МИКРОСХЕМЫ ИНТЕГРАЛЬНЫЕ
1906ВМ016, 1906ВМ01А6

Руководство пользователя

2021

Содержание

1 Введение 6
2 Краткое техническое описание 6
2.1 Обзор функциональных возможностей 6
2.2 Особенности архитектуры 7
2.3 Функциональное назначение выводов 8
2.4 Система команд 14
2.5 Карта памяти микропроцессора 14
2.6 Механизмы сбоеустойчивости 16
2.7 Основные характеристики микросхем 18
3 Начало работы 20
3.1 Режимы работы процессора 20
3.2 Подключение выводов 20
3.3 Внешнее ОЗУ 21
3.4 Инициализация кэша, конфигурация доступа к PROM 21
3.5 Запуск программы из ПЗУ 22
3.6 Отладка программ 24
3.7 Сторожевой таймер 25
3.8 Требования по инициализации областей памяти при использовании помехоустойчивого кодирования 25
4 Функциональное описание 27
4.1 Система сброса 27
4.1.1 Общие сведения 27
4.1.2 Принцип работы 27
4.2 Система тактирования 27
4.3 Контроллер конфигурации 28
4.3.1 Общие сведения 28
4.3.2 Принцип работы 29
4.3.3 Формат представления регистров 30
4.3.4 Регистры 30
4.4 Процессорное ядро LEON4 38
4.4.1 Общие сведения 38
4.4.2 Блок целочисленной арифметики LEON4 40
4.4.3 Подсистема кэша 48
4.4.4 Блок управления памятью (MMU) 54
4.4.5 Блок арифметики с плавающей запятой 55
4.4.6 AMBA интерфейс 61
4.4.7 Идентификаторы адресных пространств (ASI) 64
4.4.8 Регистры конфигурации 67
4.5 Контроллер PROM / IO 76
4.5.1 Общие сведения 76
4.5.2 Доступ к области PROM 77
4.5.3 Доступ к области IO 79
4.5.4 Управление временной диаграммой контроллера PROM / IO 80
4.5.5 8-битный режим области PROM 80
4.5.6 Система сигнализации о готовности 81
4.5.7 Описание системы сигнализации об ошибке доступа 82
4.5.8 Коррекция ошибок 83
4.5.9 Регистры 84
4.6 Контроллер SRAM 85
4.6.1 Общие сведения 85
4.6.2 Доступ к области SRAM 85
4.6.3 Управление временной диаграммой контроллера SRAM 88
4.6.4 Система сигнализации о готовности 88
4.6.5 Описание системы сигнализации об ошибке доступа 89

4.6.6 Коррекция ошибок 90
4.6.7 Регистры 91
4.7 Контроллер SDRAM 92
4.7.1 Общие сведения 92
4.7.2 Доступ к области SDRAM 93
4.7.3 Тактирование синхронной динамической памяти 93
4.7.4 Блок управления трактом задержки SDCLK 95
4.7.5 Инициализация микросхем динамической памяти 97
4.7.6 Режимы выполнения обращений к памяти 98
4.7.7 Управление параметрами конфигурации SDRAM 100
4.7.8 Коррекция ошибок 102
4.7.9 Регистры 104
4.8 Контроллер прерываний 104
4.8.1 Общие сведения 104
4.8.2 Принцип работы 104
4.8.3 Таблица прерываний 106
4.8.4 Регистры 107
4.9 Отладочный модуль LEON4 109
4.9.1 Общие сведения 109
4.9.2 Принцип работы 110
4.9.3 Буфер трассировки AHB 111
4.9.4 Буфер трассировки инструкций 112
4.9.5 Карта памяти DSU 113
4.9.6 Регистры DSU 116
4.10 Отладочный интерфейс JTAG 124
4.10.1 Общие сведения 124
4.10.2 Принцип работы 125
4.10.3 Тактовая синхронизация 126
4.10.4 Контроллер TAP JTAG 127
4.10.5 Регистры TAP контроллера 128
4.11 Отладочный интерфейс UART 128
4.11.1 Общие сведения 128
4.11.2 Принцип работы 128
4.11.3 Регистры 129
4.12 Регистры статуса AHB 131
4.12.1 Общие сведения 131
4.12.2 Принцип работы 131
4.12.3 Регистры 131
4.13 Блок таймеров общего назначения 132
4.13.1 Общие сведения 132
4.13.2 Принцип работы 133
4.13.3 Регистры 134
4.14 Порт ввода-вывода общего назначения 137
4.14.1 Общие сведения 137
4.14.2 Принцип работы 138
4.14.3 Регистры 139
4.15 Интерфейс UART 141
4.15.1 Общие сведения 141
4.15.2 Принцип работы 142
4.15.3 Генерация скорости передачи 143
4.15.4 Режим обратной петли 144
4.15.5 Режим отладки FIFO 144
4.15.6 Генерация прерываний 144
4.15.7 Регистры 145

4.16 Интерфейс MIL-STD-1553/AS15531 148
4.16.1 Общие сведения 148
4.16.2 Электрический интерфейс 149
4.16.3 Принцип работы 149
4.16.4 Работа контроллера шины 150
4.16.5 Работа удаленного терминала 156
4.16.6 Работа монитора шины 162
4.16.7 Тактирование 163
4.16.8 Регистры 163
4.17 Интерфейс SpaceWire с поддержкой RMAP 176
4.17.1 Общие сведения 176
4.17.2 Принцип работы 176
4.17.3 Интерфейс связи 178
4.17.4 Распространение системного времени (тайм-кодов) 180
4.17.5 Приёмный канал DMA 181
4.17.6 Передающий канал DMA 187
4.17.7 RMAP 191
4.17.8 Интерфейс AMBA 196
4.17.9 Аппаратные особенности 197
4.17.10 Регистры 198
4.17.11 Интерфейс прикладного программирования (API) 206
4.18 Интерфейс Ethernet с поддержкой EDCL 220
4.18.1 Общие сведения 220
4.18.2 Принцип работы 221
4.18.3 Интерфейс Tx DMA 221
4.18.4 Интерфейс Rx DMA 223
4.18.5 Интерфейс MDIO 226
4.18.6 Линия связи отладки Ethernet (EDCL) 226
4.18.7 Независимый от среды интерфейс (MII) 228
4.18.8 Программные драйверы 229
4.18.9 Регистры 229
4.19 Интерфейс CAN 233
4.19.1 Общие сведения 233
4.19.2 Общие сведения о контроллере интерфейса CAN 234
4.19.3 Интерфейс AHB 234
4.19.4 Режим BasicCAN 235
4.19.5 Режим PeliCAN 238
4.19.6 Общие регистры 249
4.19.7 Особенности интерфейса CAN 250
4.20 Интерфейс USB 2.0 250
4.20.1 Общие сведения 250
4.20.2 Принцип работы 251
4.20.3 Операции DMA 252
4.20.4 Порядок следования байт 253
4.20.5 Поддержка приемо-передатчиков 253
4.20.6 Регистры 253
4.21 Интерфейс PCI 255
4.21.1 Общие сведения 255
4.21.2 Конфигурация 256
4.21.3 Принцип работы 261
4.21.4 Интерфейс ведущего устройства PCI 263
4.21.5 Интерфейс ведомого устройства PCI 265
4.21.6 Контроллер DMA 267
4.21.7 Прерывания 269

4.21.8 Сброс 269
4.21.9 Арбитраж шины PCI 269
4.21.10 Регистры 271
4.22 Контроллер шины AHB 276
4.22.1 Общие сведения 276
4.22.2 Принцип работы 276
4.23 Контроллер шины APB 278
4.23.1 Общие сведения 278
4.23.2 Принцип работы 278
4.24 Цепь граничного сканирования 279
4.25 Конфигурация процессоров 1906ВМ016, 1906ВМ01А6 279
5 Описание архитектуры 281
5.1 Архитектура SPARC 281
5.1.1 Особенности архитектуры 281
5.1.2 Технические характеристики 281
5.1.3 Словарь терминов 282
5.2 Форматы данных 283
5.3 Регистры 286
5.3.1 Регистры IU 287
5.3.2 Регистры управления/статуса блока целочисленной арифметики (IU) 290
5.3.3 Регистры FPU 294
5.3.4 Регистры управления/статуса FPU 295
5.4 Системные прерывания 301
5.4.1 Категории прерываний 302
5.4.2 Модели системных прерываний 303
5.4.3 Идентификация системных прерываний 307
5.4.4 Описание системного прерывания 309
5.4.5 Описание отдельных исключений / прерываний 310
6 Введение в систему команд 313
6.1 Выполнение инструкций 313
6.1.1 Форматы инструкций 313
6.1.2 Поля инструкций 314
6.2 Категории инструкций 316
6.2.1 Инструкции загрузки/сохранения 316
6.2.2 Целочисленные арифметические инструкции 318
6.2.3 Инструкции передачи управления (CTI) 319
6.2.4 Чтение/запись регистров состояния 324
6.2.5 Инструкции операций с плавающей запятой (FPop) 325
6.2.6 Инструкции сопроцессора (CPop) 325
7 Программно-аппаратный комплекс средств разработки и отладки 326
7.1 Интегрированная среда разработки и отладки приложений 326
7.1.1 Состав ИСР 326
7.1.2 Интерфейс пользователя 326
7.1.3 Функциональные блоки 329
7.1.4 Подсистема структурного анализа исходных текстов 331
7.1.5 Подсистема отладки 331
7.2 Программное обеспечение «GRAIP» 331
7.3 Аппаратная часть 331
8 Заключение 334
Приложение А (обязательное) Описание инструкций 335
Приложение Б (обязательное) Конфигурация plug & play 389
Приложение В (справочное) Версии РП 393
Лист регистрации изменений 396

1 Введение

В настоящем руководстве пользователя (далее РП) приведено описание архитектуры, функциональных особенностей, системы команд и особенностей применения ИС 1906ВМ016, 1906ВМ01А6. В дальнейшем по тексту, если речь идет о микропроцессоре, процессоре или микросхеме (в единственном числе) без указания типономинала, данное описание распространяется на ИС 1906ВМ016 и на ИС 1906ВМ01А6.

ИС 1906ВМ016 и 1906ВМ01А6 построены на базе ядра LEON4 архитектуры SPARC V8/V8e и производятся по технологическим нормам 0,18 мкм.

К архитектурным особенностям ИС 1906ВМ016, 1906ВМ01А6 относятся интегрированные механизмы сбоеустойчивости.

2 Краткое техническое описание

2.1 Обзор функциональных возможностей

Микропроцессор имеет следующие особенности:

- процессорное ядро SPARC V8/LEON4 (с частичной поддержкой расширения V8e) с 2×4 Кбайт кэша инструкций, 2×4 Кбайт кэша данных, модулем MMU и встроенными механизмами сбоеустойчивости;
- встроенный отладочный модуль (DSU) с буферами трассировки;
- отладочный интерфейс JTAG;
- контроллеры памяти PROM/SRAM/SDRAM с кодами коррекции Хэмминга и Рида-Соломона (в контроллере SDRAM) и поддержкой:
  - до 256 Мбайт PROM;
  - до 256 Мбайт SRAM;
  - до 1 Гбайт SDRAM;
- 64 Кбайта встроенной памяти (OCRAM) с кодом коррекции Хэмминга;
- контроллер памяти ввода-вывода (IO) с поддержкой до 256 Мбайт;
- 4 интерфейса SpaceWire с поддержкой RMAP;
- 2 интерфейса MIL-STD-1553B;
- 2 интерфейса UART;
- 2 интерфейса CAN;
- интерфейс Ethernet с поддержкой EDCL;
- интерфейс USB 2.0;
- интерфейс PCI 2.2;
- 4 таймера-счетчика;
- сторожевой таймер;
- 16 выводов порта общего назначения;
- цепь граничного сканирования.

Поддерживаемые функции из расширения SPARC V8e:

- аппаратная поддержка операций умножения с накоплением;
- возможность одновекторной обработки системных прерываний;
- частичная запись в регистр %psr (инструкция WRPSR).

2.2 Особенности архитектуры

Структурная схема микропроцессора приведена на рисунке 2.2.1.

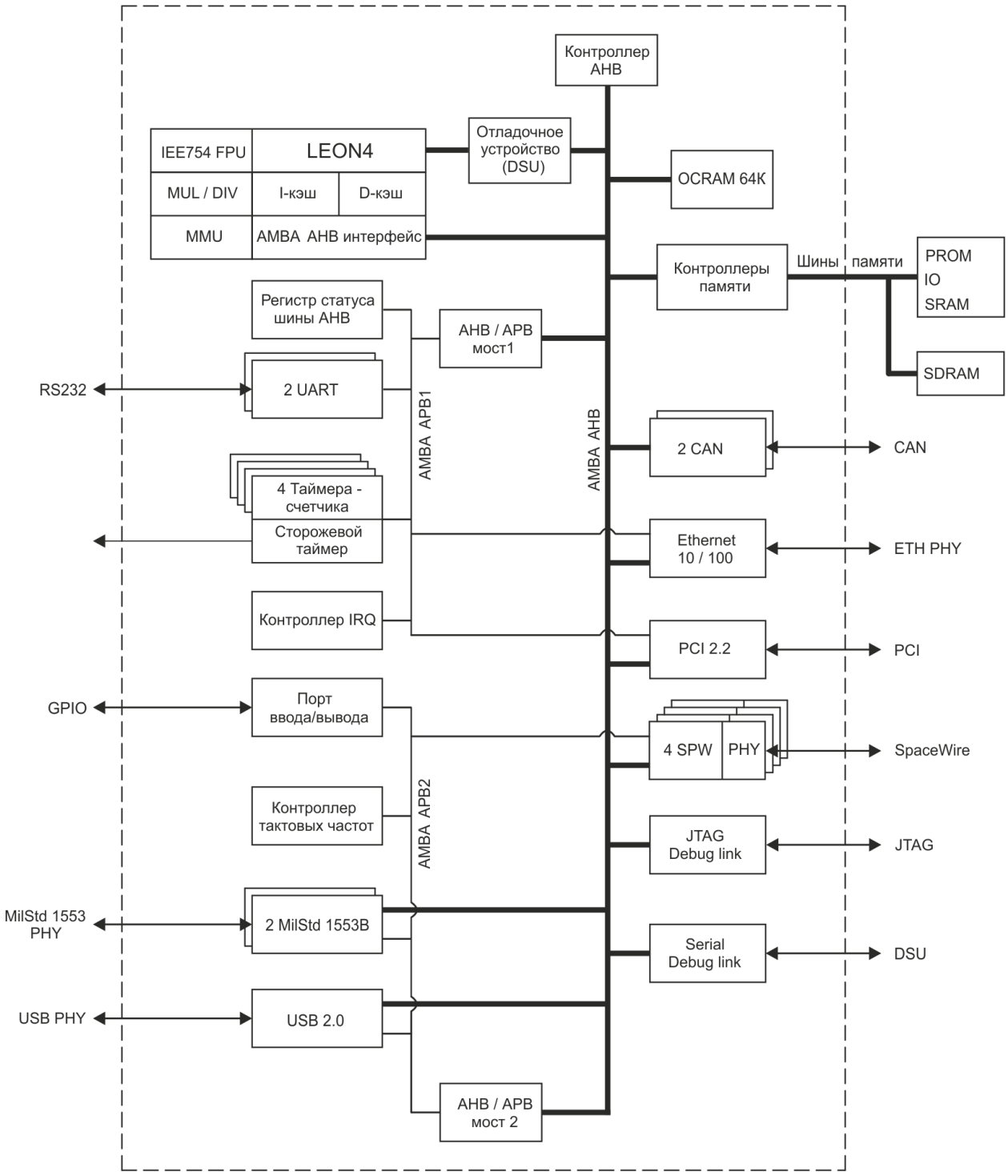


Рисунок 2.2.1 – Структурная схема микропроцессора

На рисунке 2.2.2 приведено условное графическое обозначение микропроцессора.

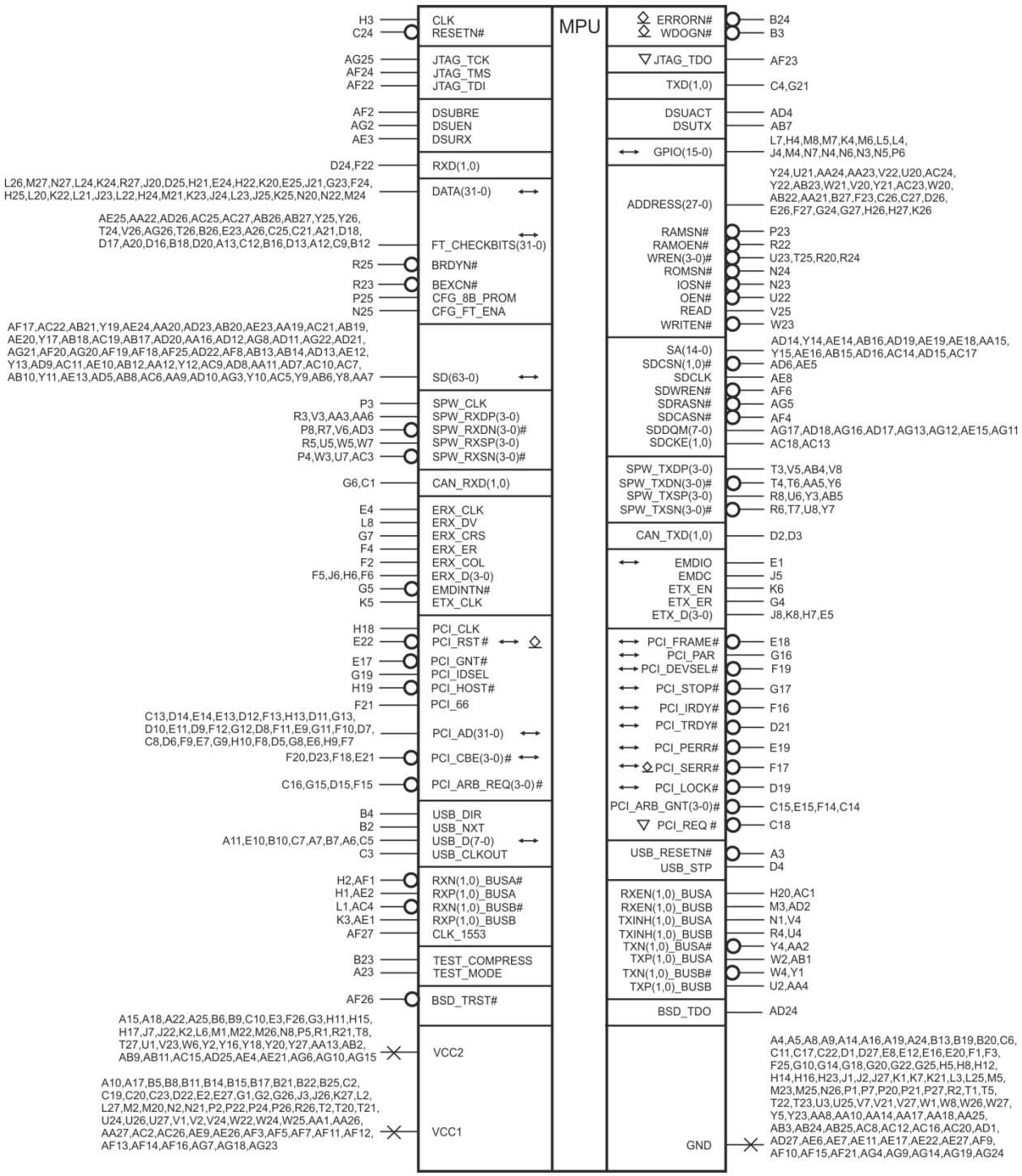


Рисунок 2.2.2 – Условное графическое обозначение микропроцессора

2.3 Функциональное назначение выводов

В таблице 2.3.1 представлен полный перечень выводов микропроцессора и их описание. На рисунке 2.3.1 приведена информация о 602-выводном металло-керамическом корпусе микропроцессора МК 6117.602-D.

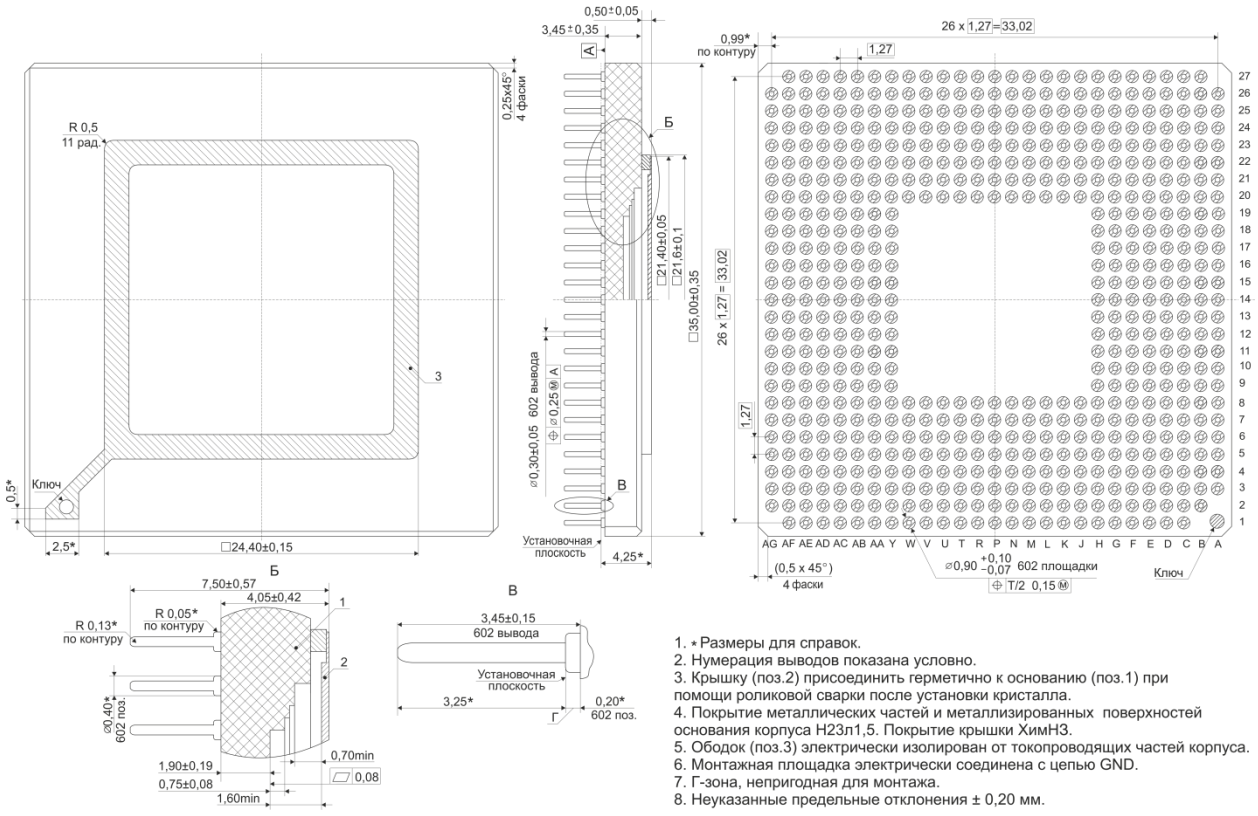


Рисунок 2.3.1 – Корпус микропроцессора

Таблица 2.3.1 – Функциональное назначение выводов

Название вывода	Обозначение вывода	Тип вывода	Функциональное назначение вывода
1	2	3	4
H3	CLK	I	Вход тактового сигнала
C24	RESETN#	I	Сброс
B24	ERRORN#	O/2	Флаг ошибки
B3	WDOGN#	O/2	Выход сторожевого таймера
AG25	JTAG_TCK	I	Вход TCK интерфейса JTAG
AF24	JTAG_TMS	I	Вход TMS интерфейса JTAG
AF22	JTAG_TDI	I	Вход TDI интерфейса JTAG
AF23	JTAG_TDO	O/Z	Выход TDO интерфейса JTAG
AF26	BSD_TRST#	I	Асинхронный сброс контроллера граничного сканирования
AD24	BSD_TDO	O	Выход данных контроллера граничного сканирования
A23 B23	TEST_MODE TEST_COMPRESS	I	Тестовые входы
N25	CFG_FT_ENA	I	Глобальное запрещение коррекции ошибок
P25	CFG_8B_PROM	I	Вход определения значения по сбросу бита активации 8-битного режима PROM
AD4	DSUACT	O	Выход ACT интерфейса DSU
AF2	DSUBRE	I	Вход BRE интерфейса DSU
AG2	DSUEN	I	Вход разрешения DSU
AE3	DSURX	I	Вход приемника DSU
AB7	DSUTX	O	Выход передатчика DSU

Продолжение таблицы 2.3.1

1	2	3	4
L7, H4, M8, M7, K4, M6, L5, L4, J4, M4, N7, N4, N6, N3, N5, P6	GPIO(15–0)	I/O	Выводы порта общего назначения, разряды 15–0
L26, M27, N27, L24, K24, R27, J20,D25, H21, E24, H22, K20, E25, J21, G23, F24, H25, L20, K22, L21, J23, L22, H24,M21, K23, J24, L23, J25, K25, N20, N22, M24	DATA(31–0)	I/O	Выводы шины данных, разряды 31–0
AE25, AA22, AD26, AC25, AC27, AB26, AB27, Y25, Y26, T24, V26, AG26, T26, B26, E23, A26, C25, C21, A21, D18, D17, A20, D16, B18, D20, A13, C12, B16, D13, A12, C9, B12	FT_CHECKBITS(31–0)	I/O	Выводы шины коррекции ошибок, разряды 31–0
R25	BRDYN#	I	Сигнал готовности SRAM, PROM, IO
R23	BEXCN#	I	Ошибка доступа SRAM, PROM, IO
Y24, U21, AA24, AA23, V22, U20, AC24, Y22, AB23, W21, V20, Y21, AC23, W20, AB22, AA21, B27, F23, C26, C27, D26, E26, F27, G24, G27, H26, H27, K26	ADDRESS(27–0)	O	Выводы шины адреса, разряды 27–0
P23	RAMSN#	O	Выбор блока SRAM
R22	RAMOEN#	O	Разрешение выдачи данных SRAM
U23, T25, R20, R24	WREN(3–0)#	O	Разрешение записи в SRAM, PROM разряды 3–0
N24	ROMSN#	O	Выбор блока PROM
N23	IOSN#	O	Выбор области IO
U22	OEN#	O	Разрешение выдачи данных PROM, IO
V25	READ	O	Чтение SRAM, PROM, IO
W23	WRITEN#	O	Запись в PROM, IO, SRAM
AF17, AC22, AB21, Y19, AE24, AA20, AD23, AB20, AE23, AA19, AC21, AB19, AE20, Y17, AB18, AC19, AB17, AD20, AA16, AD12, AG8, AD11, AG22, AD21, AG21, AF20, AG20, AF19, AF18, AF25, AD22, AF8, AB13, AB14, AD13, AE12, Y13, AD9, AC11, AE10, AB12, AA12, Y12, AC9, AD8, AA11, AD7, AC10, AC7, AB10, Y11, AE13, AD5, AB8, AC6, AA9, AD10, AG3, Y10, AC5, Y9, AB6, Y8, AA7	SD(63–0)	I/O	Выводы шины данных SDRAM, разряды 63–0

Продолжение таблицы 2.3.1

1	2	3	4
AD14, Y14, AE14, AB16, AD19, AE19, AE18, AA15, Y15, AE16, AB15, AD16, AC14, AD15, AC17	SA(14–0)	O	Выводы шины адреса SDRAM, разряды 14–0
AD6, AE5	SDCSN(1,0)#	O	Выбор блока SDRAM, разряды 1,0
AE8	SDCLK	O	Тактовый сигнал SDRAM (особенности использования описаны в подразделе «3.3»)
AF6	SDWREN#	O	Разрешение записи SDRAM
AG5	SDRASN#	O	Выход строба адреса ряда SDRAM
AF4	SDCASN#	O	Выход строба адреса колонки SDRAM
AG17, AD18, AG16, AD17, AG13, AG12, AE15, AG11	SDDQM(7–0)	O	Маска данных SDRAM, разряды 7–0
AC18, AC13	SDCKE(1,0)	O	Разрешение тактирования SDRAM, разряды 1,0
D24, F22	RXD(1, 0)	I	Входы приемника UART 1,0
C4, G21	TXD(1, 0)	O	Выходы передатчика UART 1,0
P3	SPW_CLK	I	Тактовый сигнал интерфейса SpaceWire
R3, V3, AA3, AA6	SPW_RXDP(3–0)	I	Прямые входы «D» приемника интерфейса SpaceWire, блоки 3–0
P8, R7, V6, AD3	SPW_RXDN(3–0)#	I	Инверсные входы «D» приемника интерфейса SpaceWire, блоки 3–0
R5, U5, W5, W7	SPW_RXSP(3–0)	I	Прямые входы «S» приемника интерфейса SpaceWire, блоки 3–0
P4, W3, U7, AC3	SPW_RXSN(3–0)#	I	Инверсные входы «S» приемника интерфейса SpaceWire, блоки 3–0
T3, V5, AB4, V8	SPW_TXDP(3–0)	O	Прямые выходы «D» передатчика интерфейса SpaceWire, блоки 3–0
T4, T6, AA5, Y6	SPW_TXDN(3–0)#	O	Инверсные выходы «D» передатчика интерфейса SpaceWire, блоки 3–0
R8, U6, Y3, AB5	SPW_TXSP(3–0)	O	Прямые выходы «S» передатчика интерфейса SpaceWire, блоки 3–0
R6, T7, U8, Y7	SPW_TXSN(3–0)#	O	Инверсные выходы «S» передатчика интерфейса SpaceWire, блоки 3–0
G6, C1	CAN_RXD(1,0)	I	Выводы приемника интерфейса CAN, разряды RXD 1,0
D2, D3	CAN_TXD(1,0)	O	Выводы передатчика интерфейса CAN, разряды TXD 1,0

Продолжение таблицы 2.3.1

1	2	3	4
C13, D14, E14, E13, D12, F13, H13, D11, G13, D10, E11, D9, F12, G12, D8, F11, E9, G11, F10, D7, C8, D6, F9, E7, G9, H10, F8, D5, G8, E6, H9, F7	PCI_AD(31–0)	I/O	Шина адреса интерфейса PCI, разряды 31–0
F20, D23, F18, E21	PCI_CBE(3–0) #	I/O	Сигнал CBE интерфейса PCI, разряды 3–0
C16, G15, D15, F15	PCI_ARB_REQ(3–0)#	I	Сигнал ARB _REQ интерфейса PCI, разряды 3–0
C18	PCI_REQ#	O/Z	Сигнал запроса интерфейса PCI
E18	PCI_FRAME#	I/O	Сигнал кадра интерфейса PCI
G16	PCI_PAR	I/O	Сигнал чётности интерфейса PCI
F19	PCI_DEVSEL#	I/O	Выбор устройства PCI
G17	PCI_STOP#	I/O	Сигнал остановки интерфейса PCI
F16	PCI_IRDY#	I/O	Сигнал IRDY интерфейса PCI
D21	PCI_TRDY#	I/O	Сигнал TRDY интерфейса PCI
E19	PCI_PERR#	I/O	Сигнал ошибки PERR интерфейса PCI
F17	PCI_SERR#	I/O/2	Сигнал ошибки SERR интерфейса PCI
D19	PCI_LOCK#	I/O	Сигнал блокировки интерфейса PCI
C15, E15, F14, C14	PCI_ARB_GNT(3–0)#	O	Сигнал ARB_GNT интерфейса PCI, разряды 3–0
H18	PCI_CLK	I	Тактовый сигнал интерфейса PCI
E22	PCI_RST#	I/O/2	Сигнал сброса интерфейса PCI
E17	PCI_GNT#	I	Сигнал GNT интерфейса PCI
G19	PCI_IDSEL	I	Сигнал IDSEL интерфейса PCI
H19	PCI_HOST#	I	Вход HOST интерфейса PCI
F21	PCI_66	I	Совместимость с режимом 66 МГц
B4	USB_DIR	I	Сигнал DIR интерфейса USB
B2	USB_NXT	I	NXT-вход интерфейса USB
A11, E10, B10, C7, A7, B7, A6, C5	USB_D(7–0)	I/O	Шина данных интерфейса USB, разряды 7–0
A3	USB_RESETN#	O	Сигнал сброса интерфейса USB
D4	USB_STP	O	STP-выход интерфейса USB
C3	USB_CLKOUT	I	Вход тактового сигнала интерфейса USB

Продолжение таблицы 2.3.1

1	2	3	4
E4	ERX_CLK	I	Тактовый вход приемника интерфейса Ethernet
L8	ERX_DV	I	Вход действительности приема данных интерфейса Ethernet
G7	ERX_CRS	I	Вход CRS приемника интерфейса Ethernet
F4	ERX_ER	I	Вход ER ошибки приёма интерфейса Ethernet
F2	ERX_COL	I	Вход обнаружения коллизии интерфейса Ethernet
F5, J6, H6, F6	ERX_D(3–0)	I	Входы приема данных интерфейса Ethernet, разряды RXD 3–0
G5	EMDINTN#	I	Вход прерывания интерфейса Ethernet
K5	ETX_CLK	I	Тактовый вход передатчика интерфейса Ethernet
E1	EMDIO	I/O	Вход/выход управления данными
J5	EMDC	O	Тактовый сигнал управления данными интерфейса Ethernet
K6	ETX_EN	O	Выход разрешения передатчика интерфейса Ethernet
G4	ETX_ER	O	Выход ошибки передачи интерфейса Ethernet
J8, K8, H7, E5	ETX_D(3–0)	O	Выходы передачи данных интерфейса Ethernet, разряды TXD 3–0
AF27	CLK_1553	I	Вход тактового сигнала интерфейсов Mil-STD-1553
H2, AF1	RXN(1,0)_BUSA#	I	Инверсные входы (1, 0) приёмников шины А интерфейсов Mil-STD-1553
H1, AE2	RXP(1,0)_BUSA	I	Прямые входы (1, 0) приёмников шины А интерфейсов Mil-STD-1553
L1, AC4	RXN(1,0)_BUSB#	I	Инверсные входы (1, 0) приёмников шины В интерфейсов Mil-STD-1553
K3, AE1	RXP(1,0)_BUSB	I	Прямые входы (1, 0) приёмников шины В интерфейсов Mil-STD-1553
H20, AC1	RXEN(1,0)_BUSA	O	Разрешение работы приёмников шины А интерфейсов Mil-STD-1553
M3, AD2	RXEN(1,0)_BUSB	O	Разрешение работы приёмников шины В интерфейсов Mil-STD-1553
N1, V4	TXINH(1,0)_BUSA	O	Запрещение работы передатчиков шины А интерфейсов Mil-STD-1553
R4, U4	TXINH(1,0)_BUSB	O	Запрещение работы передатчиков шины В интерфейсов Mil-STD-1553
Y4, AA2	TXN(1,0)_BUSA#	O	Инверсные выходы (1,0) передатчиков шины А интерфейсов Mil-STD-1553
W2, AB1	TXP(1,0)_BUSA	O	Прямые выходы (1,0) передатчиков шины А интерфейсов Mil-STD-1553
W4, Y1	TXN(1,0)_BUSB#	O	Инверсные выходы (1,0) передатчиков шины В интерфейсов Mil-STD-1553
U2, AA4	TXP(1,0)_BUSB	O	Прямые выходы (1,0) передатчиков шины В интерфейсов Mil-STD-1553

Окончание таблицы 2.3.1

1	2	3	4
A4, A5, A8, A9, A14, A16, A19, A24, B13, B19, B20, C6, C11, C17, C22, D1, D27, E8, E12, E16, E20, F1, F3, F25, G10, G14, G18, G20, G22, G25, H5, H8, H12, H14, H16, H23, J1, J2, J27, K1, K7, K21, L3, L25, M5, M23, M25, N26, P1, P7, P20, P21, P27, R2, T1, T5, T22, T23, U3, U25, V7, V21, V27, W1, W8, W26, W27, Y5, Y23, AA8, AA10, AA14, AA17, AA18, AA25, AB3, AB24, AB25, AC8, AC12, AC16, AC20, AD1, AD27, AE6, AE7, AE11, AE17, AE22, AE27, AF9, AF10, AF15, AF21, AG4, AG9, AG14, AG19, AG24	GND	–	Общие выводы
A10, A17, B5, B8, B11, B14, B15, B17, B21, B22, B25, C2, C19, C20, C23, D22, E2, E27, G1, G2, G26, J3, J26, K27, L2, L27, M2, M20, N2, N21, P2, P22, P24, P26, R26, T2, T20, T21, U24, U26, U27, V1, V2, V24, W22, W24, W25, AA1, AA26, AA27, AC2, AC26, AE9, AE26, AF3, AF5, AF7, AF11, AF12, AF13, AF14, AF16, AG7, AG18, AG23	VCC1	–	Выводы питания ядра
J7, A15, A18, L6, N8, A22, A25, P5, B6, T8, B9, C10, W6, E3, F26, Y16, G3, AA13, AB9, H11, AB11, H15, AC15, H17, J22, K2, M1, M22, M26, R1, R21, T27, U1, V23, Y2, Y18, Y20, Y27, AB2, AD25, AE4, AE21, AG6, AG10, AG15	VCC2	–	Выводы питания буферов ввода-вывода
Примечания 1 Выводы с символом «#» имеют низкий активный уровень сигнала. 2 В графе «Тип вывода» обозначены: I – вход, I/O – вход/выход, O – выход, O/Z – выход с тремя состояниями, O/2 – выход с открытым стоком. 3 Выводы A23, AF26, N25, AE3, F22, D24 имеют схему «Pull-up».			

