.1. Чертеж корпуса QFP-240

## 16. ИСТОРИЯ ИЗМЕНЕНИЙ

### 16.1 30 июня 2006 г.

- Уточнен подраздел 2.5 (Устройство управления памятью (MMU)).
- Уточнены п. 2.6.2, 2.7.3.10.
- Уточнены табл. 8.1 (бит INT\_CTR), п. 8.2 и табл. 8.5.
- Раздел 12 (Линковый порт): скорректированы табл. 12.1, 12.4, п. 12.4.1. Приведена временная диаграмма работы линкового порта.
- Скорректирован раздел 14. Приведены временные параметры.
- Уточнен раздел 15 в части назначения выводов.

### 16.2 23 октября 2007 г.

- В разделе 14 уточнена допустимая частота внешнего кварцевого резонатора.
- В разделе 15 дополнено описание выводов ХТІ, ХТО, RTC\_ХТІ.

### 16.3 20 февраля 2008 г.

- Уточнен раздел 13.

### 16.4 02 июня 2009 г.

- Уточнен раздел 4 в части формирования частоты CLK.
- Уточнен раздел 11 в части описания битов TFSR, DITFS, TUVF, а также описания одноканального и многоканального режимов работы SPORT.
- Введено требование подключения входа Ch\_PLL к электропитанию PVDD.

### 16.5 13 декабря 2011 г.

- Удалены таблицы частот UART (№№ 10.11-10.13).

### 16.6 27 августа 2012 г.

- Скорректирован раздел 13.
- Скорректирован раздел 14.1 в части требований к порядку подачи питающих напряжений.
- В табл. 15.2 уточнены требования к параметрам частоты на входах ХТІ, RTC\_ХТІ.
- Из документа удалены приложения. Вместо приложения 1 (краткое описание системы команд DSP-ядра ELcore-x4) доступен документ «DSP-ядро ELcore\_x4. Система инструкций (РАЯЖ. 431280.003Д2)». Приложение 2 вынесено в

документ «Микросхемы 1892ВМ3Т, 1892ВМ2Я, 1892ВМ5Я, 1892ВМ8Я.  
Примеры программирования».

### **16.7 25 февраля 2013 г.**

- Скорректирован раздел 1.2 в части областей применения микросхемы.

### **16.8 15 апреля 2013 г.**

- Уточнено описание сигналов DQM в табл. 15.1.

### **16.9 22 ноября 2018 г.**

- Уточнено описание вывода TRST в табл. 15.2;
- Изменено оформление документа.