2.4 Система команд

Описание системы команд микропроцессора приведено в Приложении А.

2.5 Карта памяти микропроцессора

Вся память, доступная микропроцессору, разделяется на внутреннюю и внешнюю.

Внешняя память размещается в адресном пространстве 0x00000000 – 0x7FFFFFFF. Адреса (0x00000000 – 0x0FFFFFFF) зарезервированы под область PROM, адреса (0x20000000 – 0x2FFFFFFF) зарезервированы под область IO, адреса (0x40000000 – 0x7FFFFFFF) зарезервированы для SDRAM/SRAM. Одновременная работа контроллеров SRAM и SDRAM не поддерживается.

Внутренняя память размещается в адресном пространстве 0x80000000 – 0xFFFFFFFF. В него отображаются области памяти высокоскоростных интерфейсов процессора, регистры управления и обмена данными низкоскоростных интерфейсов (через модули APBCTRL 1 и APBCTRL 2), регистры и области доступа отладочного модуля, внутренняя память (0xA0000000 – 0xA000FFFF).

Карта памяти микропроцессора приведена на рисунке 2.5.1.

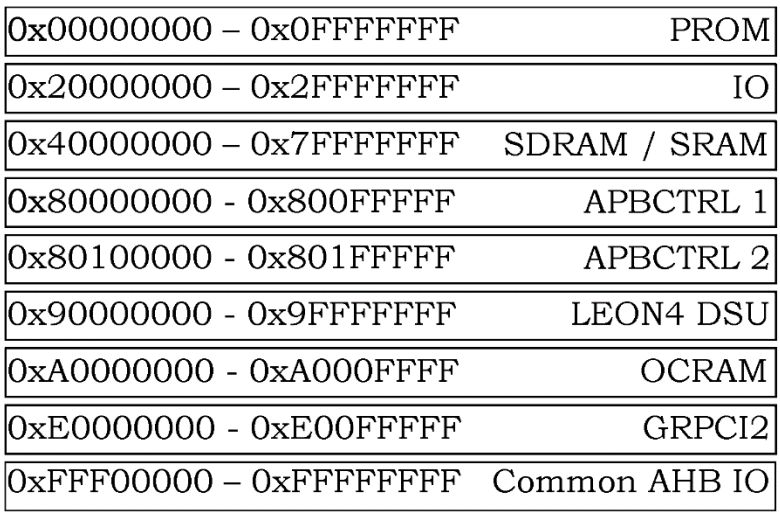


Рисунок 2.5.1 – Карта памяти микропроцессора

Подробная информация по размещению областей и модулей на шине AHB сведена в таблице 2.5.1. В графическом виде взаимное подключение модулей приведено на рисунке 2.2.1.

Таблица 2.5.1 – Адресное пространство внутренней памяти микропроцессора

Диапазон адресов	Размер	Описание	Имя модуля
1	2	3	4
0x00000000 – 0x0FFFFFFF	256 Мбайт	Контроллер PROM/IO. Область PROM	SW_ASRAM 1 (см. 4.5)
0x20000000 – 0x2FFFFFFF	256 Мбайт	Контроллер PROM/IO. Область IO	SW_ASRAM 1 (см. 4.5)
0x40000000 – 0x7FFFFFFF	1 Гбайт / 256 Мбайт	Контроллер SDRAM / Контроллер SRAM. Область RAM (SRAM / SDRAM)	SW_SDRAM (см. 4.7) / SW_ASRAM 2 (см. 4.6)
0x80000000	1 Мбайт	Мост AHB/APB	APBCTRL 1 (см. 4.23)
+0x000	256 байт	Регистр статуса AHB	AHBSTAT (см. 4.12)
+0x100	256 байт	Интерфейс UART 1	APBUART 1 (см. 4.15)
+0x200	256 байт	Контроллер прерываний	IRQMP (см. 4.8)
+0x300	256 байт	Модульный блок таймеров (Watchdog Timer)	GPTIMER 1 (см. 4.13)
+0x400	256 байт	Мост GRPCI2 PCI/AHB	GRPCI2 (см. 4.21)
+0x500	256 байт	Ethernet MAC	GRETH (см. 4.18)
+0x600	256 байт	Интерфейс UART 2	APBUART 2 (см. 4.15)
+0x700	256 байт	Отладочный интерфейс UART	AHBUART (см. 4.11)
+0x800	256 байт	Арбитр PCI	PCI_ARB (см. 4.21.9)
+0x900	256 байт	Модульный блок таймеров	GPTIMER 2 (см. 4.13)
+ 0xFF000	4 Кбайт	APBCTRL 1 plug & play конфигурация	APBCTRL 1
0x80100000	1 Мбайт	Мост AHB/APB	APBCTRL 2 (см. 4.23)
+0x000	256 байт	Контроллер конфигураций	SW_MCFG (см. 4.3)
+0x100	256 байт	USB расширенный Host-контроллер	GRUSBHC (см. 4.20)
+0x200	256 байт	MIL-STD-1553B интерфейс 1	GR1553B 1 (см. 4.16)
+0x300	256 байт	MIL-STD-1553B интерфейс 2	GR1553B 2 (см. 4.16)
+0x400	256 байт	Порт ввода-вывода общего назначения	GRGPIO (см. 4.14)
+0x500	256 байт	Интерфейс SpaceWire 1	GRSPW2 1 (см. 4.17)

Окончание таблицы 2.5.1

1	2	3	4
+0x600	256 байт	Интерфейс SpaceWire 2	GRSPW2 2 (см. 4.17)
+0x700	256 байт	Интерфейс SpaceWire 3	GRSPW2 3 (см. 4.17)
+0x800	256 байт	Интерфейс SpaceWire 4	GRSPW2 4 (см. 4.17)
+0xFF000	4 Кбайт	APBCTRL 2 plug & play конфигурация	APBCTRL 2
0x90000000	8 Мбайт	Отладочный модуль	DSU4 (см. 4.9)
0xA0000000	64 Кбайт	Внутренняя память данных	OCRAM
0xE0000000	1 Мбайт	Мост GRPCI2 PCI/AHB (PCI память)	GRPCI2 (см. 4.21)
0xFFF80000	128 Кбайт	Мост GRPCI2 PCI/AHB (PCI IO и PCI конфигурация)	GRPCI2 (см. 4.21)
+0x00000	64 Кбайт	PCI IO	
+0x10000	64 Кбайт	PCI конфигурация	
0xFFFA0000	256 байт	USB универсальный Host-контроллер	GRUSBHC (см. 4.20)
0xFFFC0000	4 Кбайт	CAN AHB интерфейс	CANMC (см. 4.19)
+0x000	32 байт	CAN 1 AHB интерфейс	
+0x100	32 байт	CAN 2 AHB интерфейс	
0xFFFFF000 – 0xFFFFFFFF	4 Кбайт	AHBCTRL plug & play конфигурация	AHBCTRL (см. 4.22)

В качестве примера использования карты памяти рассмотрим расчет адреса доступа к регистру направления порта ввода-вывода. К базовому адресу порта ввода-вывода общего назначения (0x80100000 + 0x400), определенного в таблице выше, следует прибавить соответствующее смещение из таблицы 4.14.1 из пункта «4.14.3 Регистры». Итоговый адрес – 0x80100408.

2.6 Механизмы сбоеустойчивости

Все функциональные блоки (ядро, периферийные устройства, регистровый файл), имеющие в своем составе элементы хранения (защелки и триггеры), обеспечены схемами тройного модульного резервирования (TMR). Схема TMR каждого элемента хранения представляет собой три триггера и элемент мажорирования, определяющего мажорированное значение из состояний, хранящихся в трех триггерах. В случае сбоя одного триггера на выходе элемента мажорирования будет корректное значение. Триггер, в котором произошел сбой, будет хранить некорректное значение. Если сбою подвергнется второй триггер тройки, на выходе элемента мажорирования будет некорректное значение.

Важно понимать, что штатное изменение состояния тройки триггеров элемента хранения происходит только при записи в элемент хранения нового значения.

Буферы приемо-передатчиков используют код Хэмминга для защиты данных (см. таблицу 2.6.1). В правой колонке таблицы приводятся записи в формате «Код Хэмминга (n,k)», где n – общее количество символов кодового слова, k – количество разрядов данных в кодовом слове. Поскольку буферы реализованы по типу FIFO, то данные в них записываются с дополнительным кодом Хэмминга. При извлечении данных из буфера производится проверка на соответствие данных проверочным битам. Автоматически исправляется одна ошибка на кодовое слово без информирования пользователя о данном факте. Если было искажено 2 бита, действий над данными не производится (недостаточно информации для исправления). Если ошибок больше, то с некоторой вероятностью вместо неисправимой ошибки может быть детектирована исправимая ошибка или кодовое слово будет признано корректным. Режим использования помехоустойчивого кодирования для всех буферов всегда активирован.

Таблица 2.6.1 – Защита внутренней памяти

Кэш память процессора	Код Хэмминга (39,32)
FIFO в блоках SpaceWire	Код Хэмминга (39,32)
Буфер RMAP в блоках SpaceWire	Код Хэмминга (13,8)
FIFO в блоках CAN	Код Хэмминга (13,8)
FIFO в блоке USB	Код Хэмминга (39,32)
FIFO в блоке PCI	Код Хэмминга (39,32)
FIFO в блоке Ethernet	Код Хэмминга (22,16)
Буферы EDCL в блоке Ethernet	Код Хэмминга (22,16)
Внутренняя статическая память	Код Хэмминга (39,32)
Буфер TLB модуля MMU	Код Хэмминга (39,32)
Буфер трассировки шины AHB модуля DSU	Код Хэмминга (39,32)
Буфер трассировки памяти инструкций процессора	Код Хэмминга (39,32)

Внутренняя статическая память (OCRAM). Данные защищаются кодом Хэмминга (39,32). Обнаруживаются две ошибки, исправляется одна битовая ошибка на 39-битное кодовое слово (32 бита данных плюс 7 бит кода коррекции). В случае обнаружения исправляемой ошибки, происходит ее автоматическое исправление в памяти. При обнаружении неисправимой ошибки устанавливается бит NE регистра статуса AHB и генерируется исключение instruction_access_error (при осуществлении выборки инструкций), data_access_error (при считывании данных) или data_store_error (при обращениях вида чтение-модификация-запись). Появление исключения приведет к переходу в таблицу прерываний (помимо выделения нового регистрового окна и сохранения значений PC и nPC в его регистры %l1 и %l2) или к переводу процессора в состояние ошибки, в зависимости от значения бита ET регистра %psr. Сам процесс детектирования ошибки инициализируется запросом на считывание данных или запросом на запись, который выполняется по схеме чтение-модификация-запись. Выключить режим можно глобально, подав «0» на вход CFG_FT_ENA или с помощью регистра FT_CTRL (см. таблицу 4.3.4).

В контроллерах внешней памяти реализованы механизмы помехоустойчивого кодирования (см. таблицу 2.6.2, пункты «4.5.8», «4.6.6» и «4.7.8»).

Таблица 2.6.2 – Защита внешней памяти

PROM*	Код Хэмминга (13,8)
PROM** / SRAM	Код Хэмминга (39,32)
SDRAM	Код Рида-Соломона (96,64) / Код Хэмминга (72,64)

_____

* PROM в режиме 8-битной шины данных.

** PROM в режиме 32-битной шины данных.

Стоит отметить, что минимальное расстояние Хэмминга (минимальное число искаженных символов, необходимое для перехода одной разрешенной комбинации в другую) для всех используемых в процессоре вариантов кода Хэмминга – три. Это означает, что при наличии трех или более ошибок на кодовое слово данный код не способен корректно детектировать неисправимую ошибку, с некоторой вероятностью вместо неё может быть обнаружена исправимая ошибка или кодовое слово будет признано корректным.

Защита кэшей осуществляется в полностью прозрачном для программного обеспечения режиме. Размер кодового слова для тегов и для данных одинаков, и для кэша инструкций, и для кэша данных. Он приведен в таблице 2.6.1. При обнаружении исправимой ошибки ядро использует скорректированные данные, при обнаружении неисправимой ошибки считывание из строки кэша блокируется, ядро инициирует запрос

во внешнюю память для восстановления строки инструкций или данных. Повлиять на данный алгоритм работы можно с помощью регистра управления защитой (%asr16). Установка бита SEI в регистре %asr16 позволяет принудить процессор повторно считывать данные в кэш не только при обнаружении хотя бы одной неисправимой ошибки в строке кэша, но и при обнаружении одной битовой ошибки на кодовое слово. Использование подобного алгоритма действий может быть оправдано, если внешняя кэшируемая память, подключенная к процессору, считается более защищенной от воздействия внешних факторов, и вероятность одновременного возникновения в кэше процессора двух ошибок на кодовое слово не является пренебрежимо малой. Такое поведение будет минимизировать риск возникновения ошибок, которые код Хэмминга может некорректно интерпретировать.

2.7 Основные характеристики микросхем

Электрические параметры и предельно допустимые режимы эксплуатации ИС 1906ВМ016 и ИС 1906ВМ01А6 приведены в таблицах 2.7.1 и 2.7.3 соответственно.

Таблица 2.7.1 – Электрические параметры ИС 1906ВМ016 и ИС 1906ВМ01А6 при приемке и поставке

Наименование параметра, единица измерения, режим измерения		Буквенное обозначение параметра	Норма параметра		Температура среды, °С
			не менее	не более	
1 Выходное напряжение низкого уровня буферов ввода-вывода, В, U_{CC1} = 1,62 В, U_{CC2} = 3,0 В, I_{OL} = 4,0 мА		U_{OL}	–	0,4	–60 ±3 25 ±10 85 ±3[3] 125 ± 5[2]
2 Выходное напряжение высокого уровня буферов ввода-вывода, кроме ERRORN#, WDOGN#, PCI_SERR#, В, U_{CC1} = 1,62 В, U_{CC2} = 3,0 В, I_{OH} = –4,0 мА		U_{OH}	U_{CC2}–0,3	–	
3 Входной ток по входу тактового сигнала CLK, мкА, U_{CC1} = 1,98 В, U_{CC2} = 3,6 В	U_{IL} = 0 В	I_{IL1}	–55	–	
	$U_{IH} = U_{CC2}$	I_{IH1}	–	55	
4 Входной ток, мкА,[1] U_{CC1} =1,98В, U_{CC2} = 3,6 В	по выводам без «pull-up» U_{IL} = 0 В	I_{IL2}	–15	–	
	по выводам без «pull-up» $U_{IH} = U_{CC2}$	I_{IH2}	–	15	
	по выводам с «pull-up»[2] U_{IL} = 0 В	I_{IL3}	–100	–20	
	по выводам с «pull-up»[2] $U_{IH} = U_{CC2}$	I_{IH3}	–15	15	
5 Динамический ток потребления ядра, А, U_{CC1} = 1,98 В, U_{CC2} = 3,6 В	f_{CICLK} = 1 МГц[3]	I_{OCC1}	–	0,02	
	f_{CICLK} = 80 МГц[2]		–	1,6	
6 Динамический ток потребления буфера ввода-вывода, мА, U_{CC1} = 1,98 В, U_{CC2} = 3,6 В, f_{CICLK} = 1 МГц		I_{OCC2}	–	2	
7 Функциональный контроль U_{CC1} = (1,62; 1,98) В, U_{CC2} = (3,0; 3,6) В,	f_{CISPW_CLK} = (2; 200)МГц, f_{CIPCI_CLK} = 66 МГц, f_{CICLK} = (0,001; 100)МГц[3]	ФК	–	–	
	f_{CISPW_CLK} = (2; 150) МГц, f_{CIPCI_CLK} = 33 МГц, f_{CICLK} = (0,001; 80) МГц[2]				

Продолжение таблицы 2.7.2

[1)] Параметры I_{IL2}, I_{IH2}, I_{IH3} при температуре минус 60 ºС не измеряются, а гарантируются нормой при температуре (25 ± 10) ºС.

[2)] Только для ИС 1906ВМ01А6.

[3)] Только для ИС 1906ВМ016.

Таблица 2.7.3 – Предельно допустимые и предельные режимы эксплуатации ИС 1906ВМ016 и ИС 1906ВМ01А6 в диапазоне рабочих температур от минус 60 ºС до плюс 125 ºС

Наименование параметра режима, единица измерения		Буквенное обозначение параметра	Предельно допустимый режим		Предельный режим	
			не менее	не более	не менее	не более
1 Напряжение питания ядра, В		U_{CC1}	1,62	1,98	–0,2	2,4
2 Напряжение питания буферов ввода-вывода, В		U_{CC2}	3,0	3,6	–0,3	4,4
3 Входное напряжение низкого уровня, В		U_{IL}	0	0,6	–0,3	–
4 Входное напряжение высокого уровня, В		U_{IH}	$0,2U_{CC2}+1,0$	U_{CC2}	–	$U_{CC2}+0,3$
5 Выходной ток низкого уровня, мА		I_{OL}	–	4,0	–	6,0
6 Выходной ток высокого уровня, мА		I_{OH}	–4,0	–	–6,0	–
7 Частота следования импульсов тактового сигнала, МГц	1906ВМ016	f_{CICLK}	0,001	100	–	–
	1906ВМ01А6			80		
8 Частота следования импульсов тактового сигнала интерфейса SpaceWire, МГц	1906ВМ016	f_{CISPW_CLK}	2	200	–	–
	1906ВМ01А6			150		
9 Частота следования импульсов тактового сигнала интерфейса PCI 2.2, МГц	1906ВМ016	f_{CIPCI_CLK}	–	66	–	–
	1906ВМ01А6			33		
10 Емкость нагрузки, пФ	1906ВМ016	C_L	–	20	–	80
	1906ВМ01А6		–	80	–	100
Примечание – Время работы в одном из предельных режимов должно быть не более 5 с.						

Для гарантии стабильной работы с внешней памятью (например, подключенной к области SRAM) при тактовой частоте процессора свыше 40 МГц минимальное рекомендуемое количество дополнительных тактов активного состояния чтения равно единице. Данная рекомендация в полной мере справедлива для микросхем со временем доступа в 10 нс или менее. Если время доступа превышает период тактового сигнала микропроцессора, то в первую очередь стоит ориентироваться на параметры микросхемы внешней памяти, а не на это минимальное значение.

3 Начало работы

3.1 Режимы работы процессора

Процессор может функционировать в одном из следующих режимов:
- рабочий режим;
- режим Power Down (см. пункт «4.4.2 Блок целочисленной арифметики LEON4»);
- режим отладки (см. подраздел «4.9 Отладочный модуль LEON4»);
- режим остановки (см. описание битов HL и PE регистра управления DSU);
- режим граничного сканирования (см. подраздел «4.24 Цепь граничного сканирования»).

3.2 Подключение выводов

Процессору необходимы два напряжения питания – 3,3 В (буферы) и 1,8 В (ядро), подключаемые в соответствии с таблицей 2.3.1. Логические уровни сигналов, определяющие режим работы процессора, приведены в таблице 3.2.1.

Таблица 3.2.1 – Конфигурация выводов для определения режима работы

Вывод	Логические уровни сигналов, определяющие режим работы процессора		
	Рабочий режим, Power Down	Режим отладки*	Режим граничного сканирования**
BSD_TRST#	0	0	1
DSU_EN	0	1	0
TEST_MODE	1	1	1
TEST_COMPRESS	1	1	1
CFG_FT_ENA	0/1	0/1	–
CFG_8B_PROM	0/1	0/1	–

_____

* Вход в режим описан в пункте 4.9.2.

** Вход в режим описан в подразделе 4.24.

Некоторые выводы процессора для его корректного функционирования должны быть подключены, даже если интерфейс, которому они принадлежат, не используется. Перечень таких выводов и рекомендации по уровню притяжки, если интерфейс не задействован, приведены в таблице 3.2.2.

Таблица 3.2.2 – Выводы, обязательные к подключению, если интерфейс не используется

Интерфейс	Обозначение вывода	Рекомендуемый уровень притяжки
Ethernet	ETX_CLK	0
	ERX_CLK	0
PCI	PCI_CLK	0
	PCI_RST#	0
SpaceWire	SPW_CLK	0
MIL-STD-1553 / AS15531	CLK_1553	0
USB 2.0	USB_CLKOUT	0
JTAG	JTAG_TCK	0

Остальные выводы неиспользуемых интерфейсов, а также все выводы, не приведенные в таблицах 3.2.1 и 3.2.2, можно оставить неподключенными.

На вход CLK следует подавать тактовый сигнал.

На входе RESETN# необходимо выставить низкий логический уровень сигнала, захват данного состояния осуществляется по положительному фронту CLK. На основе RESETN# генерируется внутренний сигнал сброса (см. подраздел «4.1 Система сброса»). После сброса процессор стартует с адреса 0x00000000 (область PROM). Соответственно, к области PROM должен быть подключен некоторый набор микросхем памяти (см. подраздел «4.5 Контроллер PROM / IO»).

3.3 Внешнее ОЗУ

Если в качестве внешнего ОЗУ подключена память SDRAM, то перед её использованием необходимо настроить контроллер SDRAM в соответствии с «4.7.5 Инициализация микросхем динамической памяти» и «4.7.7 Управление параметрами конфигурации SDRAM». Важно помнить, что для активации контроллера SDRAM и запуска инициализационной последовательности недостаточно установить бит FT_CTRL.RS, помимо этого требуется произвести запись в регистр SDRAM_CFG_1. Сигнал тактирования на микросхемы SDRAM должен приходить с опережением приблизительно на 2 нс (в зависимости от параметров используемых чипов) относительно CLK, как это приведено на рисунке 3.3.1. Более подробная информация по данному вопросу представлена в пункте «4.7.3 Тактирование синхронной динамической памяти».

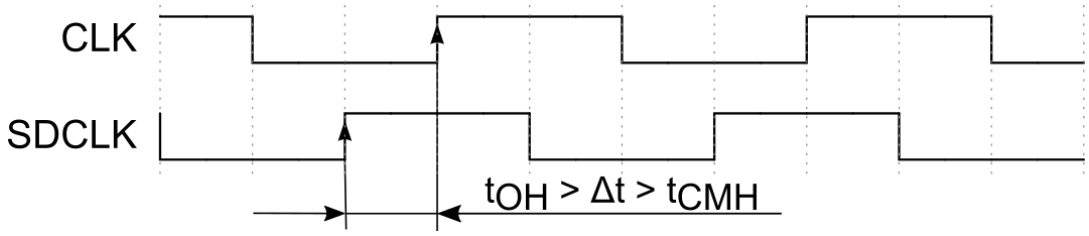


Рисунок 3.3.1 – Взаиморасположение фронтов сигналов CLK и SDCLK

Одним из вариантов организации требуемого взаиморасположения сигналов CLK и SDCLK является использование функционала, описанного в пункте «4.7.4 Блок управления трактом задержки SDCLK».

Если в качестве внешнего ОЗУ используется SRAM (используется по умолчанию после сброса), следует учесть рекомендации, которые приведены в конце подраздела «2.7 Основные характеристики микросхем».

3.4 Инициализация кэша, конфигурация доступа к PROM

После сброса процессора использование кэша инструкций и кэша данных запрещено. Из-за особенностей реализации системы помехоустойчивого кодирования, к процессу инициализации кэшей есть дополнительное требование, не присутствующее в стандарте SPARC V8. Перед включением кэшей, помимо сброса меток, также требуется осуществить инициализацию данных обоих кэшей всех каналов, за исключением нулевого (либо для всех каналов). Игнорирование данной особенности может приводить к непредсказуемому поведению микропроцессора. В подпункте «Разрешение и запрещения использования кэшей» пункта «4.4.3» приведен ассемблерный код, позволяющий выполнить описанные операции.

Перед выполнением инициализации кэша настоятельно рекомендуется выставить корректное значение дополнительных тактов доступа для области PROM (см. «Регистр управления тактами ожидания контроллера памяти»), поскольку по умолчанию используется их максимальное значение. А до включения кэша инструкций каждая итерация циклического участка кода будет приводить к выполнению как минимум одной предвыборки инструкций (восьми последовательных обращений во внешнюю память или тридцати двух, если используется 8-битный режим шины данных PROM). Игнорирование данной рекомендации приведет к существенному неоправданному замедлению, как процесса инициализации, так и дальнейшей работы процессора.

3.5 Запуск программы из ПЗУ

В качестве ПЗУ (PROM) может использоваться память типа ROM, PROM, Flash, EEPROM или MRAM.

Для компиляции ПО рекомендуется использовать компилятор **GCC** (начиная с версии 4.4.2) и утилиту **mkprom2** (http://www.gaisler.com/anonftp/mkprom2/), которая требует модификации в части инициализации кэш программ и данных перед их включением (см. подраздел «3.4 Инициализация кэша, конфигурация доступа к PROM»).

Утилита **mkprom2** выполняет следующие действия:

- сжимает пользовательский код с коэффициентом приблизительно равным 2;
- добавляет код для распаковки и переноса программы в ОЗУ;
- добавляет код инициализации блоков процессора;
- разрешает работу кэша программ и кэша данных.

Что происходит при запуске прошивки, созданной **mkprom2**:

- инициализация блоков микропроцессора (к ним не относятся контроллер конфигураций и контроллеры PROM / IO, SRAM и SDRAM);
- распаковка основного исполняемого кода в ОЗУ (стартовый адрес по умолчанию – 0x40000000);
- передача управления основной программе.

Стоит отметить, что **mkprom2** не инициализирует указатель на вершину стека (%sp). Инициализация %sp может быть выполнена либо через использование ключа «-stack», либо в пользовательском коде.

Важно также отметить, что утилита **mkprom2** при инициализации включает кэш программ и кэш данных. В связи с этим требуется произвести модификацию её исходного кода, согласно тексту подраздела «3.4» (добавить ассемблерный код из подпункта «Разрешение и запрещения использования кэшей» пункта «4.4.3» в файл ./src/prominit.S), и перекомпилировать **mkprom2**.

Пример генерации программы ПЗУ с помощью GCC:

sparc-elf-gcc -mcpu=v8 -O2 -g -o hello.o hello_world.c

Используемые ключи:

-O2 – задать 2-й тип оптимизации исходных кодов при компиляции. Тип может быть изменен в зависимости от желаемых критериев оптимизации;

-o <имя_файла> – задать имя выходного файла. Не является обязательным, по умолчанию генерируется файл с именем исходного файла и расширением «.o»;

-mcpu=v8– разрешение на использование инструкций аппаратного умножения и деления (smul, umul, sdiv, udiv);
-g – добавлять в итоговый файл отладочную информацию (имена меток, констант и т.д.). Не является обязательным.

Пример генерации программы ПЗУ с помощью **mkprom2**:

mkprom2 -mcpu=v8 -freq 50 -o hello.out hello.o

Используемые ключи:
-o <имя_файла> – задать имя выходного файла;
-mcpu=v8 – разрешение на использование инструкций аппаратного умножения и деления (smul, umul, sdiv, udiv);
-freq <X МГц> – тактовая частота микропроцессора в МГц.

Если из файла прошивки hello.out требуется сформировать прошивку формата SREC (Motorola S-record), то для этого достаточно выполнить следующую команду:

sparc-elf-objcopy -O srec hello.out hello.srec

Используемые ключи:
-O <формат> – формат выходного файла.

Если требуется интегрировать пользовательскую инициализационную последовательность, то для этого требуется создать файл bdinit.S, содержащий требуемый набор инструкций ассемблера, при запуске утилиты MKPROM2 добавить ключ «-bdinit» (без кавычек). Полная последовательность действий будет выглядеть следующим образом:

1. **sparc-elf-gcc** -c -O2 -g -o bdinit.o bdinit.S //(ключ “-c” – не линковать)
2. **sparc-elf-gcc** -mcpu=v8 -O2 -g -o hello.o hello_world.c
3. **mkprom2** -bdinit -mcpu=v8 -freq 50 -o hello.out hello.o

В качестве примера ниже приведено содержимое файла bdinit.S, осуществляющего активацию драйверов LVDS модулей SpaceWire:

```
#define LVDS_CTRL_ADDRESS   0x80100000
#define LVDS_CTRL_VALUE     0xFFFFFFFF

        .text
        .global bdinit0, bdinit1, bdinit2

bdinit0:
        retl
         nop

bdinit1:
        sethi   %hi(LVDS_CTRL_ADDRESS), %o0
        set     LVDS_CTRL_VALUE, %g1
        retl
         st     %g1, [%o0 + %lo(LVDS_CTRL_ADDRESS)]
```

bdinit2:

retl

nop

.align 4

Адрес LVDS_CTRL_ADDRESS приведен в соответствии с подразделом «2.5 Карта памяти микропроцессор» и содержимым таблицы 4.3.4, а LVDS_CTRL_VALUE – требуемому значению подпункта «Регистр управления драйверами LVDS». Информация об особенностях использования ключа «–bdinit» приведена в сопроводительной документации к MKPROM2.

3.6 Отладка программ

Для отладки пользовательских программ может использоваться одно из следующих средств:

- интегральная среда разработки и отладки (ИСР) программно-аппаратного комплекса (ПАК) разработки НИИЭТ (см. раздел «7 Программно-аппаратный комплекс средств разработки и отладки»);
- консольное приложение GRAIP, входящее в состав ПАК;
- консольное приложение GRMON2 Professional (платная версия) от фирмы Cobham Gaisler. С бесплатной версией GRMON2 отладка работать не будет.

ИСР для ведения отладки использует интерфейс JTAG.

Отладчик GRAIP может использовать один из следующих интерфейсов:

- Ethernet:

  graip_eth.exe

- UART:

  graip_ahbuart.exe

Конфигурацию параметров отладочных интерфейсов Ethernet и UART можно выполнить через внесение изменений в файл settings.txt.

Отладчик GRMON2 может использовать один из следующих интерфейсов:

- JTAG:

  grmon2 -digilent -u

  Пример приведен для случая использования кабеля Digilent JTAG HS1, поддерживаемые варианты приведены в сопроводительной документации к GRMON2.

- Ethernet:

  grmon2 -eth 192.168.0.51 -u

  192.168.0.51 – IP адрес по умолчанию (см. таблицу 4.18.17) модуля EDCL в пункте «4.18.6 Линия связи отладки Ethernet (EDCL)»).

- UART:

  grmon2 -uart <device> -baud <baudrate> -u

  В качестве <device> требуется указать /dev/tty## (для операционной системы Linux) либо \\.\com# (для операционной системы Windows), где «#» обозначает номер интерфейса, <baudrate> – скорость передачи в бодах.

- SpaceWire (через аппаратный мост GRESB Ethernet to SpaceWire).

  Подробная информация по ключам запуска приведена в сопроводительной документации GRMON2 и GRESB.

Дополнительный ключ «-u [N]» (по умолчанию используется N = 0, соответствующее UART 1) следует использовать для организации вывода информации в консоль отладки. Он требуется, если к UART 1 для этой цели не подключен сторонний терминал.

Например, для проведения отладки встраиваемой операционной системы (например, Linux) необходимо использовать дополнительный ключ «**-nb**». Этот ключ запрещает GRMON2 переходить в режим отладки при возникновении системного прерывания, вызванного обнаружением ошибки. Вместо данного механизма должна быть реализована обработка таких прерываний на стороне тестируемой операционной системы. Более подробная информация приведена в сопроводительной документации GRMON2.

3.7 Сторожевой таймер

Сторожевой таймер начинает декрементироваться непосредственно после сброса питания, бит разрешения установлен по умолчанию. Значение предделителя системной частоты по сбросу – 0xFFFF, счетчика сторожевого таймера – 0x0000FFFF. При достижении счетчиком нулевого отсчета будет сгенерирован низкий логический уровень сигнала на выводе WDOGN# и системное прерывание (оно будет обработано, только если это разрешено в контроллере прерываний, см. подраздел «4.8»), если установлен бит IE регистра управления таймером.

Если оставить значения управляющих регистров по умолчанию, то при системной частоте 50 МГц обратное переполнение счетчика сторожевого таймера произойдет приблизительно через 86 секунд.

Боле подробное описание функционирования сторожевого таймера приведено в подразделе «4.13 Блок таймеров общего назначения».

3.8 Требования по инициализации областей памяти при использовании помехоустойчивого кодирования

При осуществлении выборки инструкций, обращения чтения данных из неинициализированной области памяти или записи, которая обрабатывается через механизм чтение-модификация-запись, генерируется исключение (instruction_access_error, data_access_error и data_store_error соответственно). Появление исключения приведет к переходу в таблицу прерываний (помимо выделения нового регистрового окна и сохранения значений PC и nPC в его регистры %l1 и %l2) или к переводу процессора в состояние ошибки, в зависимости от значения бита ET регистра %psr.

Формирование корректных проверочных бит при записи и их использование при считывании происходит, только если и на входе CFG_FT_ENA установлен высокий логический уровень сигнала и в регистре FT_CTRL установлен соответствующий бит управления. Если хотя бы одно из условий не выполняется, то при осуществлении обращения записи на шине FT_CHECKBITS[31:0] выставляется значение 0x00000000, при считывании данных значение данной шины игнорируется.

Инициализировать память корректными проверочными битами можно двумя способами:

1) При отключенном помехоустойчивом кодировании для данной области.

Подходит только для заполнения нулевыми данными, поскольку на шине FT_CHECKBITS[31:0] формируется детерминированное значение (0x00000000), которое соответствует нулевым данным. Плюсом данного подхода является то, что подходят обращения записи любой разрядности (8-, 16-, 32-, 64-битные).

2) При включенном помехоустойчивом кодировании для данной области.

Заполнение памяти может производиться произвольными данными, но разрядность обращений записи должна быть либо равна, либо превосходить размер защищаемого

слова данных (см. подраздел «2.6 Механизмы сбоеустойчивости»). Для области PROM в 8-битном режиме шины данных подойдут обращения записи любой разрядности (8-, 16-, 32-, 64-битные). Для областей SRAM, OCRAM и PROM в режиме 32-битной шины данных подойдут как 32-битные, так и 64-битные обращения. Для области SDRAM только 64-битные.

Требования и рекомендации по инициализации:

1) В обязательном порядке требуется инициализировать место, предназначенное под стек, поскольку его использование в большинстве случаев не контролируется в пользовательском коде;
2) Настоятельно рекомендуется инициализировать области, выделенные под динамически создаваемые объекты, массивы и переменные. Данный шаг можно пропустить, только если ни пользовательский код, ни код библиотек, задействованных в программном продукте, не будут генерировать обращения вида чтение-модификацию-запись к таким объектам;
3) В случае использования SDRAM с помехоустойчивым кодированием все обращения записи, кроме 64-разрядных, осуществляются через механизм чтение-модификация-запись. В связи с этим требуется либо организовать распаковку и перенос программы в ОЗУ с использованием только 64-битных обращений записи (инструкции STD), либо предварительно инициализировать еще и области, в которые будет осуществляться сохранение исполняемого кода и статических данных программы (области «.text», «.data» и т.д.).

4 Функциональное описание

4.1 Система сброса

4.1.1 Общие сведения

Генератор сброса RSTGEN реализует синхронизацию, а также фильтрацию кратковременных выбросов входного сигнала сброса и генерирует внутренний сигнал сброса. Входной сигнал сброса может быть асинхронным, но в таком случае его длительность должна быть больше периода тактовой частоты.

После прохождения сигнала сброса процессор стартует с адреса 0x00000000.

4.1.2 Принцип работы

Генератор сброса защелкивает значение сигнала блокировки тактирования (CLKLOCK), который в данной реализации имеет постоянный уровень логической единицы, по каждому переднему фронту тактирования. Сигнал блокировки используется как входной в 5-битовом регистре сдвига. Три старших значащих бита данного регистра сдвига синхронизированы в выходном регистре сброса. Итоговый сигнал сброса будет находиться в высоком логическом состоянии, только если и выходной регистр сброса и входной сигнал сброса находятся в высоком логическом состоянии. Поскольку выходной регистр зависит от тактирования системы, активное низкое логическое состояние выхода сброса переходит в высокое логическое состояние синхронно с тактированием системы.

Регистр сдвига ядра имеет синхронный сброс, при этом сигналы сброса не соединены с регистром выхода сброса, см. рисунок 4.1.1. Следует обратить внимание на то, что выходной сигнал сброса при запуске входного сигнала сброса всегда переходит в низкое логическое состояние.

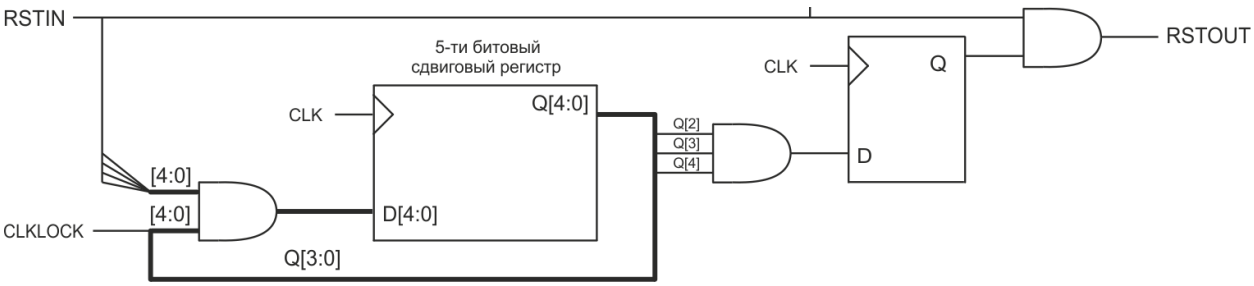


Рисунок 4.1.1 – Генератор сброса

Сигнал с входа RESETN# подается непосредственно на RSTIN. Сигнал RSTOUT поступает на все внутренние модули процессора, которым требуется осуществлять сброс к изначальному состоянию.

4.2 Система тактирования

В таблице 4.2.1 приведены тактовые входы процессоров 1906BM016, 1906BM01A6.

Таблица 4.2.1 – Тактовые входы

Название	Описание
CLK	Тактовый сигнал процессора, шины AMBA и внутренних блоков
SPW_CLK	Тактовый сигнал интерфейсов SpaceWire
CLK_1553	Тактовый сигнал интерфейсов Mil-STD-1553
ERX_CLK	Тактовый сигнал приемника интерфейса Ethernet
ETX_CLK	Тактовый сигнал передатчика интерфейса Ethernet
PCI_CLK	Тактовый сигнал интерфейса PCI
USB_CLKOUT	Тактовый сигнал интерфейса USB
JTAG_ TCK	Вход TCK интерфейса JTAG и цепи граничного сканирования

Тактовая частота SPW_CLK должна быть кратна 10 МГц. Более подробная информация приведена в подпункте «Установка скорости канала связи» пункта «4.17.3 Интерфейс связи» и в пункте «4.17.9 Аппаратные особенности».

На вход CLK_1553 следует подавать тактовый сигнал с частотой 20 МГц. Дополнительная информация приведена в пункте «4.16.7 Тактирование».

Информация о минимально допустимых тактовых частотах интерфейса Ethernet приведена в подпункте «Тактирование» пункта «4.18.2 Принцип работы».

Тактовая частота интерфейса PCI задается в соответствии со спецификацией «PCI Local Bus Specification Revision 2.3».

На вход USB_CLKOUT следует подавать тактовый сигнал с частотой 60 МГц. Дополнительная информация приведена в подпункте «Тактирование и сброс» пункта «4.20.2 Принцип работы».

SDCLK – выходной тактовый сигнал SDRAM. Его частота определяется основным тактовым сигналом (CLK). Данный выход настроен таким образом, чтобы совпадать по фазе с внутренним деревом тактирования.

4.3 Контроллер конфигурации

4.3.1 Общие сведения

Контроллер конфигурации используется для управления режимами работы блоков ввода/вывода, контроллеров памяти, блоков обнаружения и исправления ошибок. Блок-схема, описывающая общие принципы построения системы, представлена на рисунке 4.3.1. Конфигурирование осуществляется через набор регистров, доступных по шине APB с адреса 0x80100000.

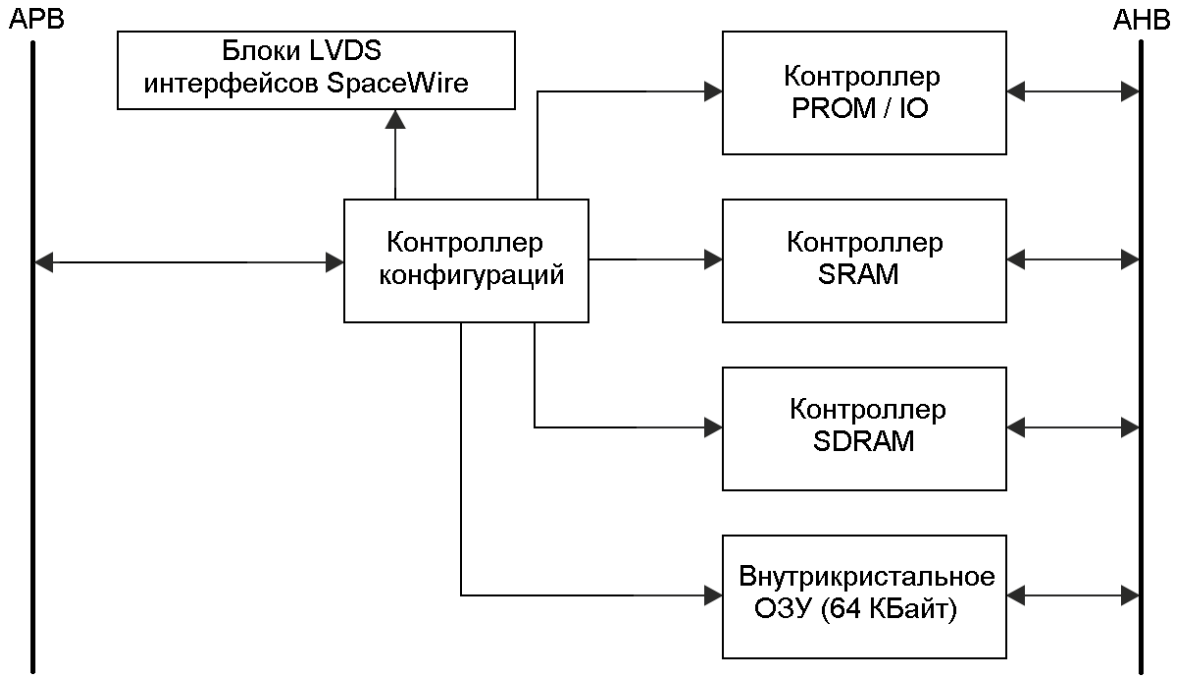


Рисунок 4.3.1 – Блок-схема управления контроллерами памяти

4.3.2 Принцип работы

Модуль состоит из подчиненного устройства на шине APB и набора управляющих регистров. Регистры блока доступны через интерфейс APB, (см. пункт 4.3.4).

Регистр управления драйверами LVDS (LVDS_CTRL) управляет драйверами LVDS модулей SpaceWire.

Регистр управления режимами помехоустойчивого кодирования (FT_CTRL) управляет отключением режима проверки, разрешением перезаписи данных в случае обнаружения исправимой ошибки, генерации проверочных бит для контроллеров памяти PROM, SRAM, SDRAM, внутрикристального ОЗУ.

Регистр управления тактами ожидания контроллера памяти (WC_CTRL) определяет количество дополнительных тактов ожидания (Wait Cycles) для контроллеров памяти PROM / IO / SRAM для чтения и записи. Значение регистра следует задавать в соответствии с параметрами используемых микросхем и текущим значением тактовой частоты.

Регистр управления контроллерами памяти (MEM_CTRL) определяет наличие дополнительных тактов lead_in / lead_out для контроллеров памяти, использование систем сигнализации о готовности и сигнализации об ошибке доступа.

Регистр XOR маски проверочных бит внешней памяти (CHECKBITS_XOR) задает маску, которая складывается по модулю 2 с контрольными битами интерфейса внешней памяти (PROM, SRAM, SDRAM). Основное назначение – тестирование механизма коррекции ошибок, организация управляющих последовательностей записи во внешнюю память.

Регистры управления SDRAM (SDRAM_CFG_1 и SDRAM_CFG_2) используются для управления контроллером динамической памяти.

Регистр управления трактом задержки SDCLK (SDCLK_CTRL) контролирует длину тракта, по которому проходит сигнал от входа CLK до вывода SDCLK.

Регистр управления механизмом обхода чтения/записи проверочных бит (BYPASS_CTRL) отвечает за активацию и деактивацию выбранного механизма тестового доступа к шине FT_CHECKBITS. Он позволяет считать данные с шины FT_CHECKBITS в регистр BYPASS_CHECKBITS, установить на шине FT_CHECKBITS

требуемое значение через регистр BYPASS_CHECKBITS или применить к шине маску CHECKBITS_XOR.

Регистр механизма обхода чтения/записи проверочных бит (BYPASS_CHECKBITS) позволяет считать значение с шины FT_CHECKBITS или же задать на ней значение при обращении чтения или записи, если активирован соответствующий режим. Основное назначение – тестирование механизма коррекции ошибок, организация управляющих последовательностей записи во внешнюю память.

4.3.3 Формат представления регистров

Формат представления регистров управления приведен в таблице 4.3.1.

Таблица 4.3.1 – Пример организации записи регистра

31 24	23 16	15 8	7 0
П3	П2	П1	П0
Значение по сбросу П3	Значение по сбросу П2	Значение по сбросу П1	Значение по сбросу П0
Тип поля/байта П3	Тип поля/байта П2	Тип поля/байта П1	Тип поля/байта П0

31–24 Поле 3 (П3) – Описание поля/бита
23–16 Поле 2 (П2) – Описание поля/бита
15–8 Поле 1 (П1) – Описание поля/бита
7–0 Поле 0 (П0) – Описание поля/бита

В таблицах 4.3.2 и 4.3.3 приведены списки возможных состояний ячеек регистров управления.

Таблица 4.3.2 – Список возможных состояний ячеек значения по сбросу

Значение	Описание
0	Значение по сбросу – 0. Используется для отдельных бит
1	Значение по сбросу – 1. Используется для отдельных бит
0xNN	Шестнадцатеричное представление числа. Используется для полей
n/r	Не сбрасывается
*	Особые условия сброса, которые разъяснены в описании соответствующего поля/бита

Таблица 4.3.3 – Список возможных состояний ячеек тип поля/бита

Значение	Описание
r	Только чтение
rw	Чтение/запись
wc	Сброс по записи. Поле доступно для считывания, при записи 1 оно сбрасывается. Запись 0 игнорируется

4.3.4 Регистры

Управление устройством осуществляется через регистры, отображенные в адресуемом пространстве APB. Список используемых регистров представлен в таблице 4.3.4, детальное описание каждого из них представлено в таблицах 4.3.5 – 4.3.15.

Таблица 4.3.4 – Регистры контроллера конфигураций (базовый адрес 0x80100000)

Смещение адреса APB	Регистр	Краткое имя
0x0	Регистр управления драйверами LVDS	LVDS_CTRL
0x4	Регистр управления режимами помехоустойчивого кодирования	FT_CTRL
0x8	Регистр управления тактами ожидания контроллера памяти	WC_CTRL
0xC	Регистр управления контроллерами памяти	MEM_CTRL
0x10	Регистр XOR маски проверочных бит внешней памяти	CHECKBITS_XOR
0x14	Регистр управления SDRAM CFG 1	SDRAM_CFG_1
0x18	Регистр управления SDRAM CFG 2	SDRAM_CFG_2
0x1C	Регистр управления трактом задержки SDCLK	SDCLK_CTRL
0x20	Регистр управления механизмом обхода чтения/записи проверочных бит	BYPASS_CTRL
0x24	Регистр механизма обхода чтения/записи проверочных бит	BYPASS_CHECKBITS

Регистр управления драйверами LVDS

Регистр управления драйверами LVDS (LVDS_CTRL) управляет драйверами LVDS модулей SpaceWire.

Таблица 4.3.5 – Регистр управления драйверами LVDS

31 21	20	19	18	17	16 0
–	E4	E3	E2	E1	–
0x7FF	0	0	0	0	0x1FFFF
r	rw	rw	rw	rw	r

31 – 21 Зарезервировано

20 Разрешение LVDS 4 (E4) – Если бит установлен, то драйвера LVDS SpaceWire 4 включены

19 Разрешение LVDS 3 (E3) – Если бит установлен, то драйвера LVDS SpaceWire 3 включены

18 Разрешение LVDS 2 (E2) – Если бит установлен, то драйвера LVDS SpaceWire 2 включены

17 Разрешение LVDS 1 (E1) – Если бит установлен, то драйвера LVDS SpaceWire 1 включены

16 – 0 Зарезервировано

Регистр управления режимами помехоустойчивого кодирования

Регистр управления режимами помехоустойчивого кодирования (FT_CTRL) управляет отключением режима коррекции ошибок и генерацией проверочных бит для контроллеров памяти PROM, SRAM, SDRAM и внутрикристального ОЗУ, регулирует функцию перезаписи данных при обнаружении исправимой ошибки.

Таблица 4.3.6 – Регистр управления режимами помехоустойчивого кодирования

31 11	10	9	8	7	6	5	4	3	2	1	0
–	WBSD	WBS	WBIO	WBP	RS	–	SDC	SFT	SDFT	PFT	OCFT
0x1FFFFF	1	1	0	0	0	1	1	1	1	1	1
r	rw	rw	r	rw	rw	r	rw	rw	rw	rw	rw

31 – 11	Зарезервировано
10	Разрешение перезаписи данных для SDRAM (WBSD) – Если бит установлен, то разрешена перезапись данных в случае обнаружения исправимой ошибки для SDRAM
9	Разрешение перезаписи данных для SRAM (WBS) – Если бит установлен, то разрешена перезапись данных в случае обнаружения исправимой ошибки для SRAM
8	Разрешение перезаписи данных для области ввода-вывода (WBIO) – Если бит установлен, то разрешена перезапись данных в случае обнаружения исправимой ошибки для области ввода-вывода. Данный функционал не поддерживается, поскольку данная область не защищена помехоустойчивым кодированием
7	Разрешение перезаписи данных для PROM (WBP) – Если бит установлен, то разрешена перезапись данных в случае обнаружения исправимой ошибки для PROM
6	Выбор типа оперативной памяти (RS) – Установка данного бита является обязательным условием для передачи управления областью памяти RAM, адресуемой с 0x40000000, контроллеру SDRAM. Если бит сброшен, область RAM переходит под управление контроллера SRAM. Следует отметить, что для активации контроллера SDRAM и запуска инициализационной последовательности недостаточно установить данный бит, помимо этого требуется произвести запись в регистр управления SDRAM_CFG_1.
5	Зарезервировано
4	Тип корректирующего кода SDRAM (SDC) – Если бит установлен, то для коррекции ошибок используется код Рида-Соломона, если он сброшен – код Хэмминга
3	Включение коррекции ошибок для SRAM (SFT) – Если бит установлен, то коррекция ошибок кодом Хэмминга для SRAM активирована
2	Включение коррекции ошибок для SDRAM (SDFT) – Если бит установлен, то коррекция ошибок для SDRAM активирована
1	Включение коррекции ошибок для PROM (PFT) – Если бит установлен, то коррекция ошибок кодом Хэмминга для PROM активирована
0	Включение коррекции ошибок для внутрикристального ОЗУ (OCFT) – Если бит установлен, то коррекция ошибок кодом Хэмминга для внутрикристального ОЗУ активирована

Регистр управления тактами ожидания контроллера памяти

Регистр управления тактами ожидания контроллера памяти (WC_CTRL) определяет количество дополнительных тактов ожидания для контроллеров памяти PROM/IO и SRAM при обращении к памяти. Значение полей регистра следует задавать исходя из временных параметров используемых микросхем и текущей тактовой частоты процессора. Длительность активного состояния управляющего сигнала (например, WRITEN) для контроллера PROM / IO рассчитывается как (1 + PWWC). Допустимые значения для каждого поля от 0 до 31.

Таблица 4.3.7 – Регистр управления тактами ожидания контроллера памяти

31 30 29	25 24	20 19	15 14	10 9	5 4	0
–	IWWC	IRWC	SWWC	SRWC	PWWC	PRWC
0x0	0x00	0x00	0x00	0x00	0x1E	0x1E
r	rw	rw	rw	rw	rw	rw

31 – 30 Зарезервировано

29 – 25 Такты ожидания при записи для области ввода-вывода (IWWC) – Количество дополнительных тактов системной частоты доступа к памяти для IO контроллера при записи

24 – 20 Такты ожидания при чтении для области ввода-вывода (IRWC) – Количество дополнительных тактов системной частоты доступа к памяти для IO контроллера при чтении

19 –15 Такты ожидания при записи для SRAM (SWWC) – Количество дополнительных тактов системной частоты доступа к памяти для SRAM контроллера при записи

14 –10 Такты ожидания при чтении для SRAM (SRWC) – Количество дополнительных тактов системной частоты доступа к памяти для SRAM контроллера при чтении

9 – 5 Такты ожидания при записи для PROM (PWWC) – Количество дополнительных тактов системной частоты доступа к памяти для PROM контроллера при записи

4 – 0 Такты ожидания при чтении для PROM (PRWC) – Количество дополнительных тактов системной частоты доступа к памяти для PROM контроллера при чтении

Регистр управления контроллерами памяти

Регистр управления контроллерами памяти (MEM_CTRL) определяет наличие такта lead_in и дополнительного lead_out для контроллеров памяти PROM / IO и SRAM.

Таблица 4.3.8 – Регистр управления контроллерами памяти

31 – 9	8	7	6	5	4	3	2	1	0
–	BEXCN	SBRDY	IBRDY	PBRDY	PROM_8B	LOS	LIS	LOP	LIP
0	0	0	0	0	*	0	0	1	1
r	rw	rw	rw	rw	rw	rw	rw	rw	rw

31 – 9 Зарезервировано

8 Разрешение сигнализации об ошибке доступа (BEXCN) – Если бит установлен, то внешний сигнал BEXCN используется для сигнализации об ошибке доступа для контроллеров PROM / IO и SRAM

7 Разрешение использования BRDYN для области SRAM(SBRDY) – Если бит установлен, то сигнал BRDYN будет использоваться в качестве индикатора готовности данных на шине при обращениях к области SRAM

6 Разрешение использования BRDYN для области IO (IBRDY) – Если бит установлен, то сигнал BRDYN будет использоваться в качестве индикатора готовности данных на шине при обращениях к области IO

5 Разрешение использования BRDYN для области PROM (PBRDY) – Если бит установлен, то сигнал BRDYN будет использоваться в качестве индикатора готовности данных на шине при обращениях к области PROM

4 Режим 8-битного доступа (PROM_8B) – Если бит установлен, то активируется 8-битный режим доступа к PROM (используются старшие 8 бит [31:24]), если данный бит сброшен – 32-битная шина данных PROM. Значение по сбросу захватывается с входа CFG_8B_PROM

3 Lead_out для SRAM (LOS) – Наличие дополнительного такта lead_out для операций записи SRAM контроллера

2 Lead_in для SRAM (LIS) – Наличие дополнительного такта lead_in для операций записи SRAM контроллера

1 Lead_out для PROM / IO (LOP) – Наличие дополнительного такта lead_out для операций записи PROM / IO контроллера

0 Lead_in для PROM / IO (LIP) – Наличие дополнительного такта lead_in для операций записи PROM / IO контроллера

Регистр XOR маски проверочных бит внешней памяти

Регистр XOR маски проверочных бит внешней памяти (CHECKBITS_XOR) задает маску, которая складывается по модулю 2 с контрольными битами интерфейса внешней памяти (PROM, SRAM, SDRAM). Данный регистр может быть использован для тестирования механизма коррекции ошибок и организации записи управляющих последовательностей во внешнюю память FLASH.

Таблица 4.3.9 – Регистр XOR маски проверочных бит внешней памяти

31	0
CHECKBIT_XOR	
0	
rw	

31 – 0 Регистр маски контрольных бит (CHECKBIT_XOR) – Рассчитывает, записываемые в память контрольные биты, как ИСКЛЮЧАЮЩЕЕ ИЛИ (XOR) между регистром CHECKBITS_XOR и контрольными битами, сгенерированными контроллером внешней памяти

Регистры управления SDRAM

Регистры управления SDRAM (SDRAM_CFG_1 и SDRAM_CFG_2) используются для управления контроллером динамической памяти и конфигурирования его временных параметров. Значение полей регистров следует задавать исходя из временных параметров используемых микросхем и текущей тактовой частоты процессора.

Таблица 4.3.10 – Регистр управления SDRAM_CFG_1

31 – 24	23 – 22	21 – 19	18 – 16	15 – 0
–	CL	CAW	RAW	IP
0	0x3	0x1	0x5	0x273A
r	rw	rw	rw	rw

31 – 24 Зарезервировано

23 – 22 Время задержки строб-импульса адреса столбца (CL) – два младших бита поля «CAS Latency» стандартного регистра управления динамической памятью (Mode Register). Старший бит всегда равен 0. При записи в регистр SDRAM_CFG_1 устанавливается запрос на выполнение команды LOAD MODE REGISTER. Этот запрос не будет обработан, если бит RS регистра FT_CTRL сброшен. Первое выполнение данной команды предваряется генерацией инициализирующей последовательности SDRAM. В дальнейшем выполняется только команда LOAD MODE REGISTER с новым значением Mode Register, с предварительной подачей PRECHARGE для всех банков данных, если есть активированный ряд. Стоит отметить, что значение CL = 0 запрещено стандартом SDRAM. Помимо этого в спецификации микросхемы приведены допустимые для неё значения CAS Latency (как правило, два или три такта). Значение CAS Latency = 1 не поддерживается контроллером

21 – 19 Ширина адреса столбцов (CAW) – Определяет количество столбцов в динамической памяти, вычисляемое по формуле $2^{(CAW + 8)}$. Допустимые значения от 0 до 4, соответственно от 256 до 4096 столбцов

18 – 16 Ширина адреса ряда (RAW) – Определяет количество рядов в динамической памяти, вычисляемое по формуле $2^{(RAW + 8)}$. Допустимые значения от 0 до 5, соответственно от 256 до 8192 рядов

15 – 0 Период инициализации (IP) – Период инициализации динамической памяти в тактах системной шины контроллера SDRAM. Значение включает в себя необходимое для инициализации памяти количество тактов (как правило, 100 или 200 мкс), а также величину 5 + tRP + 8 × tRFC + tMRD, требуемую для выполнения инициализирующей последовательности

Таблица 4.3.11 – Регистр управления SDRAM_CFG_2

31 30	29 19	18 16	15 12	11 8	7 4	3 0
PM	RP	tMRD	tWR	tRCD	tRFC	tRP
0x0	0x185	0x3	0x1	0x2	0x4	0x2
rw	rw	rw	rw	rw	rw	rw

31 – 30 Режим подачи команд PRECHARGE (PM). Доступные значения: 0x0 – режим с одним активным рядом; 0x1 – зарезервировано; 0x2 – режим чтения/записи с автоматическим PRECHARGE; 0x3 – режим с PRECHARGE после чтения/записи

29 – 19 Период обновления (RP) – количество тактов системной частоты между последовательными командами AUTO REFRESH.
Значение поля RP должно быть выбрано исходя из соотношения: [AUTO_REFRESH_CYCLE_NS × CLOCK_FREQUENCY_MHZ / 1000], где [] – обозначает взятие целой части от числа

18 – 16 Регистр управления параметром tMRD динамической памяти. Задает количество тактов NOP между загрузкой Mode Register и последующими командами ACTIVE или REFRESH

15 – 12 Регистр управления параметром tWR динамической памяти. Задает количество тактов NOP между командами WRITE и PRECHARGE. Значение должно быть выбрано исходя из соотношения [t_{WR}(нс) × CLOCK_FREQUENCY_MHZ / 1000], где [] – обозначает взятие целой части от числа

11 – 8 Регистр управления параметром tRCD динамической памяти. Задает количество тактов NOP между командой ACTIVE и командами READ или WRITE. Значение должно быть выбрано исходя из соотношения [t_{RCD}(нс) × CLOCK_FREQUENCY_MHZ / 1000], где [] – обозначает взятие целой части от числа

7 – 4 Регистр управления параметром tRFC динамической памяти. Задает количество тактов NOP после команды AUTO REFRESH. Значение должно быть выбрано исходя из соотношения [t_{RFC}(нс) × CLOCK_FREQUENCY_MHZ / 1000], где [] – обозначает взятие целой части от числа

3 – 0 Регистр управления параметром tRP динамической памяти. Задает количество тактов NOP после команды PRECHARGE. Значение должно быть выбрано исходя из соотношения [t_{RP}(нс) × CLOCK_FREQUENCY_MHZ / 1000], где [] – обозначает взятие целой части от числа

Регистр управления трактом задержки SDCLK

Основной рекомендацией при настройке тракта задержки является использование минимального количества элементов задержки. Для организации значительного сдвига вместо простого увеличения значения поля SDCD более перспективно использовать инверсию сигнала (бит SDCI установлен) с меньшим значением SDCD. Такой подход позволяет получить лучшую стабильность параметров в процессе эксплуатации схемы.

Таблица 4.3.12 – Регистр управления трактом задержки SDCLK

31 – 10	9	8	7 – 0
–	EN	SDCI	SDCD
0x000000	0	0	0x00
r	rw	rw	rw

31 – 10 Зарезервировано

9 Разрешение использования (EN) – Если бит установлен, то активирована ветвь с управляемым трактом задержки SDCLK. В противном случае на блок задержки не подается тактирование (см. пункт «4.7.4»), сигнал CLK в обход упомянутого выше тракта подается на выход SDCLK

8 Инверсия тактового сигнала SDRAM (SDCI) – Если бит установлен, то тактовый сигнал будет инвертирован в тракте CLK-SDCLK

0 – 7 Задержка тактового сигнала SDRAM (SDCD) – Определяет количество элементов задержки в тракте CLK-SDCLK (см. пункт «4.7.4 Блок управления трактом задержки SDCLK»)

Регистр управления механизмом обхода чтения/записи проверочных бит

Регистр управления механизмом обхода чтения/записи проверочных бит внешней памяти (BYPASS_CTRL) вместе с регистром BYPASS_CHECKBITS предоставляет механизм считывания контрольных бит, сгенерированных контроллером внешней памяти (области PROM, SRAM / SDRAM), а так же возможность записи вместо них альтернативного значения. Данный функционал может быть использован для тестирования механизма коррекции ошибок и организации записи управляющих последовательностей во внешнюю память FLASH.

Стоит отметить, что при BYPASS_CTRL.CRB = 1 обращения считывания с разрядностью в 64 бита (с HWRITE = 0 и HSIZE = 3 на шине AHB) не изменяют содержимое регистра BYPASS_CHECKBITS. Подобное поведение реализовано для исключения влияния предвыборки инструкций из внешней памяти на состояние регистра BYPASS_CHECKBITS.

Также особенностью использования механизма считывания проверочных бит является сохранение в регистр BYPASS_CHECKBITS разного количества бит шины FT_CHECKBITS, в зависимости от используемого кода коррекции. Биты, которые не используются для определения синдрома ошибки, не сохраняются в регистр BYPASS_CHECKBITS. Это значит, что для области памяти, в которой для организации помехоустойчивого кодирования используется код Хэмминга(13,8), только биты FT_CHECKBITS[4:0] будут влиять на значение BYPASS_CHECKBIT. Для кода Хэмминга(39,32) будут использоваться только FT_CHECKBITS[6:0], для кода Хэмминга(72,64) – FT_CHECKBITS[7:0]. При использовании кода Рида-Соломона(96,64) вся шина FT_CHECKBITS[31:0] влияет на содержимое регистра BYPASS_CHECKBITS.

Рекомендуемые к использованию комбинации управляющих бит регистра BYPASS_CTRL приведены в таблице 4.3.14.

Таблица 4.3.13 – Регистр управления механизмом обхода чтения/записи проверочных бит внешней памяти

31 – 4	3	2	1	0
–	CWX	CRX	CWB	CRB
0	1	1	0	0
r	rw	rw	rw	rw

31 – 4 Зарезервировано

3 Разрешение механизма CHECKBITS_XOR при записи проверочных бит (CWX). Если бит установлен, то при выполнении операции записи на шину FT_CHECKBITS[31:0] будет поступать результат выполнения ИСКЛЮЧАЮЩЕГО ИЛИ над сгенерированными проверочными битами и значением регистра CHECKBITS_XOR

2 Разрешение механизма CHECKBITS_XOR при считывании проверочных бит (CRX). Если бит установлен, то при выполнении операции считывания в контроллер памяти будет поступать результат выполнения ИСКЛЮЧАЮЩЕГО ИЛИ над состояния шины FT_CHECKBITS[31:0] и регистра CHECKBITS_XOR

1 Обходной механизм записи проверочных бит (CWB). Если бит установлен, то на шину FT_CHECKBITS[31:0] подается значение регистра BYPASS_CHECKBITS

0 Обходной механизм считывания проверочных бит (CRB). Если бит установлен, то в регистр BYPASS_CHECKBITS сохраняется состояние шины FT_CHECKBITS[31:0]

Таблица 4.3.14 – Рекомендуемые комбинации управляющих битов

Описание	CWX	CRX	CWB	CRB
Наложить XOR маску проверочных бит при чтении и записи	1	1	0	0
Наложить XOR маску проверочных бит при записи	1	0	0	0
Наложить XOR маску проверочных бит при чтении	0	1	0	0
Запись значения регистра BYPASS_CHECKBITS вместо автоматически генерируемых проверочных бит	0	0	1	0
Сохранение состояния используемых корректирующим кодом бит шины FT_CHECKBITS в регистр BYPASS_CHECKBITS	0	0	0	1
Механизмы манипулирования шиной проверочных бит деактивированы	0	0	0	0

Регистр механизма обхода чтения/записи проверочных бит

Регистр механизма обхода чтения/записи проверочных бит внешней памяти (BYPASS_CHECKBITS) вместе с управляющими битами CWB и CRB регистра BYPASS_CTRL предоставляет механизм считывания контрольных бит, сгенерированных контроллером внешней памяти или OCRAM, а так же возможность записи вместо него альтернативного значения. Данный функционал может быть использован для тестирования механизма коррекции ошибок и организации записи управляющих последовательностей во внешнюю память FLASH.

Таблица 4.3.15 – Регистр механизма обхода чтения/записи проверочных бит

31	0
BYPASS_CHECKBIT	
0	
rw	

31 – 0 Поле проверочных бит обходного механизма считывания/записи (BYPASS_CHECKBIT) либо содержит в себе текущее значение используемых для вычисления синдрома ошибки разрядов шины FT_CHECKBITS, либо хранит значение, которое будет выставлено на шину шины FT_CHECKBITS[31:0] вместо автоматически сгенерированных проверочных бит, в зависимости от значения полей регистра BYPASS_CTRL

4.4 Процессорное ядро LEON4

4.4.1 Общие сведения

LEON4 – 32-битовое ядро процессора, соответствующее архитектуре IEEE-1754 (SPARC V8). Оно разработано для встраиваемых систем, сочетая в себе высокую производительность, низкую сложность и энергопотребление.

Блок-схема ядра LEON4 представлена на рисунке 4.4.1.

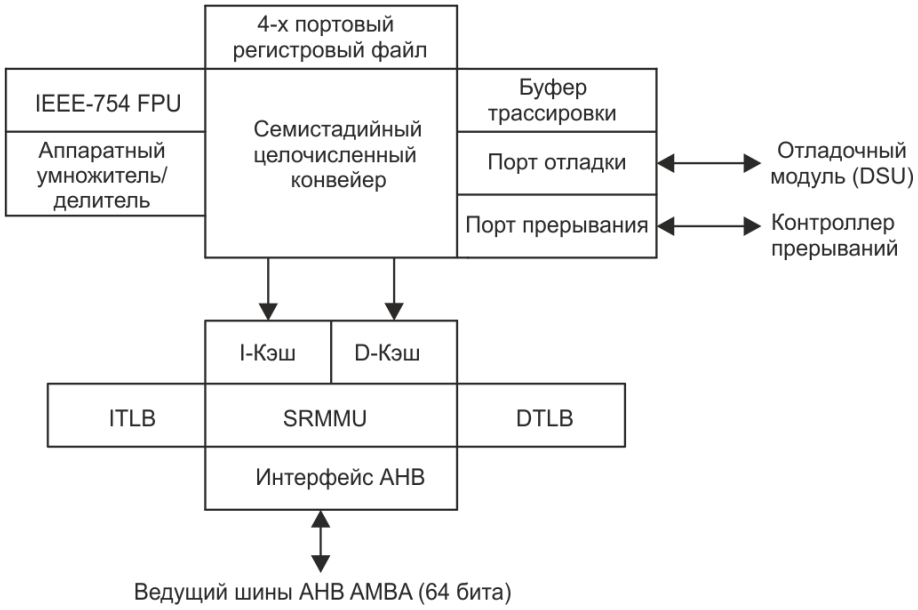


Рисунок 4.4.1 – Блок-схема ядра процессора LEON4

Данное ядро обладает следующими особенностями: 7-ступенчатый конвейер с Гарвардской архитектурой, отдельные кэш инструкций и данных, аппаратный умножитель и делитель, порт для подключения отладочного устройства.

Блок целочисленной арифметики

Блок целочисленной арифметики LEON4 реализует весь стандарт SPARC V8, включая инструкции аппаратного умножения и деления. В процессоре реализовано 8 регистровых окон, что полностью согласуется со стандартом SPARC (допустимый

диапазон от 2 до 32). Конвейер состоит из семи ступеней с отдельным интерфейсом кэша инструкций и кэша данных (Гарвардская архитектура).

Конфигурация кэша

Кэш процессора LEON4 построен по Гарвардской архитектуре (разделение памяти команд и данных). Реализовано 8 Кбайт кэша инструкций (два пути по 4 Кбайт) и 8 Кбайт кэша данных (два пути по 4 Кбайт).

Кэш инструкций содержит 32 байта на строку, в то время как кэш данных использует 16 байт на строку. Кэш данных содержит по одному биту достоверности на строку, использует метод сквозной записи и реализует буфер записи двойного слова. Кэш данных осуществляет слежение за шиной AHB (AHB snooping).

Используемый алгоритм вытеснения по умолчанию – LRU (Least Recently Used) – вытесняется запись, неиспользованная дольше всех.

Более подробная информация приведена в пункте «4.4.3 Подсистема кэша».

Блок арифметики с плавающей запятой

Блок целочисленной арифметики LEON4 обеспечивает интерфейс для высокопроизводительного модуля с плавающей запятой GRFPU. Блок арифметики с плавающей запятой работает параллельно с блоком целочисленной арифметики и не блокирует работу, если нет зависимости от данных или ресурса.

Блок управления памятью

Блок управления памятью SPARC V8 Memory Management Unit (SRMMU) реализует всю спецификацию MMU SPARC V8 и обеспечивает построения соответствия между 32-битовыми виртуальными адресными пространствами и физической памятью.

Для ускорения трансляции адреса виртуальной памяти в адрес физической памяти реализованы буферы ассоциативной трансляции (Translation Lookaside Buffer, TLB). Буфер инструкций и буфер данных содержат по 16 записей.

Встроенное отладочное устройство

Конвейер LEON4 поддерживает функциональную возможность отладки на целевых аппаратных платформах. Для обеспечения отладки программного обеспечения реализовано два регистра контрольных точек данных. Каждый регистр может вызвать системное прерывание точки останова при обращении к заданному диапазону инструкций или адресов данных. Если присоединено отладочное устройство, контрольные точки данных могут использоваться для входа в режим отладки. Через интерфейс отладочного устройства обеспечивается полный доступ ко всем регистрам процессора и кэшу. Интерфейсы отладки также обеспечивают пошаговое исполнение, трассировку инструкций и аппаратное управление точкой останова/точкой контроля данных. Внутренний буфер трассировки может контролировать и сохранять выполняемые инструкции, которые позднее считываются через интерфейс отладки.

Интерфейс прерываний

LEON4 поддерживает модель прерываний SPARC V8 с 15 асинхронными прерываниями. Интерфейс прерываний обеспечивает функциональную возможность генерации и подтверждения прерываний.

Интерфейс AMBA

Система кэширования реализует загрузку и сохранение данных в/из кэшей ведущим устройством AHB AMBA. Интерфейс совместим со стандартом AMBA-2.0. Чтобы оптимизировать количество обращений к памяти, во время пополнения строки генерируется пакетное обращение с инкрементом. Интерфейс AMBA сконфигурирован для использования 64-битовой шины при заполнении строки кэша. В процессоре также реализован порт подчиненного устройства шины AHB, осуществляющий слежение за обращениями, сделанными другими управляющими устройствами шины AHB.

Режим пониженного потребления энергии

Ядро процессора LEON4 обеспечивает возможность использования режима пониженного потребления энергии (Power-down), который останавливает конвейер и кэш до возникновения следующего прерывания. Это эффективный способ минимизировать потребление энергии, когда приложение неактивно, который не требует особых инструментов в виде управления тактированием.

4.4.2 Блок целочисленной арифметики LEON4

Общие сведения

Блок целочисленной арифметики LEON4 реализует целочисленную часть набора команд SPARC V8. Реализация сфокусирована на высокой производительности и низкой сложности. Блок целочисленной арифметики LEON4 обладает следующими особенностями:

- 7-ступенчатый конвейер инструкций;
- отдельный интерфейс кэша инструкций и кэша данных;
- 8 регистровых окон;
- аппаратный умножитель с дополнительным 16x16 – битовым MAC и 40-битовым аккумулятором;
- двоичный делитель (без восстановления);
- статическое предсказание ветвлений;
- возможность одновекторной обработки системных прерываний (SVT) для сокращения размера кода.

На рисунке 4.4.2 представлена блок-схема целочисленной арифметики.

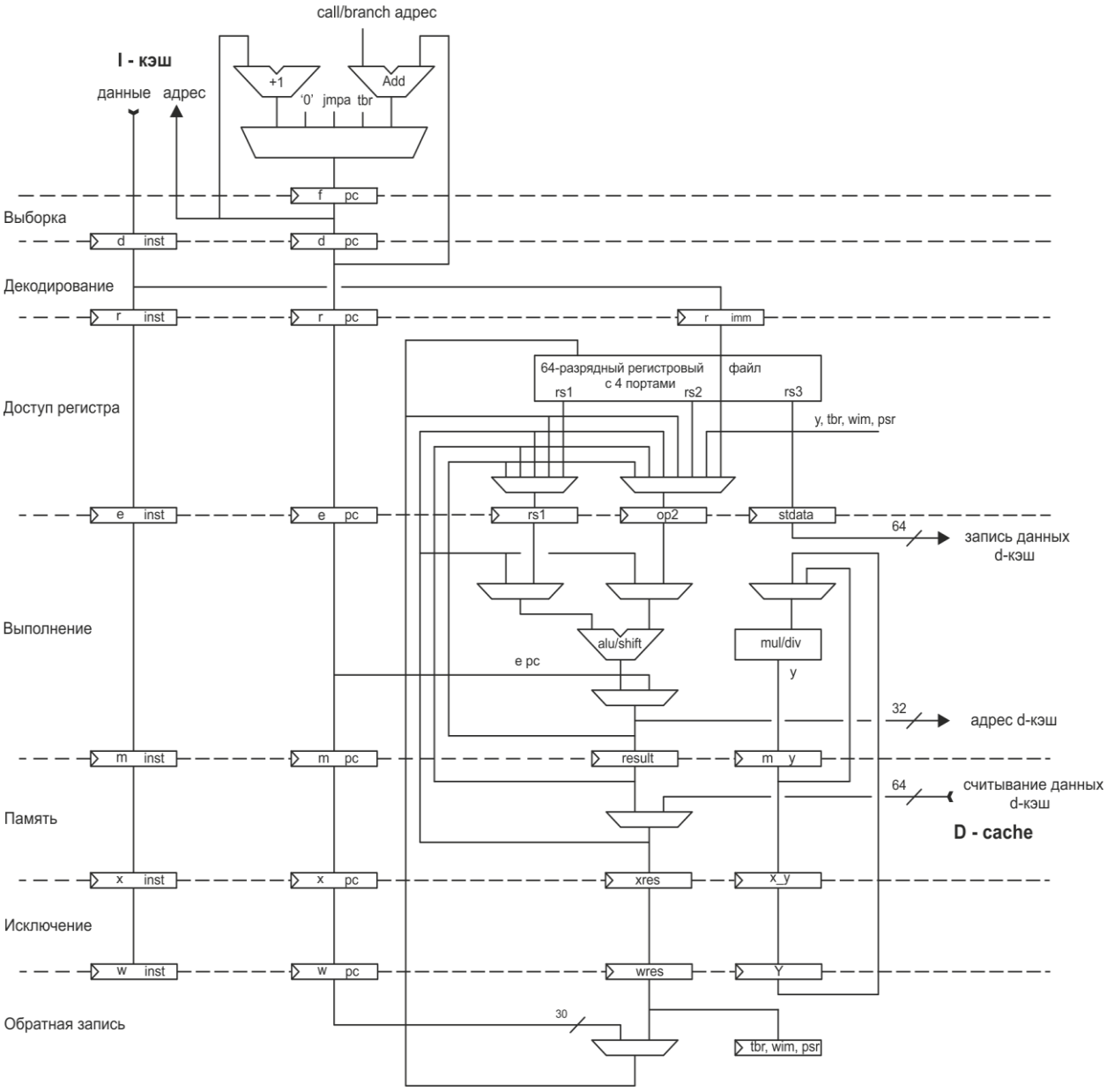


Рисунок 4.4.2 – Схема тракта блока целочисленной арифметики LEON4

Конвейер инструкций

Блок целочисленной арифметики LEON4 использует 7-ступенчатый конвейер:

1. FE (Выборка инструкции): если разрешен кэш инструкций, выборка инструкции происходит из кэша инструкций. В противном случае выборка передается в контроллер памяти. Инструкция действительна до конца этой ступени и защелкивается в блоке целочисленной арифметики.
2. DE (Декодирование): инструкция декодируется, и генерируются целевые адреса вызова процедуры/перехода.
3. RA (Доступ к регистру): компоненты считываются из регистрового файла или из внутренних цепей обходов данных.
4. EX (Выполнение): работает ALU, выполняются логические и сдвиговые операции. Генерация адреса для операций работы с памятью (например, LD) и для JMPL/RETT.
5. ME (Память): доступ к кэшу данных. Сохраняемые данные, считанные на этапе выполнения, записываются в кэш данных.

6. XC (Исключение): разбор исключительных ситуаций и прерываний. Для чтения кэша данные выравниваются по размеру.
7. WR (Запись): результаты ALU и операций кэша записываются в регистровый файл.

В таблице 4.4.1 приведено количество тактов на выполнение одной инструкции (с условием удачного обращения в кэш и отсутствия icc или блокировки загрузки).

Таблица 4.4.1 – Синхронизация инструкций

Инструкция	Циклы (MMU отключено)
JMPL, RETT	3
SMUL/UMUL	4*
SDIV/UDIV	35
Системное прерывание	5
Атомарные загрузка/сохранение	5
Все остальные инструкции	1
_____ * Цикл умножения – 4 периода тактирования для умножителя 16x16, 4 такта – задержка данных.	

Дополнительные условия, которые могут увеличить длительность нахождения инструкции в конвейере, перечислены в тексте ниже и таблице 4.4.2, которая описывает события, ведущие к возникновению этих типов блокировок и их временные характеристики.

Блокировка ветвления: Когда условное ветвление или системное прерывание выполняется через 1–2 цикла после инструкции, которая модифицирует коды условий, добавляется задержка в 1–2 цикла, чтобы позволить вычислить условия. Поскольку используется статическое предсказание ветвления, дополнительная задержка добавляется, только если ветвление не выполняется.

Задержка загрузки: Когда инструкции операций с данными идут сразу после загрузки данных, эти инструкции будут задерживаться для корректной работы конвейера. Конвейер процессора сконфигурирован на задержку в один цикл. Однотактная задержка дает более высокую производительность.

Задержка умножения: Для конвейерной реализации умножителя существует одноцикловая дополнительная задержка данных.

Такты задержки: Во время процесса кэш-промаха или при выполнении блокировки буфера хранения конвейер будет остановлен до готовности данных, растягивая время выполнения инструкции. Поскольку весь конвейер останавливается, такты задержки не будут отменять задержку загрузки или блокировку ветвления.

Задержки блока арифметики с плавающей запятой: Блоку арифметики с плавающей запятой может понадобиться дополнительно задержать конвейер или сделать более длительной специфическую инструкцию. Условия возникновения подобных ситуаций описаны в пункте «4.4.5».

Таблица 4.4.2 – Тактирование событий

Событие	Циклы
Обработка неудачного обращения в кэш инструкций, MMU отключен	3 + задержка памяти
Обработка неудачного обращения в кэш инструкций, MMU включен	5 + задержка памяти
Обработка неудачного обращения в кэш данных, MMU отключен (чтение), удачное обращение L2	3 + задержка памяти
Обработка неудачного обращения в кэш данных, MMU отключен (запись), буфер записи пуст	0
Обработка неудачного обращения в кэш данных, MMU включен (чтение)	5 + задержка памяти
Обработка неудачного обращения в кэш данных, MMU включен (запись), буфер записи пуст	0
Обход таблицы страниц MMU	10 + утроенная задержка памяти
Неудачное обращение прогнозирования переходов, переход сопровождается установкой ICC	2
Неудачное обращение прогнозирования переходов, между переходом и установкой ICC – одна инструкция	1
Перезапуск конвейера по причине изменения регистрового файла или ошибки кэша	7

Идентификационный номер SPARC разработчика

Cobham Gaisler присвоен идентификационный номер 15 (0xF) SPARC разработчика. Это значение аппаратно запрограммировано в битах 31–28 регистра **%psr.** Номер версии процессора LEON4 – 3 (для сохранения программной совместимости используется то же самое значение, что и у процессора LEON3), аппаратно запрограммирован в битах 27–24 регистра **%psr.**

Инструкции деления

Обеспечена полная поддержка инструкций деления SPARC V8 (SDIV, UDIV, SDIVCC & UDIVCC). Округление и отслеживание переполнения выполняется согласно стандарту SPARC V8.

Деление выполняется со знаком/без знака 64 бит на 32 бита. Реализован циклический итерационный алгоритм деления с основанием 2. Операция деления занимает 36 тактов. Делитель не оставляет остатка. Результат округляется до нуля. Обеспечивается обнаружение отрицательного результата, нулевого результата и переполнения (согласно методу обнаружения переполнения в руководстве архитектуры SPARC V8).

Инструкции умножения

Процессор LEON поддерживает инструкции умножения целых чисел SPARC UMUL, SMUL UMULCC и SMULCC. Эти инструкции выполняют 32×32-битовое умножение целых чисел, выдавая 64-битовый результат. SMUL и SMULCC выполняют умножение со знаком, в то время как UMUL и UMULCC выполняют умножение без знака. UMULCC и SMULCC также устанавливают коды условий для отражения результата. Инструкции умножения выполняются с использованием 16×16 аппаратного умножителя за четыре такта.

Инструкции умножителя с накоплением

Для ускорения алгоритмов DSP реализованы две инструкции умножителя с накоплением: UMAC и SMAC. UMAC выполняет 16-битовое умножение без знака, выдавая 32-битовый результат, и добавляет результат к 40-битовому аккумулятору, состоящему из 8 младших значащих битов из регистра %y и регистра %asr18. 32 младших значащих бита также записываются в регистр назначения. SMAC работает так же, но выполняет умножение и накопление со знаком. Инструкции MAC выполняются за один такт, но присутствуют два периода ожидания тактирования, т.е., если следующая инструкция использует регистр назначения MAC, как исходный компонент, вставляется один цикл останова конвейерной обработки.

Синтаксическая структура ассемблера:

umac rs1, reg_imm, rd
smac rs1, reg_imm, rd

Команда:

product[31:0] = rs1[15:0] × reg_imm[15:0]
result[39:0] = (Y[7:0] & %asr18[31:0]) + product[31:0]
(Y[7:0] & %asr18[31:0]) = result[39:0]
rd = result[31:0]

В регистр %asr18 можно осуществлять запись и считывать данные с помощью инструкций WRASR и RDASR.

Инструкция сравнения с обменом (CASA)

LEON4 реализует инструкцию альтернативного сравнения с обменом (CASA) SPARC V9. CASA работает в соответствии с руководством SPARC V9. Инструкция является привилегированной, за исключением случая использования ASI = 0xA (пользовательские данные).

Прогнозирование переходов

LEON4 реализует интеллектуальное выполнение постоянных переходов, потенциально сохраняя один – два периода тактирования, если поле %psr.icc обновлено в двух инструкциях, предшествующих условному переходу. При промахе прогнозирования не накладывается никаких дополнительных штрафов, так как инструкции выполняемой ветви могут быть получены из кэша.

Буфер трассировки инструкций

Буфер трассировки инструкций состоит из кольцевого буфера, в котором сохраняются выполненные инструкции. Работа буфера трассировки управляется через интерфейс отладочного устройства и не влияет на работу процессора. Полный объём – 1 Кбайт. В буфере трассировки сохраняется следующая информация:

- адрес инструкции и код операции;
- результат инструкции;
- загруженные/сохраненные данные и адрес;
- информация о системном прерывании;
- 30-битовая метка времени.

Исключения

LEON4 использует общую модель исключений SPARC. В таблице 4.4.3 представлены реализованные системные прерывания и их конкретный приоритет. Если бит ET регистра %psr сброшен, прерывание при возникновении исключительной ситуации останавливает работу процессора и вводит его в режим ошибки, при этом на выводе ERRORN# появляется активный уровень сигнала.

Более подробная информация о типах и механизмах работы исключений приведена в подразделе «5.4 Системные прерывания».

Таблица 4.4.3 – Размещение и приоритет системных прерываний

Прио-ритет	Системное прерывание	tt	Описание	Класс
1	2	3	4	5
1	reset	0x00	Сброс при включении	Прерывающее
2	data_store_error	0x2B	Ошибка буфера записи при сохранении данных	Прерывающее
3	instruction_access_error	0x01	Ошибка или отсутствие страницы MMU при выборке инструкции	Строгое
4	privileged_instruction	0x03	Выполнение привилегированной инструкции в пользовательском режиме	Строгое
5	illegal_instruction	0x02	UNIMP или другая нереализованная инструкция	Строгое
6	fp_disabled	0x04	Инструкция FP при отключенном блоке арифметики с плавающей запятой	Строгое
6	cp_disabled	0x24	Инструкция CP при отключенном сопроцессоре	Строгое
7	watchpoint_detected	0x0B	Обнаружение аппаратной точки останова	Строгое
8	window_overflow	0x05	SAVE в недопустимое окно	Строгое
8	window_underflow	0x06	RESTORE в недопустимое окно	Строгое
10	mem_address_not_aligned	0x07	Доступ к памяти по не выровненному адресу	Строгое
11	fp_exception	0x08	Исключение блока арифметики с плавающей запятой	Отложенное
11	cp_exception	0x28	Исключение сопроцессора	Отложенное
13	data_access_exception	0x09	Ошибка доступа во время выполнения инструкции загрузки или сохранения, отсутствие страницы MMU	Строгое
14	tag_overflow	0x0A	Переполнение тегированных арифметических операций	Строгое
15	divide_exception	0x2A	Деление на ноль	Строгое
16	trap_instruction	0x80 – 0xFF	Инструкция системного прерывания программного обеспечения (TA)	Строгое
17	interrupt_level_15	0x1F	Асинхронное прерывание 15	Прерывающее
18	interrupt_level_14	0x1E	Асинхронное прерывание 14	Прерывающее
19	interrupt_level_13	0x1D	Асинхронное прерывание 13	Прерывающее
20	interrupt_level_12	0x1C	Асинхронное прерывание 12	Прерывающее

Окончание таблицы 4.4.3

1	2	3	4	5
21	interrupt_level_11	0x1B	Асинхронное прерывание 11	Прерывающее
22	interrupt_level_10	0x1A	Асинхронное прерывание 10	Прерывающее
23	interrupt_level_9	0x19	Асинхронное прерывание 9	Прерывающее
24	interrupt_level_8	0x18	Асинхронное прерывание 8	Прерывающее
25	interrupt_level_7	0x17	Асинхронное прерывание 7	Прерывающее
26	interrupt_level_6	0x16	Асинхронное прерывание 6	Прерывающее
27	interrupt_level_5	0x15	Асинхронное прерывание 5	Прерывающее
28	interrupt_level_4	0x14	Асинхронное прерывание 4	Прерывающее
29	interrupt_level_3	0x13	Асинхронное прерывание 3	Прерывающее
30	interrupt_level_2	0x12	Асинхронное прерывание 2	Прерывающее
31	interrupt_level_1	0x11	Асинхронное прерывание 1	Прерывающее

Стандарт SPARC V8 гласит, что при возникновении системного прерывания будет выделено новое окно (значение поля CWP регистра %psr декрементируется по модулю NWINDOWS), в его локальные регистры %l1 и %l2 будут сохранены PC и nPC, в регистре %psr будет сброшен бит E, установлен бит S (его прежнее значение будет сохранено в PS).

Адрес, по которому будет осуществлен переход при возникновении системного прерывания (вектор прерывания), задается значением регистра %tbr. Если не активирована одновекторная обработка прерываний (бит SVT регистра %asr17 сброшен), то вектор прерывания полностью совпадает со значением регистра %tbr. Например, если текущая используемая таблица прерываний располагается в начале области RAM (0x40000000) и выставлено соответствующее значение поля TBA регистра %tbr, то при возникновении системного прерывания interrupt_level_2 с tt = 0x12 (внешнее прерывание по линии GPIO[2] или прерывание от интерфейса UART 1, согласно таблице 4.8.1) процессор осуществит переход на адрес 0x40000120.

На каждую запись в таблице прерываний отведено по четыре инструкции. Этого объема хватает для организации перехода к обработчику прерывания или переводу процессора в состояние ошибки (через вызов системного прерывания при сброшенном бите ET регистра %psr), если это требуется. Ниже, как пример, приведены: вектор, который переведет процессор в состояние ошибки (trap_cp_exception); вектор с вызовом кода обработчика прерывания (trap_interrupt_level_2); заготовка под пользовательский обработчик внешнего системного прерывания (interrupt_2_trap_handler).

```
trap_interrupt_level_2:
        0x40000120   sethi %hi(interrupt_2_trap_handler), %l3
        0x40000124   jmp [%l3 + %lo(interrupt_2_trap_handler)]
        0x40000128    rd %psr, %l0
        0x4000012C   nop
        …
trap_cp_exception:
        0x40000280   ta 0
        0x40000284   nop
        0x40000288   nop
        0x4000028C   nop
        …
        .global interrupt_2_trap_handler
interrupt_2_trap_handler:    ! Данный код размещается за пределами таблицы прерываний.
        or %l0, 0xF20, %l4   ! Игнорировать внешние прерывания (кроме 15-й линии) и
        wr %l4, %g0, %psr    ! установить бит ET регистра %psr.
        …                    ! «Тело» самого обработчика.
```

wr %l0, %g0, %psr ! Возврат прежнего значения %psr (устанавливать бит ET не обязательно, это произойдет автоматически при выполнении rett).

nop; **nop**; **nop** ! Может быть заменено на 3 инструкции, не изменяющие %psr.

jmp %l1 ! %l1 – старое значение PC

rett %l2 ! %l2 – старое значение nPC

Возврат к прерванной инструкции может быть заменен на возврат к следующей после прерванной инструкции (если обработчик прерывания эмулирует выполнение инструкции вызвавшей прерывание), как это приведено в подпункте «Инструкция возврата из системного прерывания».

Если планируется использовать обработку системных прерываний (установлен бит ET регистра %psr), то значение %wim следует выставлять с учетом резервирования одного окна регистров под вызов системного прерывания. Также должны быть организованы обработчики системных прерываний window_overflow и window_underflow. В противном случае при возникновении системного прерывания могут быть "испорчены" данные в актуальном окне регистров, расположенном на противоположном конце цепочки вложенных вызовов функций.

Одновекторная обработка системных прерываний (SVT)

Одновекторная обработка системных прерываний (SVT) – опция SPARC V8e, которая служит для сокращения размера кода во встраиваемых системах. При включённой данной опции системное прерывание всегда переходит по вектору сброса (%tbr.TBA + 0). Тип системного прерывания будет указан в %tbr.TT и должен быть декодирован общим обработчиком системных прерываний. SVT разрешено при установке бита 13 в %asr17.

Частичная запись WRPSR

Частичная запись в регистр %psr (WRPSR) – опция SPARC V8e, которая позволяет инструкции WRPSR влиять только на поле %psr.ET. Если поле rd инструкции WRPSR не нулевое, то будет перезаписано только поле ET.

Режим пониженного потребления энергии

Процессор поддерживает использование режима пониженного потребления энергии, чтобы минимизировать потребляемую мощность в режиме простоя. Вход в режим пониженного потребления энергии осуществляется с помощью инструкции WRASR над регистром %asr19:

wr %g0, %asr19

В режиме пониженного потребления энергии работа конвейера приостанавливается до следующего прерывания. Сигналы в конвейере процессора и кэшах статичны, сокращается потребляемая мощность на динамическое переключение. В тоже время, в текущей реализации процессора тактовый сигнал в режиме PowerDown продолжает поступать на все его компоненты. Вследствие такой реализации разница между током потребления процессора в режиме PowerDown и в активном режиме при средней загруженности конвейера и кэш составляет порядка 5%.

Важно помнить:

1. В регистр %asr19 всегда должно быть записано нулевое значение для гарантии совместимости с будущими версиями процессора.
2. Данная инструкция должна выполняться из режима операционной системы (supervisor mode), прерывания должны быть разрешены.

При выходе из режима Power-down конвейер будет вновь заполняться инструкциями и первая инструкция, следующая за WRASR, будет выполнена перед захватом прерывания. Будет осуществлена выборка вплоть до шести инструкций после WRASR (возможно с промахами кэша, если они не находятся в нем) перед началом выборки обработчика прерывания.

Сброс процессора

Сброс процессора осуществляется при установке входа RESET, по крайней мере, в течение четырёх циклов тактирования. В таблице 4.4.4 указаны значения сброса регистров, которые меняются при сбросе. Во всех других регистрах значения не меняются (или не определяются).

Таблица 4.4.4 – Значения при сбросе процессора

Регистр	Значение сброса
PC (счетчик команд)	0x00000000
nPC (следующее значение счетчика команд)	0x00000004
PSR (регистр состояния процессора)	ET = 0, S = 1

Выполнение инструкций процессором стартует с адреса 0x00000000.

4.4.3 Подсистема кэша

Процессор LEON4 выполнен на базе Гарвардской архитектуры с отдельными шинами инструкций и данных, подключенными к двум независимыми контроллерам кэша. До тех пор, пока выполнение не вызывает промаха в кэш, контроллеры кэша позволяют осуществлять выборку инструкции и выполнять одно обращение считывания/записи данных каждый такт, позволяя конвейеру работать на полной скорости.

При промахе в кэш контроллер кэша устанавливает сигнал удержания, приостанавливающий конвейер целочисленных инструкций, после получения данных сигнал удержания сбрасывается, разрешая дальнейшее выполнение инструкций. Оба контроллера кэшей для доступа по шине совместно используют одно и то же подключение к шине AHB. Некоторые части MMU (логические цепи обхода таблицы страниц, буфер TLB) также совместно используются обоими контроллерами.

Еще одним важным компонентом кэша данных является буфер записи, позволяющий выполнять операции сохранения параллельно с выполнением инструкций.

Оба кэша сконфигурированы как мультиканальные, с ассоциативностью 2 (то есть имеют по два канала). Размер каждого канала – 4 Кбайт (и для инструкций и для данных). Таким образом, общий размер кэша составляет 8 Кбайт инструкций и 8 Кбайт данных. Кэш инструкций содержит 32 байта на строку (линию), кэш данных – 16 байт на строку. Кэш данных выполнен как буфер записи двойного слова. С помощью регистров контроля кэша можно выбрать один из трёх способов замены: с наиболее давним использованием (LRU), с наиболее давней произведенной заменой (LRR) или псевдослучайный. По сбросу выбран алгоритм LRU.

Алгоритм LRU для сохранения предыстории доступа использует дополнительные триггеры на строку кэша. Алгоритм LRR, наряду с тегами, хранит индекс наиболее давно замененного. Эти данные используются для реализации политики замены. Алгоритм псевдослучайной замены реализован через счетчик по модулю N, который выбирает строку для исключения при неудачном обращении в кэш.

Управление областями кэширования для обоих кэшей осуществляется через адресную информацию plug & play AHB. Отображение памяти для каждого ведомого

устройства AHB указывает возможность кэширования области; эта информация используется для (статического) определения кэшируемого доступа. Данный подход означает, что разметка кэширования полностью определяется текущей конфигурацией шины AHB.

Принцип работы кэша

Каждый контроллер кэша имеет два основных блока памяти: блок меток и блок данных. Каждый адрес в памяти меток, каждой строки кэша, каждого канала используется для сохранения данных из определенного набора возможных адресов. Память данных хранит данные соответствующего канала (пути).

Для каждого канала память меток содержит следующую информацию:

- Бит достоверности, который сообщает, содержатся ли достоверные данные или место свободно. Оба кэша имеют по одному биту достоверности на строку.
- Метку, все биты адреса кэшируемой памяти, которые не подразумеваются набором.
- Если MMU используется, идентификатор контекста записи кэша.
- Если используется LRR, бит, определяющий порядок замещения.
- Контрольные биты для детектирования ошибок.

Когда осуществляется считывание из кэша, метки и данные для всех каналов соответствующих наборов считываются параллельно, метки и биты достоверности сравниваются с требуемым адресом, выбирается соответствующий канал (путь). В случае «попадания», все эти действия осуществляются в один такт, что обеспечивает полную скорость выполнения процессора.

В случае «промаха», кэш сначала выдаст некорректные данные. Однако на следующий такт будет установлен сигнал удержания, который воспрепятствует использованию этих данных процессором. После того как промах будет обработан, корректные данные вводятся в конвейер, после этого сигнал удержания будет снят. Если отсутствующий адрес попадает в кэшируемую область, тогда данные, считанные при промахе кэша, будут помещены в него, возможно с вытеснением строки одного из существующих каналов (путей).

В случае потокового выполнения инструкций, конвейер процессора осуществляет по одному шагу при получении очередной инструкции. Если процессору требуются дополнительные такты для выполнения многотактных инструкций, или при блокировании (см. пункт «4.4.2 Блок целочисленной арифметики LEON4. Конвейер инструкций»), или при осуществлении прыжка/выполнения операции ветвления, контроллер кэша инструкций приостановит конвейер, произведет выборку для заполнения оставшейся строки кэша, затем конвейер возобновит работу в нормальном режиме.

В случае ошибки доступа к памяти при выборке строки соответствующий бит достоверности в метке кэша сбрасывается и генерируется системное прерывание при ошибке доступа к инструкции (tt=0x01) или данным (tt=0x09).

Заблокированные передачи AHB генерируются для инструкций LDSTUB, SWAP и CASA. Заблокированные передачи всегда некэшируемые.

Отображение адреса в кэш инструкций и кэш данных

Видимый процессору адрес разбивается на части (метку, индекс и смещение) в зависимости от параметров кэша. Индекс используется для выбора строки из кэша, поэтому только ограниченное количество строк с одинаковым индексом может одновременно храниться в кэше (по количеству путей). Метка адреса, хранимая в кэше, используется для сравнения со старшими разрядами запрошенных данных. В таблице 4.4.5 представлено отображение адреса в кэш инструкций, в таблице 4.4.6 – в кэш данных:

Таблица 4.4.5 – Отображение адреса в кэш инструкций (4 Кбайт/путь, 32 байта/строка)

31 12	11 5	4 0
Tag	Index	Offset

31 – 12 Метка адреса (Tag) – старшие значащие биты адреса. К каждой строке данных в кэше сохраняется одна метка адреса
11 – 5 Индекс (Index) – порядковый номер строки кэша в пределах пути
4 – 0 Смещение (Offset) – смещение данных в пределах строки кэша

Таблица 4.4.6 – Отображение адреса в кэш данных (4 Кбайт/путь, 16 байт/строка)

31 12	11 4	3 0
Tag	Index	Offset

31 – 12 Метка адреса (Tag) – старшие значащие биты адреса. К каждой строке данных в кэше сохраняется одна метка адреса
11 – 4 Индекс (Index) – порядковый номер строки кэша в пределах пути
3 – 0 Смещение (Offset) – смещение данных в пределах строки кэша

Правила поведения кэша данных

Кэш данных использует метод сквозной записи, это означает, что каждая операция сохранения по шине будет осуществляться через буфер записи. Поэтому в кэше не будет наблюдаться «грязных» строк, в которых ещё не перезаписаны данные, кроме тех, что находятся в буфере. Операция сохранения будет также обновлять кэш, если адрес присутствует, однако в этом случае не будет выделена новая строка. Правила поведения кэша данных представлены в таблице 4.4.7.

Таблица 4.4.7 – Правила поведения кэша данных

Операция	В кэше	Кэши-руемость	Действие на шине	Действие в кэше	Загрузка данных
Загрузка данных	Нет	Нет	Чтение	Отсутствует	Шина
	Нет	Да	Чтение	Выделение / обновление строки	Шина
	Да	–	Нет	Отсутствует	Кэш
Загрузка данных с принудительным кэш-промахом (ASI 0x01)	Нет	Нет	Чтение	Отсутствует	Шина
	Нет	Да	Чтение	Выделение / обновление строки	Шина
	Да	–	Чтение	Обновление данных	Шина
Загрузка данных в обход MMU (ASI 0x1C)	–	–	Чтение (физический адрес)	Отсутствует	Шина
Сохранение данных	Нет	Нет	Запись (через буфер)	Отсутствует	Не доступно
	Нет	Да	Запись (через буфер)	Отсутствует	Не доступно
	Да	–	Запись (через буфер)	Обновление данных	Не доступно
Сохранение данных в обход MMU (ASI 0x1C)	–	–	Запись (через буфер, физический адрес)	Отсутствует	Не доступно

Буфер записи

Буфер записи (WRB) состоит из трех 32-битовых регистров, используемых для временного хранения данных до их отправки в устройство назначения. Для хранения полуслова или байтов сохраненные данные дублируются с соответствующим выравниванием байтов для записи в адресуемое с помощью слова устройство перед загрузкой в один из регистров WRB. WRB освобождается до последовательности заполнения кэша при неудачном обращении загрузки во избежание считывания из кэша данных устаревших данных.

Так как процессор работает параллельно с буфером записи, ошибка при записи не вызывает исключение инструкции сохранения. В зависимости от работы памяти и кэша, цикл записи не начнется, пока не завершится несколько циклов тактирования после выполнения инструкций сохранения. При возникновении ошибки при записи текущая инструкция принимает системное прерывание 0x2b.

Примечание – Обработчик системного прерывания 0x2b должен сбрасывать кэш данных, поскольку удачное обращение записи обновляет кэш, в то время как в памяти будет сохраняться старое значение как следствие ошибки при записи.

Работа с активным MMU

Когда разрешено использование MMU, виртуальный адрес, видимый из исполняемого кода, больше не соотносится напрямую с физическим адресом на шине AHB. Кэш использует метки, основанные на виртуальном адресе, поскольку это позволяет избежать какой-либо дополнительной работы на трансляцию в случае наиболее критичных по времени исполнения обращений.

Тем не менее, каждый раз, когда необходим доступ к шине, требуется отправить к MMU запрос на трансляцию, чтобы преобразовать виртуальный адрес в физический адрес. Для буфера записи данное действие включено в состав фоновой обработки операции сохранения. Запрос на трансляцию к MMU может привести к доступу памяти из MMU для выполнения обхода таблицы страниц, в зависимости от состояния MMU.

Идентификатор контекста MMU включается в метку кэша для того, чтобы обеспечить переключение между несколькими контекстами MMU, отображающими один и тот же виртуальный адрес для различных физических адресов. Следует обратить внимание, что кэш не обнаруживает альтернативные ссылки на тот же физический адрес, в этом случае один и тот же физический адрес может быть кэширован несколькими каналами (см. далее «Отслеживание (Snooping)»).

Отслеживание (Snooping)

Для кэша данных может быть активирован режим слежения за шиной AHB (с помощью установки бита DS регистра управления кэша). В таком случае на шине AHB, к которой подключено ядро процессора, ведется отслеживание операций записи от других управляющих устройств по адресам, которые содержатся в кэше. Если запись производится в кэшированный адрес, строка кэша помечается как недействительная, процессор будет вынужден осуществлять выборку данных из памяти при следующем запросе на их считывание.

Для использования отслеживания совместно с MMU задействована дополнительная память меток, в которой сохраняются физические метки. Они позволяют производить сравнение содержимого кэша с памятью, расположенное по реальным физическим адресам на шине AHB.

Ядро процессора не следит само за собой, поэтому если используется многократная виртуальная разметка одних и тех же физических адресов, это приведет к появлению устаревших данных в кэше при записи.

Разрешение и запрещения использования кэшей

После сброса процессора использование кэша инструкций и кэша данных запрещено. Их работа может быть активирована с помощью записи в подпункте «Регистр управления кэша» (см. пункт «4.4.8»). После сброса процессора, прежде чем разрешить использование кэшей, их следует сбросить, чтобы гарантировать, что во всех метках будет сброшен бит действительности. Также, из-за особенностей реализации системы помехоустойчивого кодирования, требуется осуществить сброс данных кэшей для всех каналов. Игнорирование данной особенности приведет к постоянным промахам в кэш, что сделает его использование полностью бесполезным с точки зрения производительности. Подходящая ассемблерная последовательность для инициализации и разрешения кэшей после сброса процессора:

! Примечание 1: Инструкции STA / LDA могут быть выполнены, только если процессор находится в режиме супервизора (установлен бит S регистра %psr)

! Примечание 2: Для осуществления доступа к меткам и данным кэша инструкций (ASI 0x0C-0x0D) требуется, чтобы он находился в состоянии «отключен»

! Примечание 3: Инструкции STA / LDA в ASI 0x0C-0x0F во время выполнения flush приведут к системному прерыванию data_access_exception

```
   .text
   .global initialize_caches
initialize_caches:
initialize_caches_tags:
   sta %g0, [%g0] 0x02        ! Отключение обоих кэшей (опционально). Можно
                              ! исключить данную инструкцию, если кэш инструкций
                              ! не был включен ранее
   flush %g0                  ! Сбросит все поля VALID обоих кэшей. Операция
                              ! не останавливает выполнение инструкций, но
                              ! во время её выполнения нельзя осуществлять доступ
                              ! к ASI 0x0C–0x0F
initialize_icache_data_except_way_0:
   set 0x00000008, %g3
   lda [%g3] 0x02, %g3        ! Чтение регистра конфигурации кэша инструкций
   srl %g3, 24, %g1
   and %g1, 0x7, %g1          ! Значение поля WAYS
                              ! (000₂ – кэш с прямым отображением,
                              ! 001₂..011₂ – 2-,3-, 4-канальный кэш соответственно
   add %g1, 1, %g1
   sll %g1, 10, %g2           ! Размер кэша инструкций, при условии, что
                              ! каждый канал – 1 Кбайт
   srl %g3, 20, %g1
   and %g1, 0xF, %g1          ! Значение поля WSIZE (2^WSIZE Кбайт/канал)
   sll %g2, %g1, %g2          ! %g2 – размер кэша инструкций

   ! Ожидание завершения операции flush
   lda [%g0] 0x02, %g1
2:
   srl %g1, 14, %g1
   andcc %g1, 0x0003, %g0  ! Проверка битов IP и DP регистра управления кэша
   bne 2b
    lda [%g0] 0x02, %g1
```

```
   ! Основной цикл записи (весь объем заполняется 0x00000000)
   .align 32                ! Принудительное размещение кода цикла в пределах
                            ! одного блока предвыборки инструкций (8 слов)
1:
   subcc %g2, 4, %g2
   bne 1b
    sta %g0, [%g2] 0x0D     ! Запись в ASI 0x0D (данные кэша инструкций)

   ! Разрешение кэша инструкций (для ускорения хода дальнейшей инициализации)
   set 0x00000003, %g1
   sta %g1, [%g0] 0x02

initialize_dcache_data_except_way_0:
   set 0x0000000C, %g3
   lda [%g3] 0x02, %g3      ! Чтение регистра конфигурации кэша данных
   srl %g3, 24, %g1
   and %g1, 0x7, %g1        ! Значение поля WAYS
                            ! (000₂ – кэш с прямым отображением,
                            ! 001₂..011₂ – 2-,3-, 4-канальный кэш соответственно
   add %g1, 1, %g1
   sll %g1, 10, %g2         ! Размер кэша данных, при условии, что
                            ! каждый канал – 1 Кбайт
   srl %g3, 20, %g1
   and %g1, 0xF, %g1        ! Значение поля WSIZE (2^WSIZE Кбайт/канал)
   sll %g2, %g1, %g2        ! %g2 – размер кэша данных

   ! Основной цикл записи (весь объем заполняется 0x00000000)
1:
   subcc %g2, 4, %g2
   bne 1b
    sta %g0, [%g2] 0x0F     ! Запись в ASI 0x0F (данные кэша данных)

   ! Разрешение обоих кэшей и слежения за метками кэша данных
   set 0x0080000F, %g2
   sta %g2, [%g0] 0x02
```

Фиксация кэша

Каждый кэш может находиться в одном из трех состояний: запрещен, разрешен, зафиксирован. Если он запрещен, то не осуществляется никаких операций с кэшем, запросы на считывание и запись отправляются напрямую контроллеру памяти. Если кэш разрешен, то операции осуществляются в соответствии с приведенным выше описанием. Если кэш зафиксирован, то при обращении остается доступен, продолжает осуществляться синхронизация данных с основной памятью, как если бы он был разрешен, но не выделяются новые строки при промахе в кэш.

Если установлены биты DF или IF, то соответствующий тип кэша будет зафиксирован при захвате асинхронного прерывания. Это может быть полезным в системах реального времени для получения более точного результата при расчете времени исполнения сегмента кода в наихудших условиях. Выполнение обработчика прерывания не будет вызывать вытеснение каких-либо строк кэша, когда управление будет возвращено прерванной задаче, состояние кэша будет оставаться идентичным тому, которое было до прерывания. Если кэш был зафиксирован по прерыванию, то единственный способ вновь его активировать – запись в пункте «4.4.8» подпункта

«Регистр управления кэша». Обычно данное действие осуществляется в конце обработчика прерывания, перед тем как управление будет возвращено прерванной задаче.

Сброс кэша (Flushing)

Сброс кэша инструкций и кэша данных осуществляется с помощью инструкции FLUSH. Кэш инструкций также сбрасывается при установке бита FI в регистре управления кэша (см. таблицу 4.4.29), в то время как кэш данных также сбрасывается при установке бита FD в регистр управления кэша. Т.к. процессор реализован с MMU, кэши I и D можно сбрасывать при записи в любую ячейку с ASI = 0x10.

Сброс кэша занимает один цикл на строку кэша, данная операция не приостанавливает работу блока целочисленной арифметики, но в это время кэши будут отключены. По завершении сброса кэш восстанавливает состояние (отключен, включен или зафиксирован), отображаемое в регистре управления кэша. Диагностический доступ к кэшу во время операции FLUSH закрыт, в случае попытки получения доступа выполняется системное прерывание data_access_exception (tt = 0x09).

Стоит отметить, что в противопоставление спецификации SPARC V8, в которой заявлено что сброс будет воздействовать только на инструкции, на которые указывает аргумент инструкции FLUSH, LEON4 будет дополнительно сбрасывать весь кэш инструкций и кэш данных. Такое поведение разрешено руководством, так как дополнительный функционал сброса повлияет только на производительность, а не на функционирование. На текущий момент LEON4 игнорирует аргумент адреса, для будущей совместимости рекомендуется использовать только *flush %g0*, если необходим полнофункциональный сброс.

Доступ для диагностического контроля кэша

Доступ к меткам и данным в кэше инструкций и кэше данных можно получить через адресуемое пространство ASI 0xC, 0xD, 0xE и 0xF при выполнении инструкций STA и LDA. Более подробная информация приведена в пункте «4.4.7 ASI 0xC–0xF. I-кэш метки/данные, D-кэш метки/данные».

4.4.4 Блок управления памятью (MMU)

Блок управления памятью (MMU) совместим с базовым блоком SPARC V8 (SRMMU), описанным в приложении H спецификации SPARC V8. Блок управления памятью можно дополнительно сконфигурировать. Для этого используется доступ к регистрам, расположенным в ASI. Более подробная информация приведена в пункте «4.4.7 Идентификаторы адресных пространств (ASI)».

Принцип работы MMU/кэша

Если MMU отключен, кэш работает в нормальном режиме с отображением физического адреса. Если MMU разрешен, метки кэша сохраняют виртуальный адрес, а также используется 8-битовое поле контекста.

Поскольку для создания меток кэша используется виртуальный адрес, в случае заполнения кэша или удачного обращения в кэш дополнительные циклы тактирования не требуются. Поскольку реализован раздельный буфер ассоциативной трансляции (TLB) с быстрым доступом, то даже при неудачном обращении в кэш или заполнении буфера на необходимую трансляцию не требуется дополнительных тактов процессора. Просмотр TLB производится одновременно с доступом к метке. Это улучшает общую пропускную способность.

В случае неудачного обращения TLB, требуется совершить обход таблицы переадресации страниц, что может вызвать вплоть до четырех доступов чтения на шине AMBA и одну возможную операцию обратной записи.

Обращение в отсутствующую страницу MMU будет генерировать системное прерывание 0x09 для кэша данных и системное прерывание 0x01 для кэша инструкций, изменит регистр статуса сбоя MMU (см. таблицу 4.4.36) в соответствии с таблицей 4.4.8 и спецификацией SRMMU. В случае множественных ошибок итоговое значение типа ошибки выбирается согласно приоритету, требуемому спецификацией SRMMU. Кэш и память не будут изменены при возникновении страничной ошибки MMU.

Таблица 4.4.8 – Значения типов ошибок регистра статуса сбоя MMU LEON4

Приоритет	SPARC V8 описание	Тип	Условия
1	Внутренняя ошибка	6	Никогда не случается в LEON SRMMU
2	Ошибка трансляции	4	Ответ ошибки AHB во время обхода таблицы страниц. Ошибки трансляции, как определено в описании архитектуры SPARC V8. Ошибка трансляции по причине ответа AMBA ERROR будет переписывать все остальные ошибки. Другие ошибки трансляции не могут переписать существующие ошибки трансляции, когда FAV = 1
3	Некорректный адрес	1	Запись таблицы страниц для адреса была помечена как недействительная
4	Ошибка права доступа	3	Доступ запрещен, основываясь на таблице страниц и статусе пользователь /супервизор (см. спецификацию SRMMU для разъяснения приоритета ошибок защиты и права доступа)
5	Ошибка защиты	2	
–	Нет	0	Нет ошибок (внутри прерывания это означает, что системное прерывание случилось, когда осуществлялась выборка актуальных данных)

Регистры MMU

Регистры MMU приведены в таблицах 4.4.33 – 4.4.37 в пункте «4.4.8 Регистры конфигурации».

Буфер ассоциативной трансляции (TLB)

MMU сконфигурировано для использования отдельного TLB для инструкций и данных. Количество записей TLB инструкций равно 16. Количество записей TLB данных – 16. Организация TLB и количество записей не видны программному обеспечению, поэтому модификаций операционной системы под различные конфигурации не требуется. Буфер ассоциативной трансляции может быть сброшен записью в ASI 0x18 (см. «4.4.7 Идентификаторы адресных пространств (ASI)»).

4.4.5 Блок арифметики с плавающей запятой

Высокопроизводительный блок GRFPU реализует операции над числами с плавающей запятой, согласно стандартам IEEE-754 и SPARC V8 (IEEE-1754).

GRFPU обрабатывает числа с плавающей запятой в формате с одинарной или удвоенной точностью, как определено в стандарте IEEE-754, за исключением денормализованных чисел.

Блок арифметики с плавающей запятой связан с помощью интерфейса с конвейером LEON4, используя специализированный контроллер FPU LEON4 (GRFPC), который позволяет выполнять инструкции FPU одновременно с инструкциями целочисленной арифметики. Целочисленный конвейер останавливается только в случае зависимости по данным или ресурсам.

GRFPU поддерживает четыре типа операций с плавающей запятой: арифметические, операции преобразования типов данных, операции сравнения и перемещения. Операции реализуют все инструкции с плавающей запятой, определенные набором команд SPARC V8, и большинство операций, определенных в IEEE-754. Все операции с кодами, компонентами, результатами и кодами исключений представлены в таблице 4.4.9.

Таблица 4.4.9 – Операции GRFPU

Операция	Код операции [8:0]	Op1	Op2	Результат	Исключения	Описание
1	2	3	4	5	6	7
Арифметические операции						
FADDS	001000001	SP	SP	SP	UNF, NV, OF, UF, NX	Сложение
FADDD	001000010	DP	DP	DP		
FSUBS	001000101	SP	SP	SP	UNF, NV, OF, UF, NX	Вычитание
FSUBD	001000110	DP	DP	DP		
FMULS	001001001	SP	SP	SP	UNF, NV, OF, UF, NX	Умножение, FSMULD дает число удвоенной точности при умножении двух операндов одинарной точности
FMULD	001001010	DP	DP	DP		
FSMULD	001101001	SP	SP	DP	UNF, NV, OF, UF	
FDIVS	001001101	SP	SP	SP	UNF, NV, OF, UF, NX, DZ	Деление
FDIVD	001001110	DP	DP	DP		
FSQRTS	000101001	–	SP	SP	UNF, NV, NX	Квадратный корень
FSQRTD	000101010	–	DP	DP		
Операции преобразования типов данных						
FITOS	011000100	–	INT	SP	NX	Преобразование целых в типы с плавающей запятой
FITOD	011001000	–		DP	–	
FSTOI	011010001	–	SP	INT	UNF, NV, NX	Преобразование типов с плавающей запятой в целые. Результат округляется в направлении нуля
FDTOI	011010010	–	DP			
FSTOI_RND	111010001	–	SP	INT	UNF, NV, NX	Преобразование типов с плавающей запятой в целые. Округление в соответствии с входом ROUND
FDTOI_RND	111010010	–	DP			
FSTOD	011001001	–	SP	DP	UNF, NV	Преобразование форматов с плавающей запятой из одного в другой
FDTOS	011000110	–	DP	SP	UNF, NV, OF, UF, NX	

Окончание таблицы 4.4.9

1	2	3	4	5	6	7
Операции сравнения						
FCMPS	001010001	SP	SP	CC	NV	Сравнение форматов с плавающей запятой. Неверное исключение генерируется, если один из компонентов выдает сигнал NaN
FCMPD	001010010	DP	DP			
FCMPES	001010101	SP	SP	CC	NV	Сравнение форматов с плавающей запятой. Неверное исключение генерируется, если один из компонентов – NaN (без сообщений или выдает сигнал)
FCMPED	001010110	DP	DP			
Отрицание, абсолютное значение и перемещение						
FABSS	000001001	–	SP	SP	–	Абсолютное значение
FNEGS	000000101	–	SP	SP	–	Отрицание
FMOVS	000000001	–	SP	SP	–	Перемещение. Копирует операнд в регистр результата

Примечание – Принятые условные обозначения:
- SP – число с плавающей запятой с одинарной точностью;
- DP – число с плавающей запятой с удвоенной точностью;
- CC – коды условий (00 – равенство; 01 – первый операнд больше второго; 10 – первый операнд меньше второго; 11 – не упорядочено);
- UNF, NV, OF, UF, NX – исключения с плавающей запятой;
- INT – 32-разрядное целое число.

Арифметические операции включают сложение, вычитание, умножение, деление и квадратный корень. Каждая арифметическая операция может быть выполнена в формате с одинарной или удвоенной точностью.

GRFPU конвейеризирован и обеспечивает старт одной инструкции при каждом цикле тактирования, за исключением FDIV и FSQRT, которые выполняются только поочередно. FDIV и FSQRT выполняются в отдельном делителе и не блокируют FPU при выполнении других параллельных операций.

Все инструкции, кроме FDIV и FSQRT, имеют период ожидания в три цикла, но для улучшения синхронизации контроллер FPU LEON4 (GRFPC) вставляет дополнительный цикл конвейера в канал передачи результата. Это обеспечивает период ожидания в четыре цикла тактирования на уровне инструкций. В таблице 4.4.10 представлена производительность блока GRFPU, а также время прохождения инструкции по конвейеру.

Таблица 4.4.10 – Производительность GRFPU с GRFPC

Инструкция	Пропускная способность	Период ожидания
FADDS, FADDD, FSUBS, FSUBD, FMULS, FMULD, FSMULD, FITOS, FITOD, FSTOI, FSTOI_RND, FDTOI, FDTOI_RND, FSTOD, FDTOS, FCMPS, FCMPD, FCMPES, FCMPED	1	4
FDIVS	14	16
FDIVD	15	17
FSQRTS	22	24
FSQRTD	23	25

Сложение, вычитание и умножение можно осуществлять каждый такт, обеспечивая высокую производительность для этих наиболее распространенных операций. Операции деления и вычисления квадратного корня имеют более низкую производительность и большую задержку ожидания ввиду сложности алгоритмов, но выполняются параллельно с другими операциями с плавающей запятой в неблокирующем итерационном модуле. Выполнение операций с изменением их очередности и с различными периодами ожидания осуществляется через интерфейс GRFPU, который присваивает каждой операции, выдающей результат на выход, соответствующий идентификатор, как только она завершена.

Операции преобразования типов данных выполняются в модуле с конвейерной организацией и имеют производительность в один такт и период ожидания – в четыре такта. Операции преобразования типов данных обеспечивают преобразование чисел с плавающей запятой в целые и наоборот.

Доступны функции сравнения, обеспечивающие обработку двух различных чисел типа QNaNs. Также доступно перемещение, отрицание и получение абсолютного значения. Эти операции никогда не генерируют исключение незаконченности (сигнал о возникновении исключения незаконченности никогда не передается, так как при сравнении, преобразовании, получении абсолютного значения и перемещении обрабатываются денормализованные числа).

Исключения

GRFPU обнаруживает все исключения, определенные в стандарте IEEE-754, включая обнаружение недопустимых операций (NV), переполнение (OF), обратное переполнение (UF), деление на ноль (DZ) и погрешность (NX). Также реализована генерация специальных результатов таких, как NaNs и бесконечность. Переполнение (OF) и обратное переполнение (UF) обнаруживаются перед округлением. Если в процессе выполнения операции происходит обратное переполнение, результат обнуляется (GRFPU не поддерживает денормализованные числа или постепенное обратное переполнение). Если один из компонентов – денормализованное число, не подлежащее обработке операциями преобразования или арифметическими операциями, подается сигнал о специальном исключении незаконченности (UNF).

Округление

Поддерживаются все четыре режима округления, определенные в стандарте IEEE-754: до ближайшего четного, в направлении плюс бесконечность, в направлении минус бесконечность и в направлении нуля.

Денормализованные числа

GRFPU не обрабатывает денормализованные числа в качестве входных операндов, будет сгенерировано системное прерывание fp_exception с типом прерывания FPU unfinished_FPOP (tt = 2).

Можно эмулировать редкие случаи операций с денормализованными числами, используя операции без плавающей запятой. Операции сравнения, перемещения, отрицания и получения абсолютного значения могут обрабатывать денормализованные числа, при этом не возникают исключения незаконченности. GRFPU не генерирует денормализованные числа во время выполнения арифметических операций и операций преобразования нормализованных чисел. Если точный результат операции – малое число (меньше, чем минимальное значение в нормальном формате), результат обнуляется (с обратным переполнением и установкой флагов погрешности).

Нестандартный режим

GRFPU может работать в нестандартном режиме, где все денормализованные компоненты арифметических операций и операций преобразования рассматриваются как нули (со знаком). Вычисления выполняются с нулевыми компонентами, а не с денормализованными числами, с соблюдением всех правил арифметики с плавающей запятой, включая округление результатов и обнаружение исключений.

NaNs

GRFPU поддерживает обработку чисел типа Не-является-Числом (NaNs), как определено в стандарте IEEE-754. Операции с уведомлением о NaNs (SNaNs) и недопустимые операции (например, inf / inf) генерируют исключение недопустимости операции и выводят в качестве результата QNaN_GEN. Операции с не уведомляющими о NaNs (QNaNs), за исключением FCMPES и FCMPED, не выдают исключения и распространяют QNaNs через операции с плавающей запятой, представляя результаты NaN, согласно таблице 4.4.11. QNaN_GEN – 0x7fffe00000000000 для результата удвоенной точности и 0x7fff0000 – для результатов с одинарной точностью.

Таблица 4.4.11 – Операции с NaNs

		Операнд 2		
		FP	QNaN2	SNaN2
Операнд 1	Отсутствует	FP	QNaN2	QNaN_GEN
	FP	FP	QNaN2	QNaN_GEN
	QNaN1	QNaN1	QNaN2	QNaN_GEN
	SNaN1	QNaN_GEN	QNaN_GEN	QNaN_GEN

GRFPC – блок управления GRFPU

Блок управления GRFPU (GRFPC) используется для подключения GRFPU к блоку целочисленной арифметики процессора LEON. Блок GRFPC обеспечивает планирование, декодирование и диспетчеризацию операций с плавающей запятой для GRFPU, а также управление набором регистров для работы с плавающей запятой, регистром состояния операций с плавающей запятой (FSR) и очередью отложенных прерываний операций с плавающей запятой (FQ). Выполнение операций с плавающей запятой идет параллельно с осуществлением целочисленных инструкций, целочисленный конвейер процессора LEON останавливается только в случае конфликтов операндов или результата.

Регистровый файл GRFPU

Регистровый файл GRFPU состоит из 32 32-битных регистров для операций с плавающей запятой (%f0–%f31). Данный набор регистров доступен с помощью соответствующих команд загрузки и сохранения (LDF, LDDF, STF, STDF) и инструкций для работы с операциями с плавающей запятой (FPop).

Регистр состояний операций с плавающей запятой (FSR)

Блок GRFPC управляет регистром состояний операций с плавающей запятой (FSR), который содержит информацию о режиме и статусе FPU. Все поля регистра FSR, как определено в спецификации SPARC V8, представлены и управляются блоком GRFPU в соответствии с данной спецификацией и стандартом IEEE-754. Бит NS (нестандартный) и поле ftt являются частями системы управления FSR, зависящими от конкретной реализации.

Если бит NS (нестандартный) регистра FSR установлен, то все операции с плавающей запятой будут выполняться в нестандартном режиме, как описано в подпункте «Нестандартный режим». Когда бит NS сброшен, все операции осуществляются в стандартном IEEE-совместимом режиме.

Следующие типы прерываний операций с плавающей запятой никогда не могут возникнуть, и поэтому они не представлены в поле ftt:

- unimplemented_FPop: все операции FPop представлены;
- hardware_error: аппаратные ошибки, не позволяющие возможность возобновления работы;
- invalid_fp_register: не осуществляется проверка на выровненность операнда операций двойной точности (в качестве операндов должны выступать регистры с номером 0 mod 2).

В GRFPU реализован функционал бита qne регистра FSR. Если отложенная очередь операций с плавающей запятой (FQ) пуста, то считывается – 0, в противном случае – 1.

Доступ к регистру FSR осуществляется с помощью инструкций LDFSR и STFSR.

Исключения операций с плавающей запятой и отложенная очередь операций с плавающей запятой

GRFPU реализует модель отложенных прерываний архитектуры SPARC для исключений, вызванных операциями с плавающей запятой (fp_exception). Они могут произойти в результате выполнения одного из следующих условий:

- операция с плавающей запятой вызывает исключение IEEE (ftt = IEEE_754_exception), например выполнение такой недопустимой операции как 0/0, в то время как бит NVM поля TEM установлен (разрешена генерация исключений при возникновении недопустимой операции);
- операция над денормализованными числами с плавающей запятой (в стандартном IEEE-режиме) вызывает исключение unfinished_FPop;
- последовательность ошибок: аномальное ошибочное состояние в FPU, возникшее в результате ошибочного использования операций с плавающей запятой в программном обеспечении операционной системы.

Прерывание откладывается на одну инструкцию операции с плавающей запятой (FPop, FP загрузка/сохранение, FP ветвление), следующую за инструкцией, вызвавшей прерывание (следует принять к сведению, что это может быть не следующая инструкция операции с плавающей запятой в последовательности программы из-за изменения порядка выполнения инструкций и реализации механизма обнаружения исключений в GRFPC). Когда прерывание осуществлено, отложенная очередь операций с

плавающей запятой содержит инструкцию, вызвавшую прерывание, и вплоть до семи FPop инструкций, которые были отправлены в GRFPC, но не были завершены.

После того как прерывание осуществлено, устанавливается бит qne регистра FSR, который остается установленным до тех пор, пока FQ не станет пустой. Инструкция STDFQ считывает двойное слово из очереди отложенных операций с плавающей запятой, первое слово содержит адрес инструкции, второе – код инструкции. В FQ попадают исключительно FPop инструкции. Первый доступ к FQ позволит получить двойное слово с инструкцией, вызвавшей прерывание, последующие двойные слова содержат ожидающие выполнения инструкции операций с плавающей запятой. Программное обеспечение операционной системы должно эмулировать FPop из FQ в том же самом порядке, в котором они были прочитаны из FQ.

Следует принять к сведению, что инструкции в FQ могут располагаться не в том же самом порядке, что и в программной последовательности, поскольку GRFPU выполняет инструкции операций с плавающей запятой не по порядку. Прерывание операции с плавающей запятой никогда не откладывает на потом инструкции, которые задают исходные регистры, регистры результата или коды условий, которые могут быть модифицированы инструкцией, вызвавшей прерывание. По этой причине выполнение или эмуляция инструкций из FQ программным обеспечением операционной системы приведет к тому же состоянию FPU, что и выполнение инструкций в соответствии с программной последовательностью.

4.4.6 AMBA интерфейс

Обзор

Процессоры 1906ВМ016 и 1906ВМ01А6 содержат один ведущий AHB интерфейс. Типы доступов AHB, поддерживаемых процессором, зависят от кэшируемости области памяти, к которой получают доступ, ширины шины AMBA AHB, если соответствующий кэш разрешен и если область памяти отмечена на шине, как найденная.

Сигналы HPROT шины AHB отображают, являются ли данные доступа инструкцией или данными, а также определяют тип доступа – пользовательский или супервизора (операционной системы), как показано в таблице 4.4.12.

Таблица 4.4.12 – Значения сигнала HPROT в зависимости от типа доступа и режима

Тип доступа	Пользователь/Супервизор	HPROT
Инструкция	Пользователь	1100
Инструкция	Супервизор	1110
Данные	Пользователь	1101
Данные	Супервизор	1111
MMU	Любой	1101

В случае атомарного (неделимого) доступа, на шине AMBA будет использоваться блокирующий доступ для гарантии атомарности данных.

Кэшируемость

Возможность кэшируемости для обоих видов кэша базируется на plug & play информации, отображающейся для каждого ведомого устройства AHB. Она показывает, является ли область кэшируемой, эта информация будет использована для определения, какой доступ будет обработан, как кэшируемый.

В микропроцессорах 1906ВМ016, 1906ВМ01А6 в качестве кэшируемых определены следующие области адресов:

0x00000000 – 0x1FFFFFFF;
0x40000000 – 0x7FFFFFFF.

Размер AMBA-доступа

Выборка кэшируемых данных осуществляется с помощью пакетных 64-битных доступов. Доступ данных к некэшируемым областям может быть выполнен только в виде 8-, 16- и 32-битовых доступов, т.е. инструкции LDD и STD не могут быть использованы. Если область отмечена как кэшируемая, тогда кэш данных будет стараться использовать 64-битные доступы. Это означает, что если ведомый AHB не поддерживает 64-битные доступы и отображается, как кэшируемый, тогда программное обеспечение должно выполнять доступ к данным с принудительным кэш-промахом и без 64-битной загрузки (LDD), когда осуществляется доступ к ведомому. Ниже приведен пример, как использовать принудительный кэш-промах при загрузке данных.

```
static inline int load (int address)
   {
    int read_value;
    asm volatile ("lda [%1] 0x01, %0"
                  : "=r" (read_value)
                  : "r" (address)
                 );
    return read_value;
   }
```

В микропроцессорах 1906ВМ016, 1906ВМ01А6 определены области адресов 0 – 0x20000000 и 0x40000000 – 0x80000000, поддерживающие 64-битный доступ.

Инструкции сохранения определяют AMBA-доступы с размером, соответствующим выполняемой инструкции. Инструкции сохранения STD всегда транслируются в 64-битные доступы. В таблицах 4.4.13 и 4.4.14 приведены типы доступов, используемые для инструкций и данных в зависимости от кэшируемости, маски разрядности шины (wbmask) и конфигурации кэша.

Таблица 4.4.13 – Описание типов доступов, wbmask(address) = 0

<table>
<tr><td rowspan="3">Операции процессора</td><td colspan="3">Адресуемая область памяти только 32-битная, wbmask(address) = 0</td></tr>
<tr><td rowspan="2">Область не кэшируемая (1)</td><td colspan="2">Область кэшируемая (1)</td></tr>
<tr><td>Кэш разрешен (2)</td><td>Кэш запрещен</td></tr>
<tr><td>Выборка инструкции</td><td colspan="3">Пакеты 32-битных доступов чтения</td></tr>
<tr><td>Загрузка данных ≤ 32 бита</td><td rowspan="3">Доступы чтения с размером, определенным инструкцией загрузки</td><td rowspan="2">Недопустимо (3, 5). При 32-битном пакетном доступе можно получить некорректные данные</td><td rowspan="2">Доступы чтения с размером, определенным инструкцией загрузки</td></tr>
<tr><td>Загрузка данных ≤ 32 бита через ASI 0x01</td></tr>
<tr><td>Загрузка данных ≤ 32 бита через ASI 0x1C</td><td colspan="2">Доступы чтения с размером, определенным инструкцией загрузки (5)</td></tr>
<tr><td>Загрузка данных 64 бита (LDD)</td><td>Недопустимо (4)</td><td>Недопустимо (3)</td><td>Недопустимо (3)</td></tr>
<tr><td>Сохранение данных ≤ 32 бита</td><td colspan="3">Доступ сохранения с размером, определяемым инструкцией сохранения</td></tr>
<tr><td>Сохранение данных 64 бита (STD)</td><td colspan="3">Недопустимо</td></tr>
</table>

Таблица 4.4.14 – Описание типов доступов, wbmask(address) = 1

Операции процессора	Адресуемая область памяти, wbmask(address) = 1		
	Область не кэшируемая (1)	Область кэшируемая (1)	
		Кэш разрешен (2)	Кэш запрещен
Выборка инструкции	Пакеты 64-битных доступов чтения		
Загрузка данных ≤ 32 бита	Доступы чтения с размером, определенным инструкцией загрузки	Пакет 64-битных доступов	Доступы чтения с размером, определенным инструкцией загрузки
Загрузка данных ≤ 32 бита через ASI 0x01			
Загрузка данных ≤ 32 бита через ASI 0x1C		Доступы чтения с размером, определенным инструкцией загрузки (5)	
Загрузка данных 64 бита (LDD)	Недопустимо (4)	Пакет 64-битных доступов	Одиночный 64-битный доступ чтения
Сохранение данных ≤ 32 бита	Доступ сохранения с размером, определяемым инструкцией сохранения		
Сохранение данных 64 бита (STD)	64-битный доступ сохранения		

Пояснения для таблиц 4.4.13, 4.4.14:

(1) – Кэшируемость определяется через AMBA plug & play.

(2) – Доступы шины для чтения будут выполняться только вынужденным кэш-промахом.

(3) – Процессор LEON4 поддерживает только 64-битные доступы для кэширования данных.

(4) – Обращения к данным в некэшируемых областях могут выполняться только в виде 8-, 16- и 32-битных доступов.

(5) – Загрузка с вынужденным кэш-промахом может быть использована для представления единичного доступа к кэшируемым ведомым шины AHB.

Обработка ошибок

Отклик AHB ERROR, полученный во время выборки инструкции, в обычных условиях приведет к системному прерыванию «instruction_access_exception» (tt = 0x1). Однако если такой отклик будет получен во время выборки инструкций, которые не требуются процессору для продолжения исполнения потока инструкций, то только контроллер кэша инструкций не будет устанавливать бит действительности (valid) в теге кэша, генерации системного прерывания не происходит. Если целочисленный модуль в дальнейшем осуществит выборку инструкции по адресу, который не был отмечен битом действительности, произойдет кэш-промах и будет выполняться новый доступ по этому адресу.

Отклик AHB ERROR, полученный во время выборки данных из кэша данных, в обычных условиях вызовет системное прерывание «data_access_exception» (tt = 0x9). Если ошибка относится к части строки кэша, которая в текущий момент не используется конвейером, то системное прерывание не генерируется и бит действительности для строки кэша не устанавливается.

Отклик ERROR во время обхода таблицы MMU приведет MMU к установке типа ошибки, соответствующего «Ошибке трансляции» (2) в таблице 4.4.8 и генерации системного прерывания instruction_access_exception или data_access_exception, в зависимости от типа доступа, который стал причиной обхода таблицы.

4.4.7 Идентификаторы адресных пространств (ASI)

Для доступа к памяти процессор архитектуры SPARC, кроме адреса, также использует 8-разрядный идентификатор адресного пространства (ASI), обеспечивая до 256 отдельных 32-битовых адресуемых пространств. Во время нормальной работы процессор LEON4 получает доступ к инструкциям и данным, используя ASI 0x8 – 0xB, как определено в стандарте SPARC. Используя инструкции LDA/STA, можно получить доступ к альтернативным адресным пространствам. В таблице 4.4.15 представлено использование ASI для LEON. Для отображения используется только ASI [5:0]; ASI [7:6] не влияет на работу.

Таблица 4.4.15 – Использование ASI

ASI	Использование
0x01	Принудительный промах кэш
0x02	Регистры управления системой (регистр управления кэшем)
0x08, 0x09, 0x0A, 0x0B	Обычный кэшируемый доступ (замена, если кэшируется)
0x0C	Метки кэша инструкций
0x0D	Данные кэша инструкций
0x0E	Метки кэша данных
0x0F	Данные кэша данных
0x10	Сброс кэша инструкций и данных т.к. реализован MMU
0x11	Сброс кэша данных
0x13	Сброс кэша инструкций и данных
0x18	Сброс кэша инструкций, данных и TLB
0x19	Регистры MMU
0x1C	Обращение в обход MMU и кэш
0x1E	Диагностический доступ к меткам отслеживания MMU

Запись в ASI 0x18 сбросит оба буфера ассоциативной трансляции (TLB) MMU, кэш данных и кэш инструкций. Данная операция на несколько тактов заблокирует работу процессора, а затем выполнение асинхронно продолжится параллельно с фоновым сбросом кэша.

ASI 0x1. Принудительный кэш-промах

ASI 0x1 используется для систем без согласования кэшей, чтобы загрузить данные, которые могут быть изменены в фоновом режиме, например, с помощью блоков DMA. Он также может быть использован и по другим причинам, например, в диагностических целях, чтобы осуществить считывание по шине AHB из памяти вне зависимости от состояния кэша.

Адресное отображение этого ASI сопоставляется с регулярным адресным пространством, и если MMU включен, то адрес будет транслирован в обычном режиме. Сохранения в этот ASI будут работать так же, как обычные сохранения данных. В ситуациях, где необходимо гарантировать, что кэш не модифицируется доступом, может быть использован ASI 0x1C.

ASI 0x2. Системные регистры управления

ASI 0x2 содержит несколько управляющих регистров, которые не были назначены в качестве регистров %asr. Эти регистры могут быть считаны и записаны только с использованием 32-битных инструкций LDA / STA.

Все регистры кэша доступны через операции LDA/STA в ASI 0x2. В таблице 4.4.16 приведены адреса регистров.

Таблица 4.4.16 – Карта адресов ASI 0x2

Адрес	Регистр
0x00	Регистр управления кэша
0x04	Зарезервировано
0x08	Конфигурационный регистр кэша инструкций
0x0C	Конфигурационный регистр кэша данных

ASI 0x8–0xB. Данные/Инструкции

В таблице 4.4.17 приведено распределение адресных пространств обычного кэшируемого доступа.

Таблица 4.4.17 – Использование ASI

ASI	Адресное пространство
0x08	Пользовательские инструкции (User instruction)
0x09	Инструкции операционной системы (Supervisor instruction)
0x0A	Пользовательские данные (User data)
0x0B	Данные операционной системы (Supervisor data)

Эти ASI назначены стандартом SPARC для обычных выборок данных и инструкций.

Доступ к ASI инструкций, выполняемый напрямую с помощью инструкций LDA/STA, в LEON4 не поддерживается. Использование LDA / STA вместе с ASI данных пользователя/супервизора (операционной системы) даст эффект в соответствии с сигналом HPROT, выдаваемого процессором в соответствии с пунктом «4.4.6 AMBA интерфейс», но контроль доступа MMU будет по-прежнему осуществляться в соответствии с состоянием бита S регистра %psr (пользователь/супервизор).

ASI 0xC–0xF. I-кэш метки/данные, D-кэш метки/данные

ASI 0xC–0xF обеспечивают диагностический доступ к кэш-памяти инструкций и данных. Эти ASI доступны только через 32-разрядные инструкции LDA/STA. Эти ASI не могут быть использованы во время выполнения операции FLUSH. В рабочем режиме доступ к ASI 0xC и 0xD может быть осуществлен, только если кэш инструкций отключен (сброшено поле ICS регистра управления кэша).

Те же биты адреса, которые обычно используются в качестве индекса, также используются для индексации кэша при диагностическом доступе. Для многоканального кэша биты, которые обычно используются в качестве тега, будут использованы при выборе пути (канала) кэша для чтения/записи. Остальные адресные биты не важны, и приводят к циклическому возврату по карте адресов. Смотри таблицы 4.4.18 – 4.4.21.

Таблица 4.4.18 – ASI 0xC диагностики метки для 2-канального кэша инструкций 4 Кбайт/канал, 32 байта/строка

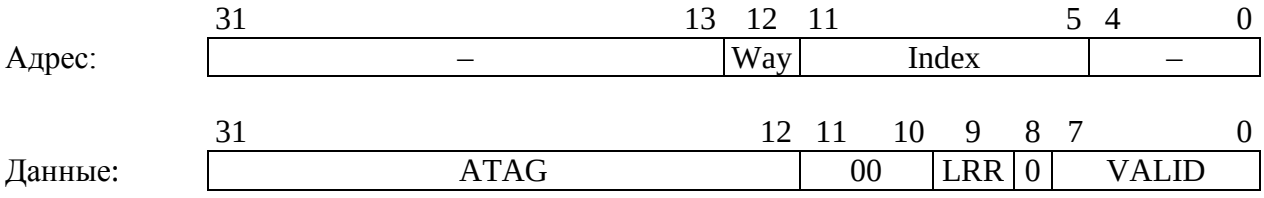


Таблица 4.4.19 – ASI 0xD диагностики данных для 2-канального кэша инструкций 4 Кбайт/канал, 32 байта/строка

	31 … 13	12	11 … 5	4 … 0
Адрес:	–	Way	Index	Offset

	31 … 0
Данные:	Кэшированное слово данных

Таблица 4.4.20 – ASI 0xE диагностики метки для 2-канального кэша данных 4 Кбайт/канал, 16 байт/строка

	31 … 13	12	11 … 4	3 … 0
Адрес:	–	Way	Index	–

	31 … 12	11 10	9	8	7 … 0
Данные:	ATAG	00	LRR	0	VALID

Таблица 4.4.21 – ASI 0xF диагностики данных для 2-канального кэша данных 4 Кбайт/канал, 16 байт/строка

	31 … 13	12	11 … 4	3 … 0
Адрес:	–	Way	Index	Offset

	31 … 0
Данные:	Кэшированное слово данных

Описание полей:
Канал (Way) – Определяет используемый канал (путь) кэша.
Индекс (Index) – Порядковый номер строки кэша.
Метка адреса (ATAG) – содержит метку адреса строки кэша.
LRR – используется LRR алгоритмом для хранения истории замены, иначе – 0.
Действительный (VALID) – когда установлено, строка кэша содержит действительные данные.

В LEON4 кэши имеют только один бит действительности на строку кэша, которая дублируется для всего 8-битного диагностического поля, чтобы сохранить обратную совместимость программного обеспечения.

ASI 0x10, 0x11, 0x13, 0x18. Сброс (flush)

По историческим причинам есть несколько ASI, которые сбрасывают кэш по-разному.
Запись в ASI 0x10 сбросит кэш инструкций целиком. Кэш данных также будет сброшен, т.к. реализован MMU.
Запись в ASI 0x11 сбросит только кэш данных.
Запись в ASI 0x13 сбросит кэш инструкций и кэш данных.
Запись в ASI 0x18 сбросит оба кэша и MMU TLB. В отличие от других ASI, выполнение будет заблокировано до завершения FLUSH.

ASI 0x19. Регистры MMU

Этот ASI предоставляет доступ к регистрам контроля и статуса MMU. Реализованы следующие регистры, смотри таблицу 4.4.22.

Таблица 4.4.22 – Регистры MMU (ASI = 0x19)

Адрес	Регистр
0x000	Регистр управления MMU
0x100	Регистр указателя контекста MMU
0x200	Регистр контекста MMU
0x300	Регистр статуса сбоя MMU
0x400	Регистр адреса сбоя MMU

ASI 0x1C. Обход MMU и кэша

Выполнение доступа через ASI 0x1C будет действовать, как если бы были отключены MMU и кэш. Адрес не будет транслироваться и кэш не будет использоваться или обновляться при выполнении доступа.

ASI 0x1E. Диагностический доступ к меткам отслеживания MMU

В процессорах 1906BM016, 1906BM01A6 MMU реализовано использование раздельных меток отслеживания (snoop tags), они доступны через ASI 0x1E. Это в первую очередь полезно для тестирования оперативной памяти, но не должно использоваться во время нормальной работы. Этот ASI адресуется таким же образом, как и обычные диагностические ASI: 0xC, 0xE. Формат чтения / записи данных показан в таблице 4.4.23 (для кэша данных 4 КБ/канал).

Таблица 4.4.23– Формат данных

31 … 12	11 … 2	1	0
ATAG	–	PAR	IV

31 – 12 Метка адреса (ATAG) – Метка физического адреса строки кэша
11 – 2 Зарезервировано
1 Четность (PAR) – Не используется
0 Недействителен (IV) – Когда установлен, строка кэша недействительна и при доступе процессора будет происходить промах кэша

4.4.8 Регистры конфигурации

Регистры PSR, WIM, TBR

Регистры %psr, %wim, %tbr реализованы согласно стандарту SPARC V8, они представлены в таблицах 4.4.24 – 4.4.26, подробное описание полей и битов управления можно найти в пункте «5.3.2 Регистры управления/статуса блока целочисленной арифметики (IU)».

Таблица 4.4.24 – Регистр состояния процессора (%psr) LEON4

31 … 28	27 … 24	23 … 20	19 … 14	13	12	11 … 8	7	6	5	4 … 0
IMPL	VER	ICC	–	EC	EF	PIL	S	PS	ET	CWP
0xF	0x3	0x0	0x0	0	0	0x0	1	1	0	0x0
r	r	r	r	r	rw	rw	rw	rw	rw	rw

31 – 28 Идентификатор реализации (IMPL) – Аппаратно запрограммировано значение 1111_2
27 – 24 Идентификатор версии (VER) – Аппаратно запрограммировано значение 0011_2 для процессоров LEON3 / LEON4

23 – 20 Целочисленные коды условий (ICC)
19 – 14 Зарезервировано
13 Разрешение сопроцессора (EC) – В реализации отсутствует сопроцессор
12 Разрешение блока арифметики с плавающей запятой (EF)
11 – 8 Уровень прерываний процессора (PIL) – Определяет низший уровень запроса прерывания (IRQ), который сгенерирует системное прерывание
7 Супервизор (S) – Определяет режим процессора, 1 – супервизор (операционная система), 0 – пользователь
6 Предшествующий супервизор (PS)
5 Разрешение прерываний (ET)
4 – 0 Указатель текущего окна (CWP)

Таблица 4.4.25 – Регистр маски недопустимости окна (%wim)

31 … 8	7 … 0
	WIM
0x000000	n/r
r	rw

31 – 8 Зарезервировано
7 – 0 Маска недопустимости окна (WIM) – Если инструкция SAVE, RESTORE или RETT должна изменить указатель на текущее регистровое окно (%psr.CWP) таким образом, что WIM[%psr.CWP] = 1, то выполнение такой инструкции будет прервано генерацией системного прерывания window_overflow (в случае SAVE) или window_underflow (в случае RESTORE или RETT).

Таблица 4.4.26 – Базовый регистр системных прерываний (%tbr)

31 … 12	11 … 4	3 … 0
TBA	TT	–
0x00000	0x00	0x0
rw	r	r

31 – 12 Базовый адрес системного прерывания (TBA) – 20 старших бит, используемых для адресации таблицы системных прерываний.
11 – 4 Тип прерывания (TT) – тип последнего принятого системного прерывания. Только для считывания
3 – 0 Зарезервировано

Стоит отметить, что системное прерывание автоматически выделяет новое окно под хранение PC, nPC и нужды обработчика системного прерывания. Исходя из этого факта, в ходе инициализации в регистре %wim следует установить бит, отвечающий не за первое используемое регистровое окно, а за соседнее с ним (окно с номером %psr.CWP + 1 по модулю 8). Таким образом, под системное прерывание всегда будет «зарезервировано» одно окно. Подразумевается, что бит %psr.ET установлен, а в регистре %tbr записан базовый адрес таблицы прерываний, в которой есть вызовы обработчиков прерываний window_overflow и window_underflow. Если в ходе инициализации полю %psr.CWP присвоено значение 0x00 (окно 0), то чтобы зарезервировать окно 1 можно использовать следующую инструкцию: «wr %g0, 0x02, %wim». В таком случае вложенные вызовы не листовых подпрограмм, требующие выделения нового окна через инструкцию SAVE, будут изменять CWP следующим образом: 0 → 7 → 6 → … →2. Следующий вложенный вызов, который должен уменьшить CWP на 1, приведёт к генерации системного прерывания window_overflow. Обработчик данного системного прерывания должен переместить данные из окна 0 в стек и циклически сдвинуть вправо значение

регистра %wim, что равносильно вычитанию 1 по модулю 8 (количество реализованных в процессоре регистровых окон). После возврата из обработчика системного прерывания повторное выполнение прерванного ранее вложенного вызова пройдет без генерации системного прерывания. После декремента значения %psr.CWP по-прежнему останется одно зарезервированное под вызов системного прерывания регистровое окно (но теперь это будет окно 0). При возврате из глубокой цепочки вызовов (превышающей количество физически реализованных регистровых окон процессора) при возникновении потребности регистровые окна будут восстановлены из стека. За данный процесс отвечает обработчик системного прерывания window_underflow.

Также с особой осторожностью стоит относиться к выделению новых окон из обработчика прерывания, поскольку если последнее «свободное окно» уже было выделено под обработчик текущего системного прерывания, произойдет новое системное прерывание window_overflow. Если в этот момент времени бит %psr.ET будет сброшен (автоматически сбрасывается при возникновении системного прерывания), процессор перейдет в режим ошибки. Стоит отметить, что для корректной работы обработчиков системных прерываний window_overflow и window_underflow требуется, чтобы под стек было выделено необходимое пространство, т.е. на этапе инициализации процессора был установлен указатель вершины стека. Часто в %sp начального регистрового окна – в приведенном случае это окно 0 – заносят адрес выравненный по границе максимального размера строки данных кэша (32 байта на строку), близкий к верхней границе физической памяти RAM (при использовании помехоустойчивого кодирования рекомендуется инициализировать область памяти, отведенную под стек, см. подраздел «3.8 Требования по инициализации областей памяти при использовании помехоустойчивого кодирования»).

Регистр конфигурации процессора

Специализированный регистр 17 (%asr17) предоставляет информацию об установке различных параметров конфигурации. Он может использоваться для улучшения производительности программного обеспечения или поддержки регистрации в многопроцессорных системах. Доступ к регистру можно получить через инструкцию RDASR. Регистр имеет топологию, представленную в таблице 4.4.27.

Таблица 4.4.27 – Регистр конфигурации LEON4 (%asr17)

31 28	27	26	25	24 18	17	16 15	14	13	12	11 10	9	8	7 5	4 0
INDEX	DBP	–	DBPM	–	CS	CF	DWT	SVT	LD	FPU	M	V8	NWP	NWIN
0x0	0	0	1	0x00	0	0x0	0	0	0	0x1	1	1	0x2	0x7
r	rw	r	rw	r	r	r	rw	rw	r	r	r	r	r	r

31 – 28 Индекс процессора (INDEX). Аппаратно запрограммировано значение «0000»

27 Отключение прогнозирования переходов (DBP). Если установлено, то прогнозирование переходов не будет осуществляться

26 Зарезервировано

25 Отключение прогнозирования переходов при промахе в кэш (DBPM). Если установлено, то не осуществляется выборка инструкций в кэш (и возможно обход таблицы MMU) для прогнозируемых инструкций, которые могут быть аннулированы

24 – 18 Зарезервировано

17 Переключение тактирования разрешено (CS). Аппаратно запрограммировано значение «0», переключение частоты AHB и CPU не доступно

16, 15	Частота тактирования CPU (CF). Ядро CPU работает с частотой тактирования AHB (CF+1)
14	Выключение системного прерывания при ошибке записи (DWT). Если установлено, системное прерывание при ошибке записи (tt = 0x2b) игнорируется. Очищается после сброса
13	Разрешение одновекторной обработки системных прерываний (SVT). Функционал доступен. Очищается после сброса
12	Задержка загрузки (LD). Аппаратно запрограммировано значение «0», конвейер использует одну тактовую задержку загрузки
11, 10	Блок арифметики с плавающей запятой (FPU). Аппаратно запрограммировано значение «01», в качестве блока арифметики с плавающей запятой используется GRFPU
9	Инструкции умножения с накоплением (MAC). Аппаратно запрограммировано значение «1», инструкции UMAC / SMAC доступны
8	Инструкции умножения и деления SPARC V8 (M). Аппаратно запрограммировано значение «1», данные инструкции доступны
7 – 5	Количество реализованных точек контроля данных (NWP) – 2
4 – 0	Количество реализованных регистровых окон – 8 (NWIN + 1)

Аппаратные точки останова/контрольные точки данных

Целочисленный модуль имеет две аппаратные точки останова. Каждая из точек останова состоит из пары регистров (%asr24/25, %asr26/27), один из которых содержит адрес останова, второй – маску, что показано в таблице 4.4.28.

Таблица 4.4.28 – Регистры точек останова

%asr24, %asr26

31 … 2	1	0
WADDR[31:2]	–	IF
n/r	0	0
rw	r	rw

%asr25, %asr27

31 … 2	1	0
WMASK[31:2]	DL	DS
n/r	0	0
rw	rw	rw

WADDR – адрес, с которым осуществляется сравнение;

WMASK – битовая маска, которая определяет контролируемые биты адреса. Если бит WMASK[x] сброшен, то значение бита WADDR[x] игнорируется при сравнении;

IF – останов при выборке инструкций по совпадению комбинации адреса/маски;

DL – останов при считывании данных по совпадению комбинации адреса/маски;

DS – останов при сохранении данных по совпадению комбинации адреса/маски.

Срабатывание аппаратной точки останова/контроля данных приводит к генерации системного прерывания с tt = 0x0B (watchpoint_detected).

Если все 3 бита IF, DL и DS сброшены, то аппаратная точка останова/контроля данных не активна.

Регистр управления кэша

Работа кэша инструкций и данных управляется через общий регистр управления кэша (CCR) (см. таблицу 4.4.29). Описание доступа к данному регистру располагается в пункте «ASI 0x2. Системные регистры управления».

Таблица 4.4.29 – Регистр управления кэша

31	30	29	28	27 – 24	23	22	21	20 – 19	18	17	16	15	14	13 – 12	11 – 10	9 – 8	7 – 6	5	4	3 – 2	1 – 0
–	STE	–	PS	TB	DS	FD	FI	FT	–	ST	–	IP	DP	ITE	IDE	DTE	DDE	DF	IF	DCS	ICS
0	0	0	0	0x0	0	0	0	0x0	0	1	0	0	0	0x0	0x0	0x0	0x0	0	0	0x0	0x0
r	r	r	r	r	rw	rw	rw	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw

31	Зарезервировано
30	Флаг меток отслеживания (STE). Устанавливается, когда обнаружена ошибка четности в аппаратно реализованных метках отслеживания. Функционал недоступен
29	Зарезервировано
28	Выбор четности (PS) – Если установлено, при диагностическом считывании 4 контрольных бита возвращаются в разряд младших значащих битов, в противном случае возвращается метка или слово данных. Функционал не доступен
27 – 24	Тестовые биты (TB) – Если установлено, над контрольными битами будет выполнена операция «ИСКЛЮЧАЮЩЕЕ ИЛИ» с тестовыми битами TB во время диагностической записи. Функционал недоступен
23	Разрешение на отслеживание кэша данных (DS) – Если бит установлен, то активируется механизм отслеживания кэша данных (см. «Отслеживание (Snooping)» пункта «4.4.3 Подсистема кэша»)
22	Сбрасывание кэша данных (FD). Если установлено, сбрасывается кэш данных. Всегда считывается, как ноль
21	Сбрасывание кэша инструкций (FI). Если установлено, сбрасывается кэш инструкций. Всегда считывается, как ноль
20, 19	Схема защиты данных (FT). «00» = отсутствует
18	Зарезервировано
17	Раздельные метки отслеживания (ST) Если бит установлен, то в кэше данных используются раздельные аппаратно реализованные метки отслеживания (snoop tag). «1» = присутствует
16	Зарезервировано
15	Ожидание сбрасывания кэша инструкций (IP). Этот бит устанавливается во время операции сбрасывания кэша инструкций
14	Ожидание сбрасывания кэша данных (DP). Этот бит устанавливается во время операции сбрасывания кэша данных
13, 12	Ошибки метки инструкции (ITE) – Количество обнаруженных ошибок при проверке на четность в кэше метки инструкции. Функционал недоступен
11, 10	Ошибки данных инструкции (IDE) – Количество обнаруженных ошибок при проверке на четность в кэше данных инструкции. Функционал недоступен
9, 8	Ошибки метки данных (DTE) – Количество обнаруженных ошибок при проверке на четность в кэше метки данных. Функционал недоступен
7, 6	Ошибки данных (DDE) – Количество обнаруженных ошибок при проверке на четность в кэше данных. Функционал недоступен
5	Фиксация кэша данных при прерывании (DF) – Если установлено, кэш данных автоматически фиксируется при принятии асинхронного прерывания
4	Фиксация кэша инструкций при прерывании (IF) – Если установлено, кэш инструкций автоматически фиксируется при принятии асинхронного прерывания

3, 2	Статус кэша данных (DCS) – Определяет текущий статус кэша данных согласно следующему: X0 = отключен, «01» = зафиксирован, «11» = включен
1, 0	Статус кэша инструкций (ICS) – Определяет текущий статус кэша данных согласно следующему: X0 = отключен, «01» = зафиксирован, «11» = включен

Если установлен бит DF или IF, соответствующий кэш фиксируется при принятии асинхронного прерывания. Это может быть полезно в системе, работающей в реальном времени, для обеспечения более точного вычисления времени выполнения для наихудшего случая кодового сегмента. Работа сортировщика прерываний не исключает строки кэша, при этом при возврате управления к прерванной задаче статус кэша до и после прерывания сохраняется. Если кэш останавливается при прерывании, его можно снова включить в CCR. Это обычно делается по окончании работы сортировщика прерываний прежде, чем управление возвращается к прерванной задаче.

Регистры конфигурации кэша

Конфигурация двух кэшей определяется в двух регистрах: регистре конфигурации кэша инструкций и регистре конфигурации кэша данных. Эти регистры только для считывания, за исключением поля REPL, и указывают размер и конфигурацию кэшей (см. таблицу 4.4.30). Описание доступа к данным регистрам располагается в пункте «ASI 0x2. Системные регистры управления».

Таблица 4.4.30 – Регистр конфигурации кэша

31	30	29 28	27	26 24	23 20	19	18 16	15 4	3	2 0
CL	–	REPL	SN	WAYS	WSIZE	LR	LSIZE	–	M	–
0	0	0x1	*	0x1	0x2	0	*	0x000	1	0
r	r	rw	r	r	r	r	r	r	r	r

31	Блокировка кэша (CL). Устанавливается, если реализована блокировка кэша
30	Зарезервировано
29, 28	Принцип замены кэша (REPL). 00 – отсутствие принципа замены (кэш с прямым отображением), 01 – с наиболее давним использованием (LRU), 10 – с наиболее давней произведенной заменённой (LRR) 11 – псевдослучайный
27	Отслеживание кэша (SN). Устанавливается, если реализован механизм сохранения когерентности кэша с использованием отслеживания (snooping). Значение в регистре конфигурации кэша инструкций – 0. Значение в регистре конфигурации кэша данных – 1
26 – 24	Ассоциативность кэша (WAYS). Количество каналов в кэше: 000 – с прямым отображением, 001 – 2-канальная ассоциативность, 010 – 3-канальная ассоциативность, 011 – 4-канальная ассоциативность
23 – 20	Размер канала (WSIZE). Указывает размер (в Кбайтах) каждого канала кэша. Размер = 2^{WSIZE}
19	Зарезервировано. Для ядра LEON4 всегда считывается как 0
18 – 16	Размер строки (LSIZE). Указывает размер (в словах) каждой строки кэша. Размер строки = 2^{LSIZE} Значение в регистре конфигурации кэша инструкций – 0x3. Значение в регистре конфигурации кэша данных – 0x2
15 – 4	Зарезервировано

3	Наличие MMU (M). Устанавливается, если блок управления памятью (MMU) реализован в составе процессора
2 – 0	Зарезервировано

Счетчик тактов

Пара вспомогательных регистров состояния %asr22 и %asr23 представляет собой внутренний счетчик тактов, значение которого может быть считано программным путем без формирования какого-либо запроса на внутреннюю шину AMBA. %asr22 содержит старшие значащие разряды счетчика (см. таблицу 4.4.31), а %asr23 – младшие (см. таблицу 4.4.32). Значение полностью соответствует метке времени, используемой в системных буферах трассировки.

Счетчик метки времени будет инкрементироваться, если какой-либо из буферов трассировки активирован, или если установлен его принудительный запуск через интерфейс регистров управления DSU, или если поле DUCNT регистра %asr22 сброшено процессором.

Таблица 4.4.31 – Старшие значащие разряды счетчика тактов LEON4 (%asr22)

31	30 … 0
DUCNT	UPCNT(62:32)
1	0x00000000
rw	r

31	Отключение счетчика тактов (DUCNT). Когда бит установлен, то инкрементирующийся счетчик тактов может быть отключен. Когда сброшен, счетчик будет инкрементироваться каждый такт системной частоты
30 – 0	Значение счетчика (UPCNT(62:32)). Старшие значащие разряды внутреннего счетчика тактов

Таблица 4.4.32 – Младшие значащие разряды счетчика тактов LEON4 (%asr23)

31 … 0
UPCNT(31:0)
0x00000000
r

31 – 0	Значение счетчика (UPCNT(31:0)). Младшие значащие разряды внутреннего счетчика тактов

Регистр управления MMU

Регистр управления MMU размещен в адресном пространстве с идентификатором ASI = 0x19, с нулевым смещением (см. таблицу 4.4.33).

Таблица 4.4.33 – Регистр управления MMU

31 … 28	27 … 24	23 … 21	20 … 18	17 … 16	15	14	13 … 2	1	0
IMPL	VER	ITLB	DTLB	–	TD	ST	–	NF	E
0x0	0x0	0x4	0x4	0x0	n/r	1	0x000	0	0
r	r	r	r	r	rw	r	r	rw	rw

31 – 28	Идентификатор реализации (IMPL) – Идентификатор реализации MMU
27 – 24	Версия (VER) – Идентификатор версии MMU

23 – 21 Количество записей TLB инструкций (ITLB) – Количество записей TLB инструкций, рассчитывается, как 2^{ITLB}

20 – 18 Количество записей TLB данных (DTLB) – Количество записей TLB данных, рассчитывается, как 2^{DTLB}

17 – 16 Зарезервировано

15 Запрещение TLB (TD) – Если поле установлено, TLB будет запрещен, каждый доступ данных будет генерировать обход таблицы страниц MMU

14 Раздельные TLB (ST) – Этот бит сообщает пользователю о наличии общего (0) или раздельных (1) TLB для инструкций и данных

13 – 2 Зарезервировано

1 Нет сбоя (NF) – Если NF = 0, любой сбой, определяемый MMU, приведет к обновлению регистра статуса сбоя MMU (FSR) и регистра адреса сбоя MMU (FAR) и генерации сбоя в ядро процессора. Когда NF = 1, сбой при доступе к ASI = 0x09 будет обрабатываться, как в случае с NF = 0. При доступах к любому другому ASI, FSR и FAR будут обновлены, но генерация сбоя в ядро процессора выполняться не будет

0 Разрешение MMU (E). «1» – MMU разрешен, «0» – MMU запрещен

Регистр указателя контекста MMU

Регистр указателя контекста MMU размещен в адресном пространстве с идентификатором ASI = 0x19, со смещением 0x100. Регистр указателя контекста MMU и регистр контекста MMU вместе определяют положение корневой таблицы страниц для текущего контекста. Определение полей регистра приведено в таблице 4.4.34 и соответствует спецификации SPARC V8.

Таблица 4.4.34 – Регистр указателя контекста MMU

31 2	1 0
CONTEXT TABLE POINTER	–
n/r	0x0
rw	r

31 – 2 Указатель на таблицу контекстов (CONTEXT TABLE POINTER) – Биты 35:6 физического адреса (важно помнить, что адрес сдвинут на 4 разряда)

1, 0 Зарезервировано

Регистр контекста MMU

Регистр контекста MMU размещен в адресном пространстве с идентификатором ASI = 0x19, со смещением 0x200. Определение полей регистра приведено в таблице 4.4.35 и соответствует спецификации SPARC V8.

Таблица 4.4.35 – Регистр контекста MMU

31 8	7 0
–	CONTEXT
0x000000	0x00
r	rw

31 – 8 Зарезервировано

7 – 0 Идентификатор текущего контекста (CONTEXT)

В LEON4 для вычисления адреса используется операция логического ИЛИ над битами идентификатора контекста и младшими битами указателя на таблицу контекстов,

таким образом можно использовать меньше бит контекста для сокращения требований размера/выравнивания таблицы контекстов.

Регистр статуса сбоя MMU

Регистр статуса сбоя MMU размещен в адресном пространстве с идентификатором ASI = 0x19, со смещением 0x300, его описание основывается на спецификации SPARC V8. Спецификация SPARC V8 гласит, что весь регистр статуса сбрасывается при считывании, в LEON4 только бит FAV сбрасывается при считывании. В реализации LEON4 бит FAV всегда устанавливается при сбое, поэтому его можно использовать в качестве бита достоверности остальных полей. Регистр представлен в таблице 4.4.36.

Таблица 4.4.36 – Регистр статуса сбоя MMU

31 – 18	17 – 10	9, 8	7 – 5	4 – 2	1	0
–	EBE	L	AT	FT	FAV	OW
0x0000	0x00	0x0	0x0	0x0	0	0
r	r	r	r	r	r	r

31 – 18 Зарезервировано
17 – 10 Внешняя ошибка шины (EBE) – Никогда не возникает в LEON4
9, 8 Уровень (L) – Уровень записи в таблице страниц, вызвавшей ошибку
7 – 5 Тип доступа (AT)
4 – 2 Тип сбоя (FT) – Устанавливается в соответствии с таблицей 4.4.8
1 Адрес сбоя действителен (FAV) Отчищается при считывании, всегда устанавливается при возникновении сбоя
0 Перезаписано (OW) – Наблюдаются множественные сбои одинакового приоритета

Регистр адреса сбоя MMU

Регистр адреса сбоя MMU размещен в адресном пространстве с идентификатором ASI = 0x19, со смещением 0x400. Определение полей регистра приведено в таблице 4.4.37 и соответствует спецификации SPARC V8.

Таблица 4.4.37 – Регистр адреса сбоя MMU

31 – 12	11 – 0
FAULT ADDRESS	–
n/r	0x000
rw	r

31 – 12 Адрес сбоя (FAULT ADDRESS) – 20 старших бит виртуального адреса, вызвавшего сбой трансляции.
11 – 0 Зарезервировано.

Регистр управления защитой

Вспомогательный регистр состояния %asr16 позволяет управлять режимом инвалидации кэша инструкций и кэша данных. Описание данного регистра приведено в таблице 4.4.38.

Некоторые рекомендации по выбору наиболее подходящей конфигурации приведены в подразделе «2.6 Механизмы сбоеустойчивости».

Таблица 4.4.38 – Регистр управления защитой (%asr16)

31 – 23	22	21 – 0
–	SEI	–
0x000	0	0x000000
r	rw	r

31 – 23 Зарезервировано

22 Инвалидация при одиночной ошибке (SEI) – Если бит установлен, то строка кэша с детектированной исправимой ошибкой будет принудительно считываться из внешней памяти. Если бит сброшен, то повторное заполнение строки кэша будет производиться только при обнаружении неисправимой ошибки

21 – 0 Зарезервировано

4.5 Контроллер PROM / IO

4.5.1 Общие сведения

Контроллер PROM / IO реализует мост между внешней памятью и шиной AHB. Он обеспечивает подключение двух типов устройств: PROM и отображаемых в памяти устройств ввода-вывода (IO). Поддерживается два режима организации шины данных: 32-битный и 8-битный (см. пункт 4.5.5). Память PROM может быть обеспечена средствами обнаружения и исправления ошибок кодом Хэмминга (39,32) или кодом Хэмминга (13,8), в зависимости от разрядности используемой шины данных PROM. Код Хэмминга обеспечивает обнаружение до двух ошибок и исправление одной ошибки на кодовое слово.

Структурная схема контроллера PROM / IO приведена на рисунке 4.5.1 (реализовано подключение PROM в 32-битном режиме организации шины данных, о чем свидетельствует подача низкого логического уровня сигнала на вход CFG_8B_PROM).

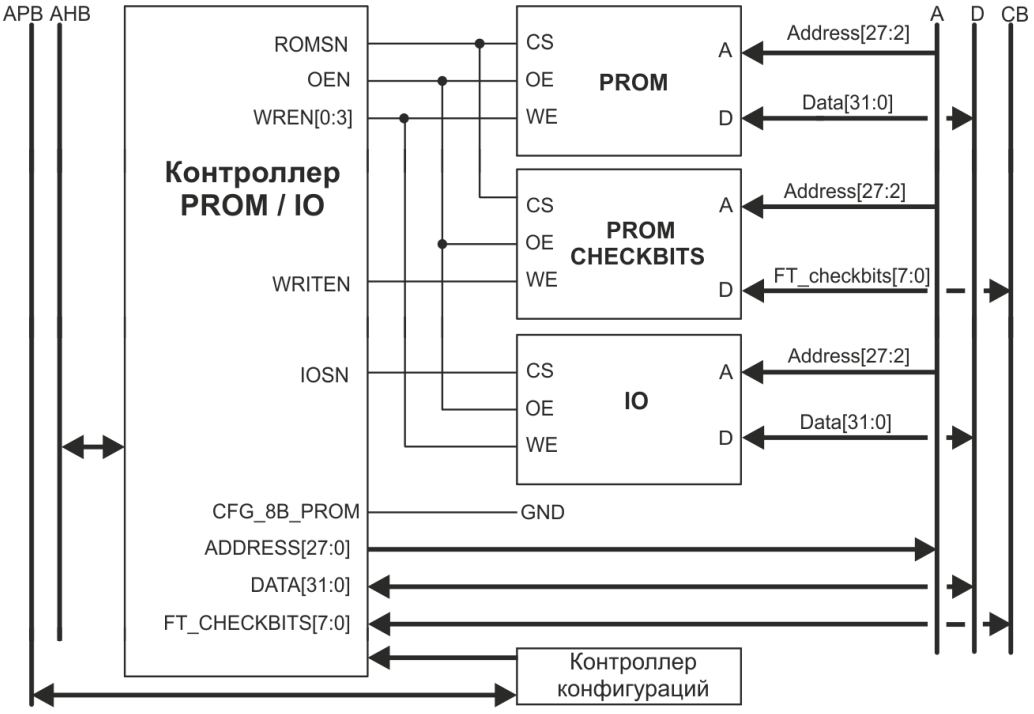


Рисунок 4.5.1 – Структурная схема контроллера PROM / IO (PROM в 32-битном режиме организации шины данных)

Временными параметрами функционирования и помехоустойчивого кодирования контроллера PROM / IO можно управлять с помощью регистров, описанных в пункте «4.3.4 Регистры». Данный контроллер памяти отвечает за два адресных пространства (PROM и IO). Область PROM отображается в диапазоне адресов 0x00000000 – 0x0FFFFFFF, является кэшируемой с упреждающей выборкой инструкций. Область IO отображается в диапазоне адресов 0x20000000 – 0x2FFFFFFF, она отмечена как некэшируемая в plug & play конфигурации шины AHB.

4.5.2 Доступ к области PROM

Для организации доступа к области PROM используется один общий сигнал выбора кристалла – ROMSN. В качестве сигналов разрешения записи микросхем используется шина WREN[3:0]. Вывод WREN[0] отвечает за линии данных DATA[31:24], WREN[1] – за DATA[23:16], WREN[2] – за DATA[15:8], WREN[3] – за DATA[7:0]. Блок проверочных бит в качестве сигнала разрешения записи использует вывод WRITEN. Активный уровень WRITEN выставляется, если хотя бы один из разрядов WREN находится в активном состоянии. Если область PROM планируется использовать только с активированным помехоустойчивым кодированием, то вместо каждого из разрядов WREN можно подключить WRITEN. Это допустимо, поскольку в таком случае все обращения записи с разрядностью меньшей ширины используемой внешней шины данных будут обрабатываться через алгоритм чтения-модификации-записи. Сигнал разрешения чтения – OEN.

Выходной сигнал READ сбрасывается на все время выполнения операции записи, в том числе и на стадиях lead-in и lead-out, в остальное время он остается установлен. Данный сигнал можно использовать для управления направлением буферов шины данных микросхем внешней памяти, если таковые присутствуют в устройстве.

Если используется помехоустойчивое кодирование, то к микросхеме, отвечающей за данный функционал, в качестве сигнала разрешения записи подключается вывод WRITEN, в качестве шины данных выступают младшие разряды шины FT_CHECKBITS, остальные сигналы управления совпадают с таковыми для основных микросхем PROM. При вычислении синдрома ошибки используются только FT_CHECKBITS[6:0], при выполнении операций записи старшие разряды этой шины будут иметь нулевое значение.

На рисунке 4.5.2 представлена временная диаграмма одиночной записи в область PROM (со всеми возможными дополнительными тактами). Более подробная информация о конфигурировании временной диаграммы контроллера представлена в пункте «4.5.4 Управление временной диаграммой контроллера PROM / IO».

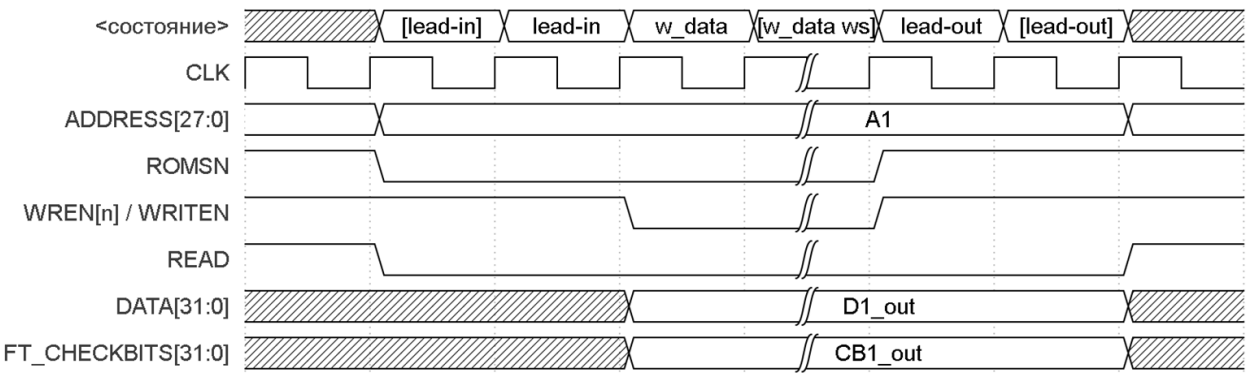


Рисунок 4.5.2 – Временная диаграмма одиночной записи в область PROM

Если разрядность обращения превышает ширину используемой внешней шины данных, то контроллер формирует последовательность обращений, реализующую запрос. На рисунке 4.5.3 приведен пример выполнения 64-битного доступа записи в PROM, находящейся в режиме 32-битной организации шины данных, без каких-либо дополнительных тактов.

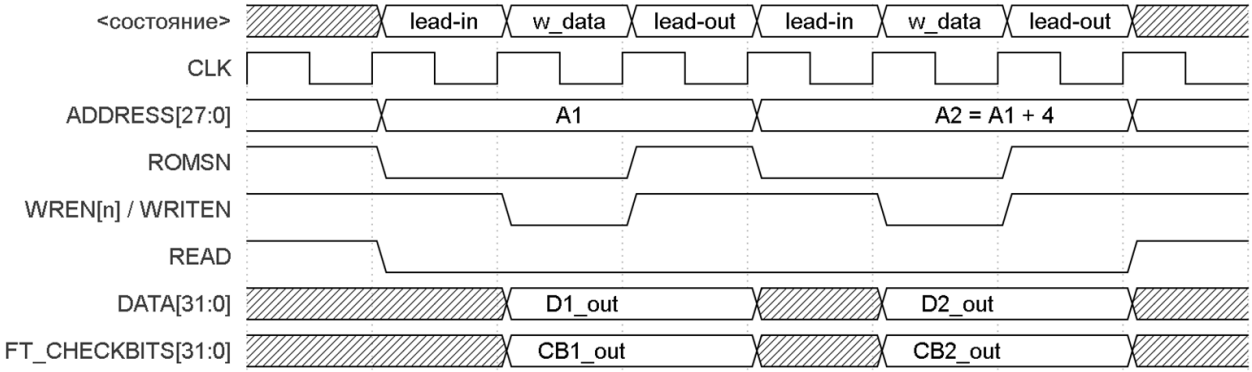


Рисунок 4.5.3 – Временная диаграмма 64-х битной записи в область PROM без дополнительных тактов

На рисунке 4.5.4 в качестве примера приведена временная диаграмма чтения из области PROM или IO с тремя дополнительными тактами активного состояния.

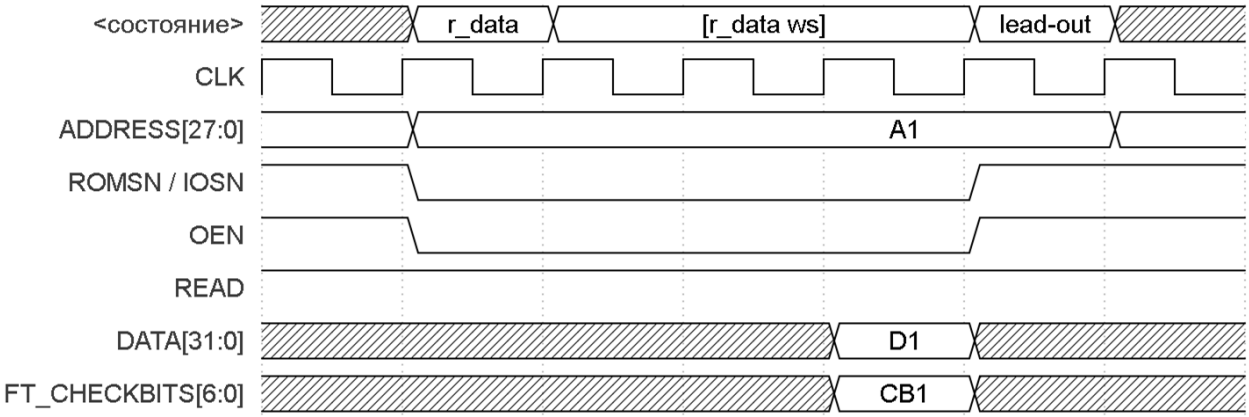


Рисунок 4.5.4 – Временная диаграмма чтения из области PROM или IO с тремя дополнительными тактами активного состояния

Если разрядность обращения меньше чем ширина используемой внешней шины данных, то формируются сигналы побайтного разрешения записи (соответствующие разряды WREN[3:0]). Если помехоустойчивое кодирование данной области запрещено, то при вышеупомянутых обращениях на шине данных выставляется несколько экземпляров записываемых данных (их количество равно отношению разрядности внешней шины данных к разрядности обращения записи). Если активирован режим использования корректирующих кодов и разрядность обращения меньше, чем ширина используемой внешней шины данных, то перезапись осуществляется через механизм чтение-модификация-запись, поскольку для формирования корректных проверочных бит требуется полное слово данных (32 бита).

На рисунке 4.5.5 приведена векторная диаграмма, которая иллюстрирует процесс чтения-модификации-записи в область PROM без дополнительных тактов состояний.

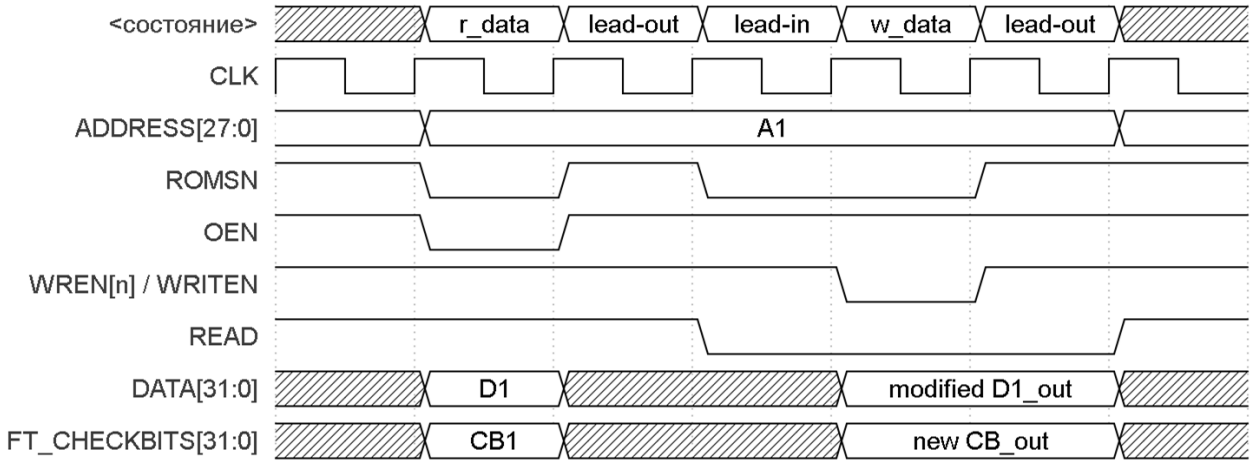


Рисунок 4.5.5 – Временная диаграмма операции чтение-модификация-запись в область PROM без дополнительных тактов

4.5.3 Доступ к области IO

Для организации доступа к области IO используется один общий сигнал выбора кристалла – IOSN. В качестве сигналов разрешения записи микросхем используется шина WREN[3:0]. Вывод WREN[0] отвечает за линии данных DATA[31:24], WREN[1] – за DATA[23:16], WREN[2] – за DATA[15:8], WREN[3] – за DATA[7:0]. Если в данную область не планируется осуществлять запись данных, разрядность которых меньше слова (32 бита), то вместо каждого из разрядов WREN можно подключить WRITEN. Активный уровень WRITEN выставляется, если хотя бы один из разрядов WREN находится в активном состоянии. Сигнал разрешения чтения – OEN.

Выходной сигнал READ сбрасывается на все время выполнения операции записи, в том числе и на стадиях lead-in и lead-out, в остальное время он остается установлен. Данный сигнал можно использовать для управления направлением буферов шины данных микросхем внешней памяти, если таковые присутствуют в устройстве.

На рисунке 4.5.6 представлена временная диаграмма одиночной записи в область IO. Следует помнить, что за наличие или отсутствие дополнительных тактов lead-in и lead-out при записи отвечают те же разряды регистров управления, что и для области PROM.

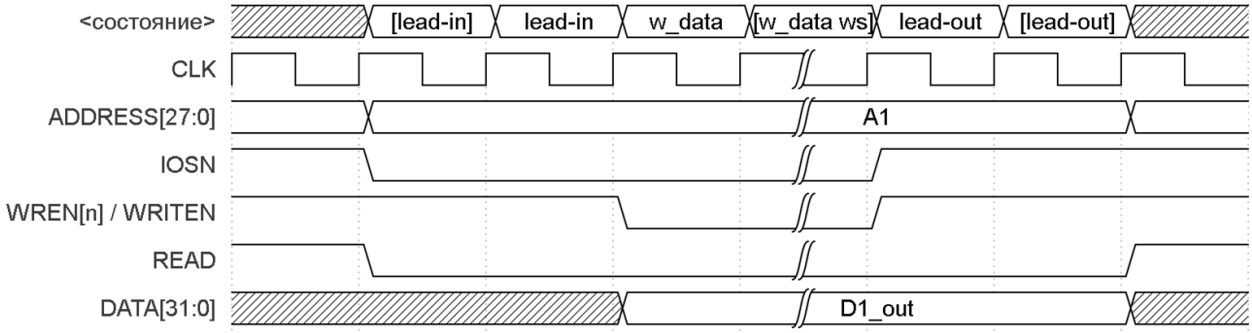


Рисунок 4.5.6 – Временная диаграмма одиночной записи в область IO

4.5.4 Управление временной диаграммой контроллера PROM / IO

Транзакции на шине контроллера PROM / IO состоят из следующих последовательностей:

[LEAD-IN] LEAD-IN WRITE [WAIT CYCLES] LEAD-OUT [LEAD-OUT]
READ [WAIT CYCLES] LEAD-OUT

– Стадия LEAD-IN активируется только для операций записи. Если бит LIP регистра управления контроллерами памяти установлен, то она длится на такт дольше (состояние по умолчанию для контроллера PROM / IO).
– В зависимости от операции записи или чтения активируется стадия READ / WRITE.
– Продолжительность предыдущей стадии варьируется в зависимости от количества дополнительных тактов ожидания. Они позволяют выбрать длительность цикла активности в зависимости от частоты тактового сигнала процессора и требований внешней памяти. Количество дополнительных циклов ожидания задается с помощью полей регистра управления тактами ожидания контроллера памяти PWWC (задержка при записи в PROM), PRWC (задержка при чтении из PROM), IWWC (задержка при записи в IO) и IRWC (задержка при чтении из IO). Количество дополнительных тактов ожидания может быть от 0 до 31. Значения по умолчанию PRWC = PWWC = 11110_2 = 30. Значения по умолчанию IRWC = IWWC = 00000_2 = 0.
– Стадия LEAD-OUT операции чтения занимает один такт. Для операций записи она продляется на один такт, если бит LOP регистра управления контроллерами памяти установлен (состояние по умолчанию для контроллера PROM / IO).

4.5.5 8-битный режим области PROM

Контроллер PROM / IO поддерживает подключение микросхем памяти к области PROM через 8-битную шину данных, за выбор активного режима отвечает внешний сигнал разрешения CFG_8B_PROM и бит PROM_8B регистра управления контроллерами памяти.

Для того чтобы перевести контроллер PROM / IO в режим использования 8-битной шины данных PROM, в процессе сброса необходимо установить внешний сигнал разрешения CFG_8B_PROM, который определяет значение по сбросу бита PROM_8B. После сброса за включение/отключение 8-битного режима доступа к PROM (в качестве шины данных используются старшие 8 бит [31:24]) отвечает только бит PROM_8B. Если бит PROM_8B установлен, используется 8-битная шина данных PROM. В противном случае используется 32-битная шина данных PROM.

Если используется помехоустойчивое кодирование, то к микросхеме, отвечающей за данный функционал, в качестве шины данных выступают младшие разряды шины FT_CHECKBITS, остальные сигналы управления совпадают с таковыми для основных микросхем PROM. При вычислении синдрома ошибки используются только FT_CHECKBITS[4:0], при выполнении операций записи старшие разряды этой шины будут иметь нулевое значение.

Схема использования 8-битного подключения микросхем памяти к области PROM приведена на рисунке 4.5.7.

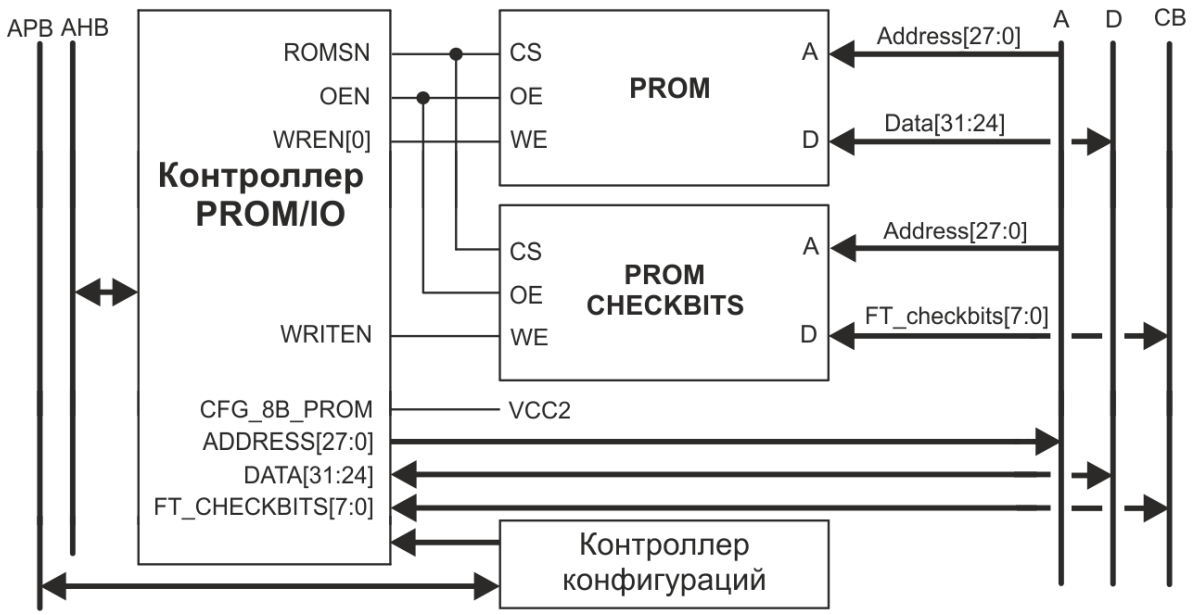


Рисунок 4.5.7 – Схема 8-битного подключения микросхем памяти к области PROM

Операции любой разрядности кроме 8-битных формируют последовательность обращений, реализующую запрос.

4.5.6 Система сигнализации о готовности

Сигнал BRDYN может быть использован для удлинения цикла доступа к областям памяти PROM и IO при выполнении операции чтения, записи и чтения-модификации-записи. Доступ по-прежнему будет иметь как минимум предустановленное значение дополнительных тактов активного состояния (см. значение соответствующих полей регистра WC_CTRL), но также может быть удлинен до тех пор, пока сигнал BRDYN не будет вновь установлен. Чтобы повлиять на продолжительность цикла доступа, сигнал BRDYN должен быть сброшен, по крайней мере, не позже такта, предшествующего первому такту lead-out. Выборка сигнала BRDYN осуществляется по положительному фронту тактового сигнала.

Для управления доступностью функционала BRDYN для областей PROM и IO используются биты PBRDY и IBRDY регистра управления контроллерами памяти (MEM_CTRL) соответственно. По сбросу для обеих областей функционал не активен. На рисунке 4.5.8 приведен пример цикла чтения с двумя дополнительными тактами задержки и одним тактом задержки, сгенерированным сигналом BRDYN.

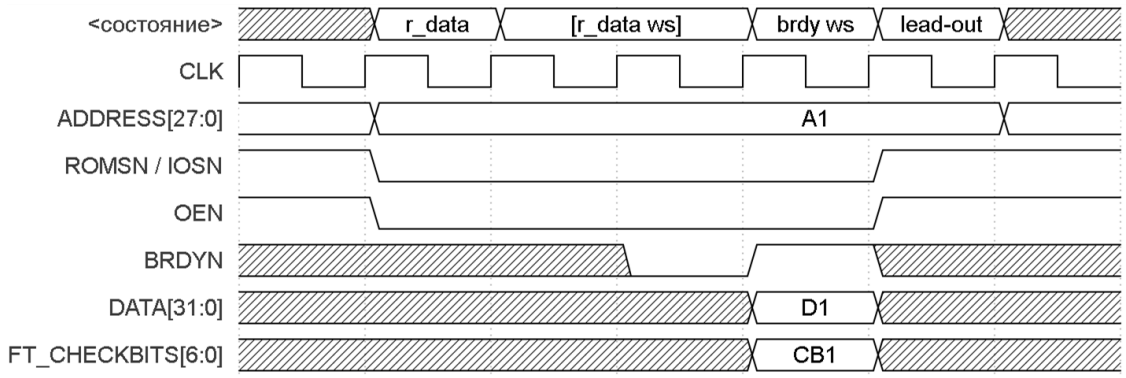


Рисунок 4.5.8 – Обращение чтения с двумя дополнительными тактами задержки и одним тактом задержки, сгенерированным сигналом BRDYN

4.5.7 Описание системы сигнализации об ошибке доступа

Поддерживается возможность сигнализации контроллеру (процессору) об ошибке доступа чтения или записи установкой внешнего сигнала BEXCN (активный уровень сигнала – 0). Для операций записи проверка состояния данного сигнала осуществляется в последний такт, перед тем как сигнал выбора кристалла станет неактивным. Если использование BEXCN разрешено в регистре управления контроллерами памяти (бит BEXCN), то при активном уровне сигнала BEXCN во время проверки будет сгенерирован сигнал ошибки на шине AHB. Для всех видов внешней памяти (PROM, IO и SRAM) используется один бит разрешения/запрещения сигнализации об ошибке доступа, по умолчанию функционал отключен, состояние внешнего вывода BEXCN игнорируется.

Важно отметить, что контроль осуществляется только в последнем из доступов транзакции по шине AHB. Из нескольких доступов состоят 64-битные обращения чтения/записи, а также все обращения в 8-битном режиме шины данных PROM кроме 8-битных.

На рисунке 4.5.9 представлена операция чтения с двумя дополнительным тактами активного состояния с активным уровнем сигнала BEXCN.

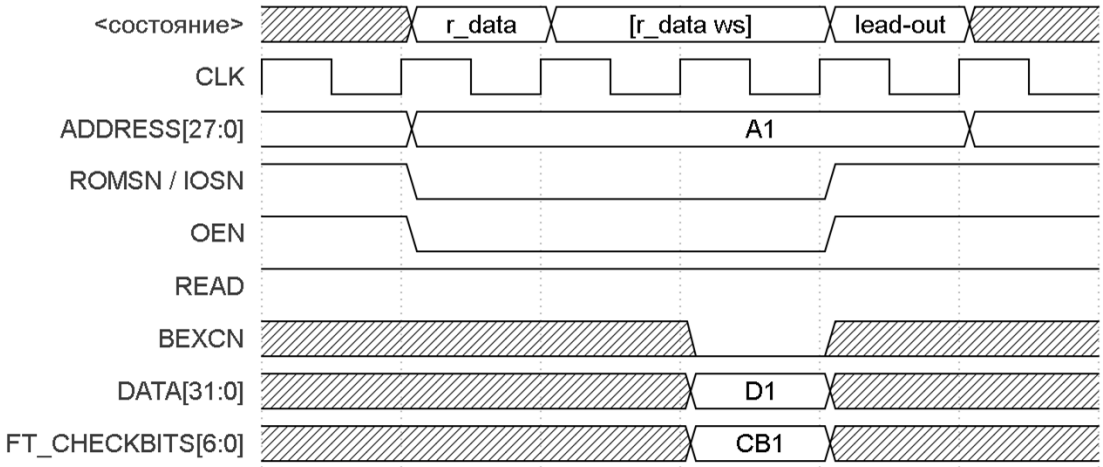


Рисунок 4.5.9 – Доступ чтения с двумя дополнительными тактами активного состояния, который вызовет ошибку на шине AHB

На рисунке 4.5.10 представлена операция записи с двумя дополнительным тактами активного состояния с активным уровнем сигнала BEXCN.

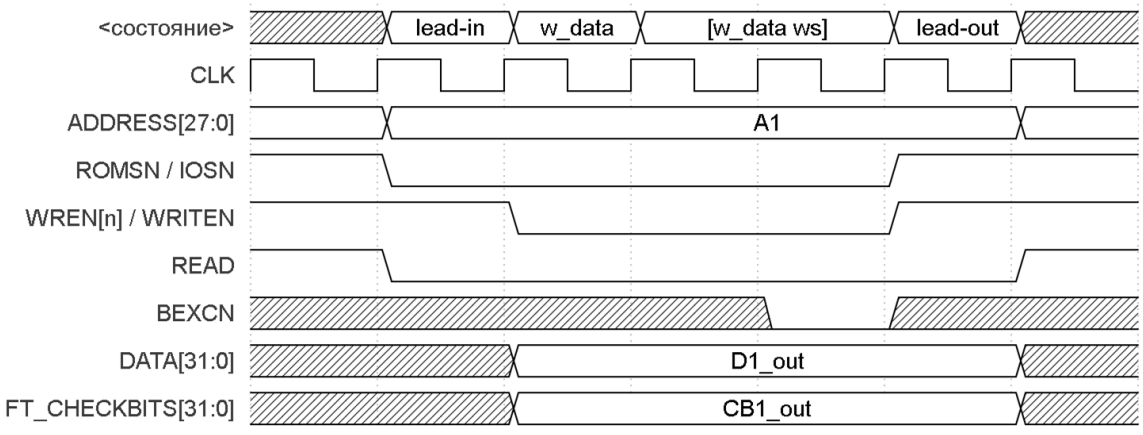


Рисунок 4.5.10 – Доступ записи с двумя дополнительными тактами активного состояния, который вызовет ошибку на шине AHB

4.5.8 Коррекция ошибок

Активация режима коррекции ошибок для контроллера PROM происходит, только если на вход CFG_FT_ENA подается высокий логический уровень сигнала и установлен бит FT_CTRL.PFT контроллера конфигураций. В противном случае контроль проверочных бит не осуществляется, при выполнении операций записи на шине FT_CHECKBITS[31:0] выставляется значение 0x00000000.

32-битный режим PROM

Если режим коррекции ошибок разрешен для контроллера PROM, то для обеспечения помехоустойчивости используется код Хэмминга(39,32). Он позволяет обнаружение до двух ошибок и исправление одной ошибки на кодовое слово.

В данном режиме внешней шины данных контроллера PROM кодовое слово состоит из исходных 32 бит данных и семь проверочных бит (восьмой бит всегда равен «0»). Каждый из них вычисляется, используя операции ИСКЛЮЧАЮЩЕГО ИЛИ над соответствующими битами данных. Ниже представлены формулы для вычисления проверочных бит:

```
CB0 = D0  ^ D1  ^ D2  ^ D3  ^ D4  ^ D5  ^ D6  ^ D7  ^ D8  ^ D9  ^ D10  ^
      D11 ^ D12 ^ D13 ^ D14 ^ D15 ^ D16 ^ D17 ^ D18 ^ D19 ^ D20 ^ D21  ^
      D22 ^ D23 ^ D24 ^ D25 ^ D26 ^ D27 ^ D28 ^ D29 ^ D30 ^ D31 ^ CB1  ^
      CB2 ^ CB3 ^ CB4 ^ CB5 ^ CB6 ^ CB7;
CB1 = D0  ^ D1  ^ D3  ^ D4  ^ D6  ^ D8  ^ D10 ^ D11 ^ D13 ^ D15 ^ D17  ^
      D19 ^ D21 ^ D23 ^ D25 ^ D26 ^ D28 ^ D30;
CB2 = D0  ^ D2  ^ D3  ^ D5  ^ D6  ^ D9  ^ D10 ^ D12 ^ D13 ^ D16 ^ D17  ^
      D20 ^ D21 ^ D24 ^ D25 ^ D27 ^ D28 ^ D31;
CB3 = D1  ^ D2  ^ D3  ^ D7  ^ D8  ^ D9  ^ D10 ^ D14 ^ D15 ^ D16 ^ D17  ^
      D22 ^ D23 ^ D24 ^ D25 ^ D29 ^ D30 ^ D31;
CB4 = D4  ^ D5  ^ D6  ^ D7  ^ D8  ^ D9  ^ D10 ^ D18 ^ D19 ^ D20 ^ D21  ^
      D22 ^ D23 ^ D24 ^ D25;
CB5 = D11 ^ D12 ^ D13 ^ D14 ^ D15 ^ D16 ^ D17 ^ D18 ^ D19 ^ D20 ^ D21  ^
      D22 ^ D23 ^ D24 ^ D25;
CB6 = D26 ^ D27 ^ D28 ^ D29 ^ D30 ^ D31;
CB7 = 0;
```

Контроллер PROM при обнаружении исправимой ошибки восстановит корректные данные и сигнализирует о произошедшем событии установкой пары битов CE и NE в регистре статуса AHB. Если выявлено больше ошибок, чем модуль в состоянии исправить, то будет установлен только бит NE. В регистрах статуса AHB отображается только транзакция, в которой произошла ошибка. Поскольку подключение контроллера к шине AHB осуществляется через 64-разрядную шину (транзакция может содержать до двух кодовых слов), данная информация может не полностью характеризовать местоположение некорректного кодового слова. Помимо отображения факта неисправимой ошибки в регистрах статуса AHB, в процессоре генерируется исключение instruction_access_error (для операций выборки инструкций), data_access_error (для запросов на считывание данных) или data_store_error (при обращениях вида чтение-модификация-запись). Появление исключения приведет к переходу в таблицу прерываний (помимо выделения нового регистрового окна и сохранения значений PC и nPC в его регистры %l1 и %l2) или к переводу процессора в состояние ошибки, в зависимости от значения бита ET регистра %psr.

Если установлен бит FT_CTRL.WBP (по сбросу данный режим не активен для области PROM) при обнаружении исправимой ошибки контроллер автоматически

перезапишет исправленное кодовое слово в память (если в качестве PROM используется MRAM или её аналог). Если транзакция по шине AHB состоит из нескольких кодовых слов и не содержит неисправимых ошибок, то осуществляется перезапись только тех слов, в которых была обнаружена исправимая ошибка. Если транзакция содержит хотя бы одну неисправимую ошибку, то в качестве ответа выставляется ошибка AHB и никаких дополнительных действий с внешней памятью не выполняется. Важно понимать, что механизм автоматической перезаписи исправленных данных не рассчитан на использование Flash-памяти в качестве PROM, поскольку для реализации такого функционала требуется предварительное стирание отдельного слова и использование специфической для микросхемы последовательности записи.

8-битный режим PROM

Если активирован 8-битный режим PROM, то каждому 8-битному слову данных соответствует пять проверочных бит (старшие биты всегда равны «0»). Каждый из них вычисляется, используя операции ИСКЛЮЧАЮЩЕГО ИЛИ над соответствующими битами данных. Такой подход позволяет обнаруживать до двух ошибок и исправлять одну ошибку на кодовое слово (оно состоит из исходных 8 бит данных и 5 бит проверочных). Ниже представлены формулы для вычисления проверочных бит (схема их формирования аналогична 32-битной, если подставить в неё D31–D8 равные «0»). Стоит отметить, что битам D7–D0 в формуле обозначены соответствующие разряды данных с шины Data[31:24] (то есть D7 = Data[31], D6 = Data[30] и так далее):

```
CB0 = D0  ^ D1  ^ D2  ^ D3  ^ D4  ^ D5  ^ D6  ^ D7  ^ CB1 ^ CB2 ^ CB3  ^
      CB4 ^ CB5 ^ CB6 ^ CB7;
CB1 = D0  ^ D1  ^ D3  ^ D4  ^ D6;
CB2 = D0  ^ D2  ^ D3  ^ D5  ^ D6;
CB3 = D1  ^ D2  ^ D3  ^ D7;
CB4 = D4  ^ D5  ^ D6  ^ D7;
CB5 = 0;
CB6 = 0;
CB7 = 0;
```

Сигнализация в регистрах статуса AHB о произошедшей ошибке, также как и в 32-битном режиме внешней шины данных, привязана к осуществляемой транзакции. Поскольку подключение контроллера к шине AHB осуществляется через 64-разрядную шину (транзакция может содержать до восьми кодовых слов), данная информация может не полностью характеризовать местоположение некорректного кодового слова. Аналогично режиму с 32-х битной внешней шиной данных при обнаружении неисправимой ошибки в процессоре генерируется исключение data_access_error или data_store_error.

Описание функционала автоматической перезаписи исправленных кодовых слов полностью аналогично таковому в подпункте «32-битный режим PROM».

4.5.9 Регистры

Регистры управления контроллерами памяти PROM / IO, SRAM, SDRAM вынесены в модуль контроллера конфигураций (их непосредственное описание приведено в пункте «4.3.4 Регистры»).

4.6 Контроллер SRAM

4.6.1 Общие сведения

Контроллер SRAM реализует мост между внешней памятью и шиной AHB. Память SRAM может быть обеспечена средствами обнаружения и исправления ошибок кодом Хэмминга (39,32). Код Хэмминга обеспечивает обнаружение до двух ошибок и исправление одной ошибки на кодовое слово.

Структурная схема контроллера SRAM приведена на рисунке 4.6.1.

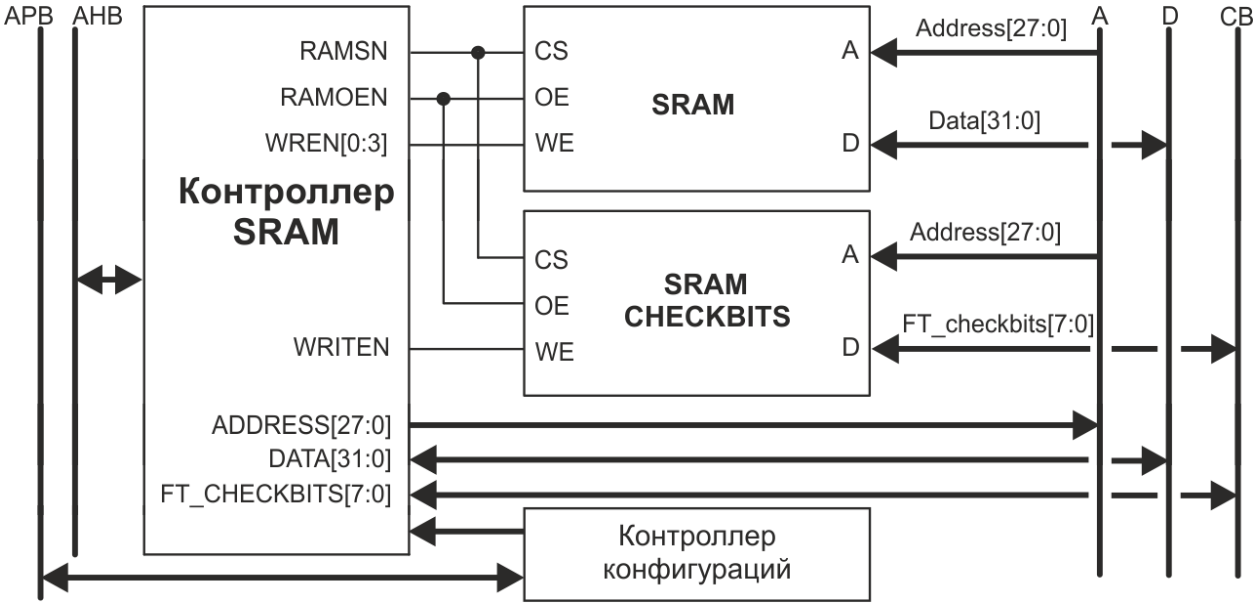


Рисунок 4.6.1 – Структурная схема контроллера SRAM

Временными параметрами функционирования и помехоустойчивого кодирования контроллера SRAM можно управлять с помощью регистров, описанных в пункте «4.3.4 Регистры» контроллера конфигураций. Контроллер отвечает за одно адресное пространство (SRAM). Контроллер SRAM конкурирует с контроллером SDRAM за диапазон адресов 0x40000000 – 0x4FFFFFFF. Одновременная работа двух вышеперечисленных контроллеров не поддерживается, за выбор активного отвечает бит RS регистра управления режимами помехоустойчивого кодирования. Область RAM (SRAM или SDRAM), отображаемая в диапазоне адресов 0x40000000 – 0x7FFFFFFF, является кэшируемой с упреждающей выборкой инструкций, что отображено в plug & play конфигурации шины AHB.

4.6.2 Доступ к области SRAM

Для организации доступа к области SRAM используется один общий сигнал выбора кристалла – RAMSN. В качестве сигналов разрешения записи микросхем используется шина WREN[3:0]. Вывод WREN[0] отвечает за линии данных DATA[31:24], WREN[1] за DATA[23:16], WREN[2] за DATA[15:8], WREN[3] за DATA[7:0]. Блок проверочных бит в качестве сигнала разрешения записи использует вывод WRITEN. Активный уровень WRITEN выставляется, если хотя бы один из разрядов WREN находится в активном состоянии. Если область SRAM планируется использовать только с активированным помехоустойчивым кодированием, то вместо каждого из разрядов WREN можно подключить WRITEN. Это допустимо, поскольку в таком случае все обращения

записи с разрядностью меньшей ширины используемой внешней шины данных будут обрабатываться через алгоритм чтения-модификации-записи. Сигнал разрешения чтения – RAMOEN.

Выходной сигнал READ сбрасывается на все время выполнения операции записи, в том числе и на стадиях lead-in и lead-out, в остальное время он остается установлен. Данный сигнал можно использовать для управления направлением буферов шины данных микросхем внешней памяти, если таковые присутствуют в устройстве.

Когда активировано помехоустойчивое кодирование, при 8- или 16-битной записи используется последовательность чтение-модификация-запись, поскольку проверочные биты формируются исходя из полного 39-битного кодового слова (см. пункт «4.6.6 Коррекция ошибок»).

Если используется помехоустойчивое кодирование, то к микросхеме, отвечающей за данный функционал, в качестве сигнала разрешения записи подключается вывод WRITEN, в качестве шины данных выступают младшие разряды шины FT_CHECKBITS, остальные сигналы управления совпадают с таковыми для основных микросхем SRAM. При вычислении синдрома ошибки используются только FT_CHECKBITS[6:0], при выполнении операций записи старшие разряды этой шины будут иметь нулевое значение.

На рисунке 4.6.2 представлена временная диаграмма одиночной записи в область SRAM (со всеми возможными дополнительными тактами). Более подробная информация о конфигурировании временной диаграммы контроллера представлена в пункте «4.6.3 Управление временной диаграммой контроллера SRAM».

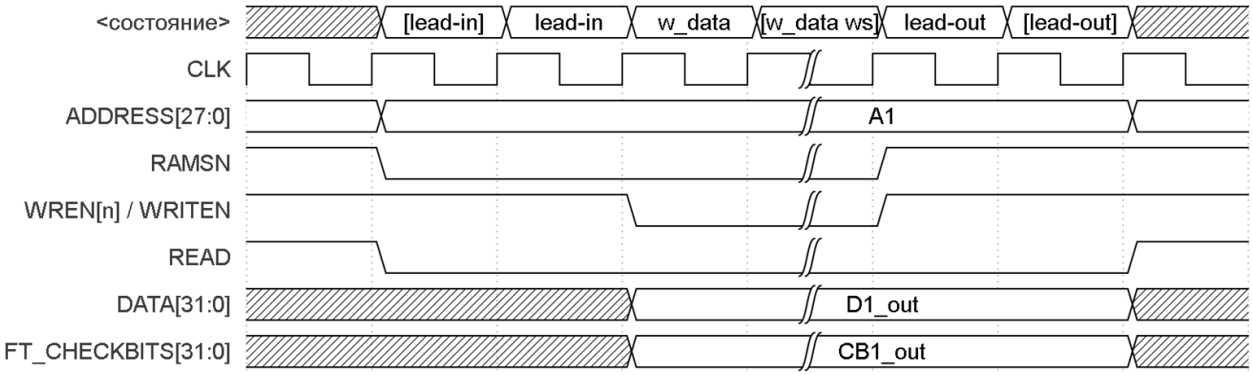


Рисунок 4.6.2 – Временная диаграмма одиночной записи в область SRAM

На рисунке 4.6.3 приведена временная диаграмма чтения из области SRAM с дополнительными тактами активного состояния.

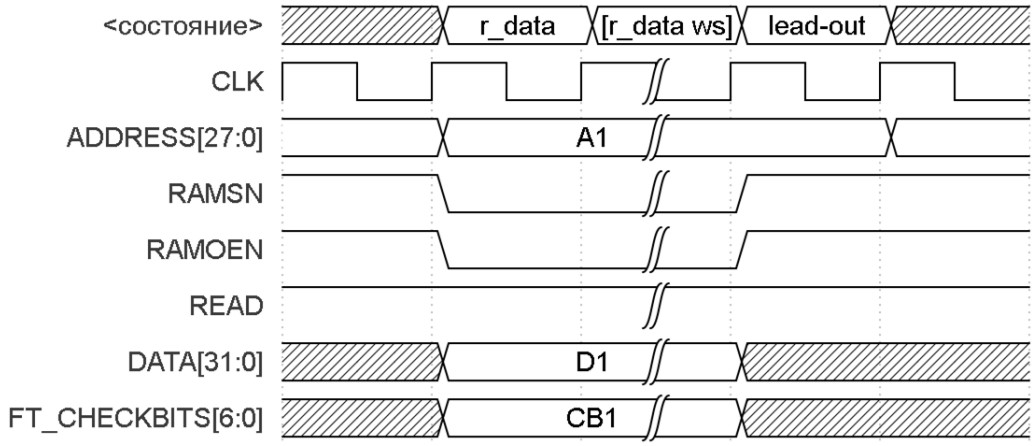


Рисунок 4.6.3 – Временная диаграмма чтения из области SRAM с дополнительными тактами активного состояния

Если разрядность обращения превышает ширину используемой внешней шины данных (64-битные выборки инструкций или доступы к данным из области SRAM, записи двойного слова в эту область), то контроллер формирует последовательность обращений, реализующую запрос. На рисунке 4.6.4 приведен пример выполнения 64-битного доступа записи в SRAM без каких-либо дополнительных тактов.

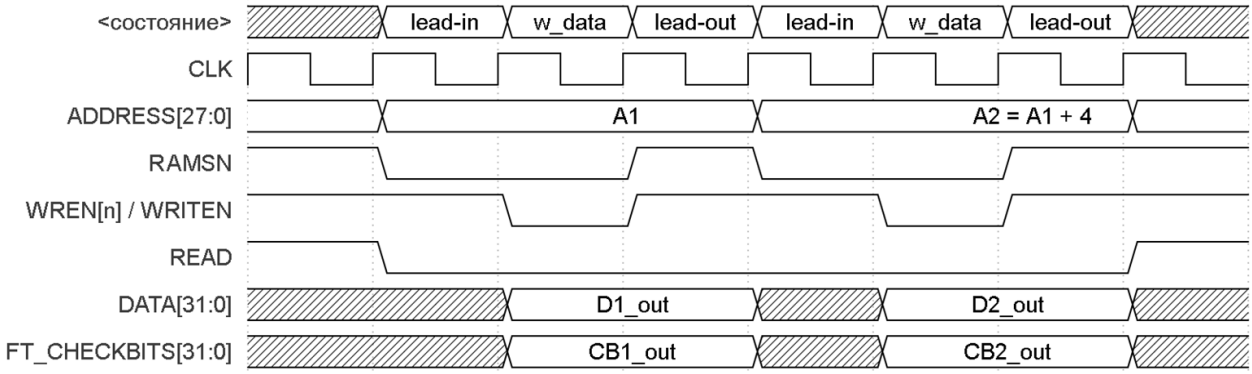


Рисунок 4.6.4 – Временная диаграмма 64-х битной записи в область SRAM без дополнительных тактов

Если разрядность обращения меньше чем ширина используемой внешней шины данных, то формируются сигналы побайтного разрешения записи (соответствующие разряды WREN[3:0]). Если помехоустойчивое кодирование данной области запрещено, то при вышеупомянутых обращениях на шине данных выставляется несколько экземпляров записываемых данных (их количество равно отношению разрядности внешней шины данных к разрядности обращения записи). Если активирован режим использования корректирующих кодов и разрядность обращения меньше чем ширина используемой внешней шины данных, то запись осуществляется через механизм чтение-модификация-запись, поскольку для формирования корректных проверочных бит требуется полное слово данных (32 бита).

На рисунке 4.6.5 приведена векторная диаграмма, которая иллюстрирует процесс чтения-модификации-записи в область SRAM с одним дополнительным тактом активного состояния чтения.

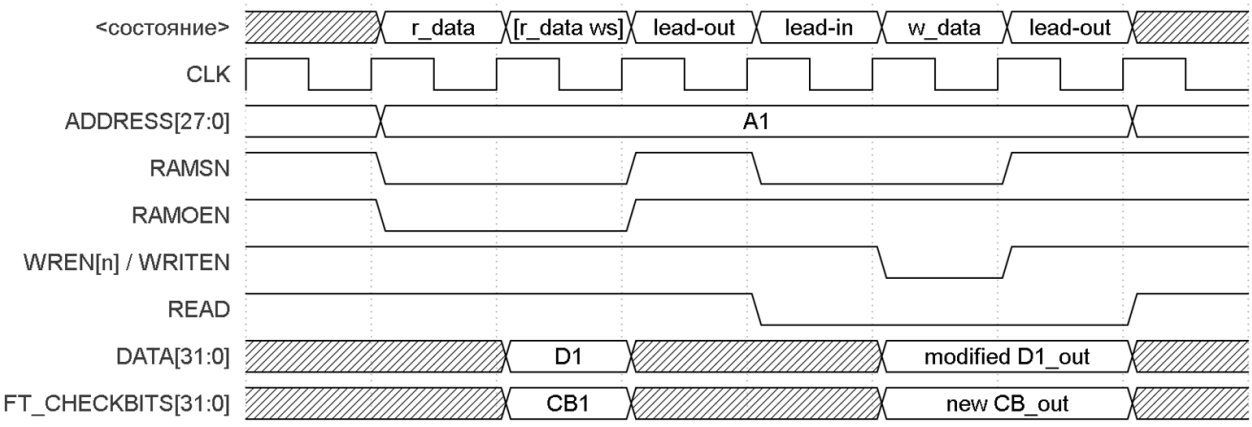


Рисунок 4.6.5 – Временная диаграмма операции чтение-модификация-запись в область SRAM с дополнительным тактом активного состояния чтения

4.6.3 Управление временной диаграммой контроллера SRAM

Транзакции на шине контроллера SRAM состоят из следующих последовательностей:

[LEAD-IN] LEAD-IN WRITE [WAIT CYCLES] LEAD-OUT [LEAD-OUT]
READ [WAIT CYCLES] LEAD-OUT

– Стадия LEAD-IN активируется только для операций записи. Если бит LIS регистра управления контроллерами памяти установлен, то она длится на такт дольше (по умолчанию данный бит сброшен для контроллера SRAM).
– В зависимости от операции записи или чтения активируется стадия READ / WRITE.
– Продолжительность предыдущей стадии варьируется в зависимости от количества дополнительных тактов ожидания. Они позволяют выбрать длительность цикла активности в зависимости от частоты тактового сигнала процессора и требований внешней памяти. Количество дополнительных циклов ожидания задается с помощью полей регистра управления тактами ожидания контроллера памяти SWWC (задержка при записи в SRAM), SRWC (задержка при чтении из SRAM). Количество дополнительных тактов ожидания может быть от 0 до 31. Значения по умолчанию SRWC = SWWC = 00000_2 = 0.
– Стадия LEAD-OUT операции чтения занимает один такт. Для операции записи она продляется на один такт, если бит LOS регистра управления контроллерами памяти установлен (по умолчанию данный бит сброшен для контроллера SRAM).

4.6.4 Система сигнализации о готовности

Сигнал BRDYN может быть использован для удлинения цикла доступа к области SRAM, включая операции чтения, записи и чтения-модификации-записи. Доступ по-прежнему будет иметь как минимум предустановленное значение дополнительных тактов ожидания (см. значение соответствующих полей регистра WC_CTRL), но также может быть удлинен до тех пор, пока сигнал BRDYN не будет вновь установлен. Чтобы повлиять на продолжительность цикла доступа, сигнал BRDYN должен быть сброшен, по крайней мере, не позже такта, предшествующего первому такту lead-out. Выборка сигнала BRDYN осуществляется по положительному фронту тактового сигнала.

Для управления доступностью функционала BRDYN для области SRAM используется бит SBRDY регистра управления контроллерами памяти (MEM_CTRL). По сбросу функционал не активен.

На рисунке 4.6.6 представлен пример обращения записи с двумя дополнительными тактами активного состояния, заданными значением поля SWWC, и одним дополнительным тактом, сгенерированным благодаря наличию активного уровня сигнала BRDYN.

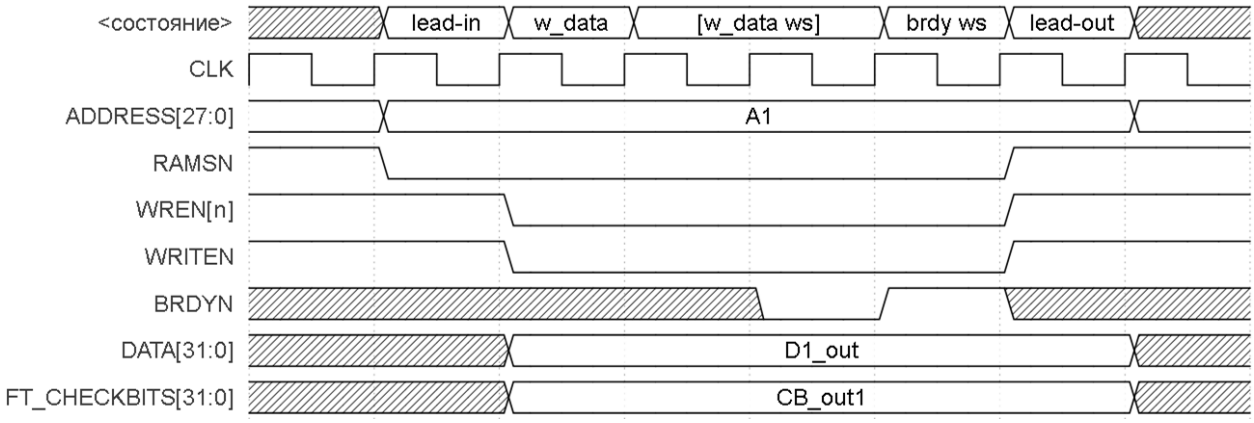


Рисунок 4.6.6 – Обращение записи с двумя дополнительными тактами активного состояния и одним дополнительным тактом, сгенерированным сигналом BRDYN

На рисунке 4.6.7 приведен пример цикла чтения с минимальным значением поля SRWC регистра WC_CTRL и тремя тактами активного состояния, сгенерированными сигналом BRDYN. Для организации подобного поведения при заявленном значении поля SRWC требуется, чтобы активный уровень сигнала BRDYN был выставлен в тот же такт, что и активный уровень сигнала выбора кристалла (RAMSN).

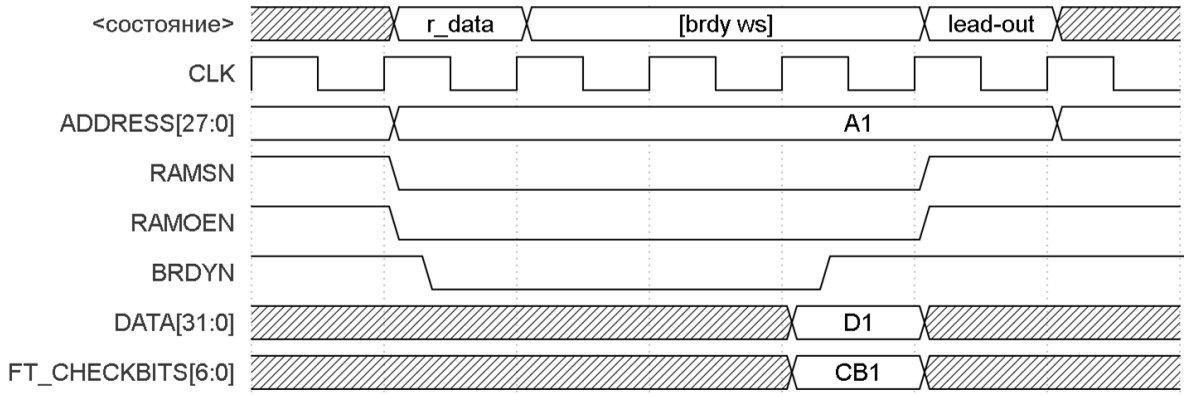


Рисунок 4.6.7 – Цикл чтения с тремя дополнительными тактам активного состояния, сгенерированным сигналом BRDYN

4.6.5 Описание системы сигнализации об ошибке доступа

Поддерживается возможность сигнализации контроллеру (процессору) об ошибке доступа чтения или записи установкой внешнего сигнала BEXCN (активный уровень сигнала – 0). Для операций записи проверка состояния данного сигнала осуществляется в последний такт, перед тем как сигнал выбора кристалла станет неактивным. Если использование BEXCN разрешено в регистре управления контроллерами памяти (бит BEXCN), то при активном уровне сигнала BEXCN во время проверки будет сгенерирован сигнал ошибки на шине AHB. Для всех видов внешней памяти (PROM, IO и SRAM) используется один бит разрешения/запрещения сигнализации об ошибке доступа, по умолчанию функционал отключен, состояние внешнего вывода BEXCN игнорируется.

Важно отметить, что контроль осуществляется только в последнем из доступов транзакции по шине AHB. Из нескольких доступов состоят 64-битные обращения чтения/записи, а также все обращения в 8-битном режиме шины данных PROM кроме 8-битных.

На рисунках 4.6.8 и 4.6.9 представлены операции чтения и записи (в качестве примера приведены временные диаграммы с двумя дополнительным тактами активного состояния доступа) и с активным уровнем сигнала BEXCN.

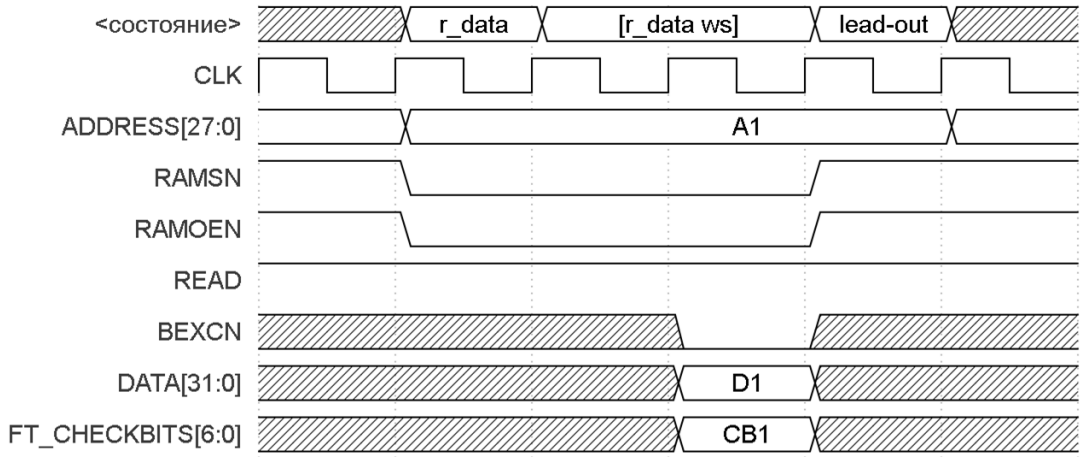


Рисунок 4.6.8 – Доступ чтения с двумя дополнительными тактами активного состояния и с активным уровнем сигнала BEXCN

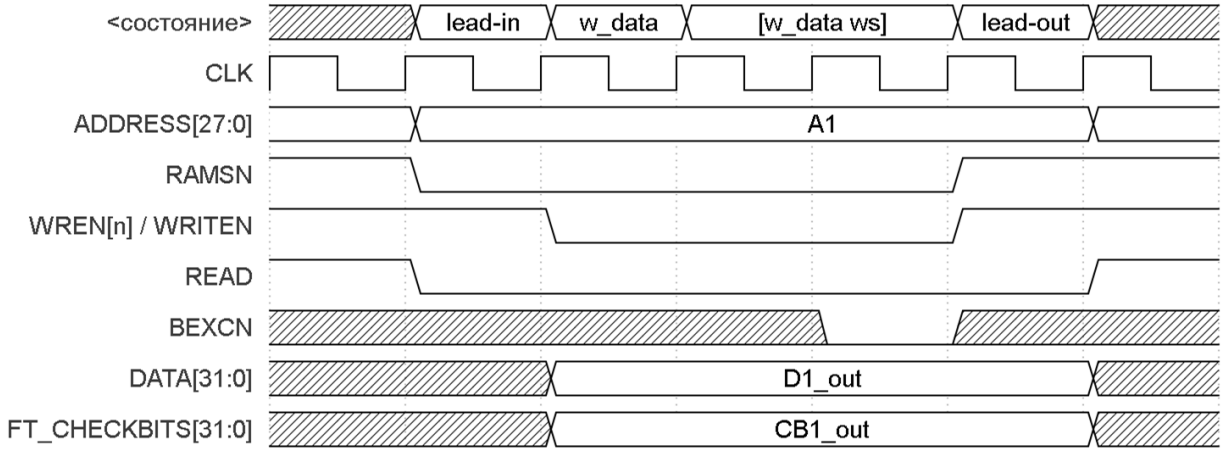


Рисунок 4.6.9 – Доступ записи с двумя дополнительными тактами активного состояния и с активным уровнем сигнала BEXCN

4.6.6 Коррекция ошибок

Активация режима коррекции ошибок для контроллера SRAM происходит, только если на вход CFG_FT_ENA подается высокий логический уровень сигнала и установлен бит FT_CTRL.SFT контроллера конфигураций. В противном случае контроль проверочных бит не осуществляется, при выполнении операций записи на шине FT_CHECKBITS[31:0] выставляется значение 0x00000000. Если режим коррекции ошибок разрешен для контроллера SRAM, то для обеспечения помехоустойчивости используется код Хэмминга(39,32). Он позволяет обнаружение до двух ошибок и исправление одной ошибки на кодовое слово.

В контроллере SRAM кодовое слово состоит из исходных 32 бит данных и 7 проверочных бит (восьмой бит всегда равен «0»). Каждый из них вычисляется, используя операции ИСКЛЮЧАЮЩЕГО ИЛИ над соответствующими битами данных. Ниже представлены формулы для вычисления проверочных бит:

```
CB0 = D0  ^ D1  ^ D2  ^ D3  ^ D4  ^ D5  ^ D6  ^ D7  ^ D8  ^ D9  ^ D10  ^
      D11 ^ D12 ^ D13 ^ D14 ^ D15 ^ D16 ^ D17 ^ D18 ^ D19 ^ D20 ^ D21  ^
      D22 ^ D23 ^ D24 ^ D25 ^ D26 ^ D27 ^ D28 ^ D29 ^ D30 ^ D31 ^ CB1  ^
      CB2 ^ CB3 ^ CB4 ^ CB5 ^ CB6 ^ CB7;
CB1 = D0  ^ D1  ^ D3  ^ D4  ^ D6  ^ D8  ^ D10 ^ D11 ^ D13 ^ D15 ^ D17  ^
      D19 ^ D21 ^ D23 ^ D25 ^ D26 ^ D28 ^ D30;
CB2 = D0  ^ D2  ^ D3  ^ D5  ^ D6  ^ D9  ^ D10 ^ D12 ^ D13 ^ D16 ^ D17  ^
      D20 ^ D21 ^ D24 ^ D25 ^ D27 ^ D28 ^ D31;
CB3 = D1  ^ D2  ^ D3  ^ D7  ^ D8  ^ D9  ^ D10 ^ D14 ^ D15 ^ D16 ^ D17  ^
      D22 ^ D23 ^ D24 ^ D25 ^ D29 ^ D30 ^ D31;
CB4 = D4  ^ D5  ^ D6  ^ D7  ^ D8  ^ D9  ^ D10 ^ D18 ^ D19 ^ D20 ^ D21  ^
      D22 ^ D23 ^ D24 ^ D25;
CB5 = D11 ^ D12 ^ D13 ^ D14 ^ D15 ^ D16 ^ D17 ^ D18 ^ D19 ^ D20 ^ D21  ^
      D22 ^ D23 ^ D24 ^ D25;
CB6 = D26 ^ D27 ^ D28 ^ D29 ^ D30 ^ D31;
CB7 = 0;
```

Контроллер SRAM при обнаружении исправимой ошибки восстановит корректные данные и сигнализирует о произошедшем событии установкой бита CE в регистре статуса AHB (см. таблицу 4.12.2). Если выявлено больше ошибок, чем модуль в состоянии исправить, то будет установлен только бит NE. В регистрах статуса AHB отображается только транзакция, в которой произошла ошибка. Поскольку подключение контроллера к шине AHB осуществляется через 64-разрядную шину (транзакция может содержать до двух кодовых слов), данная информация может не полностью характеризовать местоположение некорректного кодового слова. Помимо отображения факта неисправимой ошибки в регистрах статуса AHB, в процессоре генерируется исключение instruction_access_error (для операций выборки инструкций), data_access_error (для запросов на считывание данных) или data_store_error (при обращениях вида чтение-модификация-запись). Появление исключения приведет к переходу в таблицу прерываний (помимо выделения нового регистрового окна и сохранения значений PC и nPC в его регистры %l1 и %l2) или к переводу процессора в состояние ошибки, в зависимости от значения бита ET регистра %psr.

Если установлен бит FT_CTRL.WBS (по сбросу данный режим активен для области SRAM), при обнаружении исправимой ошибки контроллер автоматически перезапишет исправленное кодовое слово в память. Если транзакция по шине AHB состоит из нескольких кодовых слов и не содержит неисправимых ошибок, то осуществляется перезапись только тех слов, в которых была обнаружена исправимая ошибка. Если транзакция содержит хотя бы одну неисправимую ошибку, то в качестве ответа выставляется ошибка AHB и никаких дополнительных действий с внешней памятью не выполняется.

4.6.7 Регистры

Регистры управления контроллерами памяти PROM / IO, SRAM, SDRAM вынесены в модуль контроллера конфигураций (их непосредственное описание приведено в пункте «4.3.4 Регистры»).

4.7 Контроллер SDRAM

4.7.1 Общие сведения

Контроллер SDRAM реализует мост между внешней памятью и шиной AHB. Память SDRAM может быть обеспечена средствами обнаружения и исправления ошибок кодом Хэмминга (72,64) или кодом Рида-Соломона (96,64). Код Хэмминга обеспечивает обнаружение до двух ошибок и исправление одной ошибки на кодовое слово, реализована полностью комбинаторная схема. Код Рида-Соломона в состоянии исправить до двух 8-ми битных блоков ошибок на 64 бит данных и 32 проверочных бита, (от двух до шестнадцати некорректных бит при условии, что они локализованы в двух байтах).

Структурная схема контроллера SDRAM приведена на рисунке 4.7.1. Приведен вариант с подключением микросхем(ы) для хранения проверочных бит, который позволит реализовать любой из вариантов помехоустойчивого кодирования. Если планируется использовать только код Хэмминга (72,64), то для каждого блока памяти будет достаточно подключения 8-битной микросхемы в качестве SDRAM CHECKBITS. На входы DQM микросхем проверочных бит следует подавать низкий уровень логического сигнала, остальные сигналы управления совпадают с таковыми для основных микросхем.

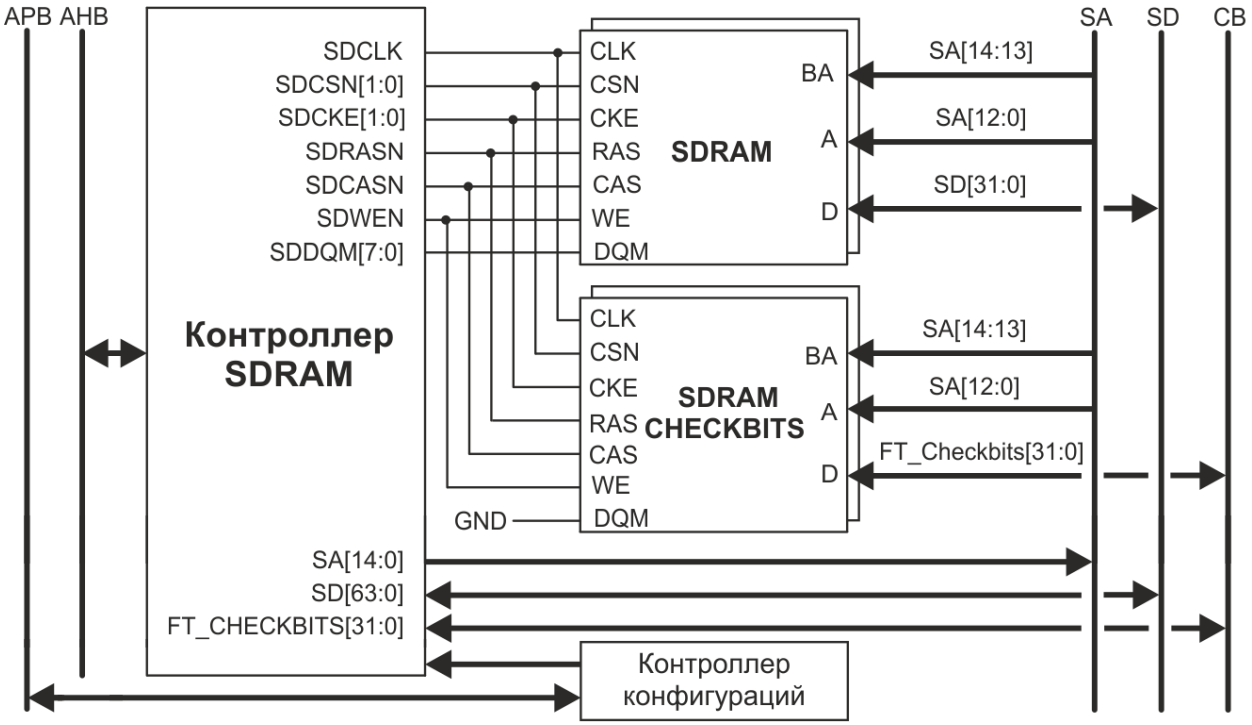


Рисунок 4.7.1 – Структурная схема контроллера SDRAM

Временными параметрами функционирования и помехоустойчивого кодирования контроллера SDRAM можно управлять с помощью регистров контроллера конфигураций, описанных в пункте «4.3.4 Регистры». Контроллер отвечает за одно адресное пространство (SDRAM) – 0x40000000 – 0x7FFFFFFF. Контроллер SDRAM конкурирует с контроллером SRAM за диапазон адресов 0x40000000 – 0x4FFFFFFF. Одновременная работа двух вышеперечисленных контроллеров не поддерживается. По умолчанию активирован контроллер SRAM. Для активации контроллера SDRAM требуется установить бит RS регистра FT_CTRL и осуществить запись в регистр SDRAM_CFG_1 (см. пункт «4.7.5»).Область RAM (SRAM или SDRAM), отображаемая в диапазоне адресов 0x40000000 – 0x7FFFFFFF, является кэшируемой с упреждающей выборкой инструкций, что отображено в plug & play конфигурации шины AHB.

4.7.2 Доступ к области SDRAM

Для организации доступа к области SDRAM используется ряд управляющих сигналов и шин:

SDCKE – сигналы разрешений тактирования. При низком уровне сигнала блокируется подача тактового сигнала на микросхему. Команды не обрабатываются, состояние других командных линий игнорируется.

SDCSN – сигналы обращения к микросхемам. При высоком уровне сигнала все прочие управляющие линии, кроме SDCKE, игнорируются. Действует как команда NOP (нет оператора).

SDDQM – шина маски данных. Высокий уровень на линии данной шины запрещает чтение/запись соответствующего байта данных. Вывод SDDQM[7] отвечает за линии данных SD[63:56], SDDQM[6] за SD[55:48], SDDQM[5] за SD[47:40], SDDQM[4] за SD[39:32], SDDQM[3] за SD[31:24], SDDQM[2] за SD[23:16], SDDQM[1] за SD[15:8], SDDQM[0] за SD[7:0].

SDRASN, SDCASN, SDWEN – сигналы строба адреса ряда (строки), строба адреса столбца и разрешения записи соответственно. Вместе они кодируют одну из 8 команд SDRAM.

SA и SD являются шинами адреса и данных контроллера. Старшие два разряда SA (SA[14:13]) отвечают за выбор банка данных. Контроллер SDRAM поддерживает только микросхемы с четырьмя банками данных.

К контроллеру SDRAM можно подключить до двух блоков памяти, в качестве сигналов выбора микросхемы и сигнала разрешения тактирования для второго блока памяти следует использовать выводы SDCSN[1], SDCKE[1], остальные сигналы являются общими.

Если разрядность обращения записи меньше чем ширина используемой шины данных, то активируются только требуемые разряды SDDQM[7:0]. Если разрешен режим использования корректирующих кодов и разрядность обращения меньше чем ширина используемой шины данных, то перезапись осуществляется через механизм чтение-модификация-запись, поскольку для формирования корректных проверочных битов требуется полное двойное слово (64 бита).

Выходной сигнал READ сбрасывается на все время выполнения операции записи, в остальное время он остается установлен. Данный сигнал можно использовать для управления направлением буферов шины данных микросхем внешней памяти, если таковые присутствуют в устройстве.

4.7.3 Тактирование синхронной динамической памяти

Для корректной работы контроллера при использовании синхронной динамической памяти тактовый сигнал SDCLK, приходящий на микросхемы памяти, должен опережать CLK приблизительно на 2 нс.

В обобщенном виде требуется выполнение следующих условий:

1. Соблюдение режима подачи управляющих сигналов относительно положительного фронта SDCLK.
2. Момент захвата данных контроллером должен быть синхронизирован с периодом гарантированного удержания корректных данных на шине.

В качестве примера в дальнейшем будут рассматриваться временные параметры микросхем MT48LC16M16A2.

На пределы работоспособности могут оказывать влияние следующие параметры микросхем SDRAM (все параметры приводятся с привязкой к положительному фронту тактового сигнала SDCLK):

- t_{CMS} – время установки значений CS#, RAS#, CAS#, WE#, DQM перед фронтом SDCLK;
- t_{CMH} – время удержания сигналов CS#, RAS#, CAS#, WE#, DQM после фронта SDCLK;
- t_{LZ} – время, в течение которого выходной буфер шины данных остается в состоянии высокого импеданса;
- t_{AC} – время доступа относительно положительного фронта SDCLK;
- t_{OH} – время удержания корректных данных на шине DQ.

На выполнение первого из условий напрямую влияют параметры t_{CMS} и t_{CMH}. Второе условие находится в зависимости от значения t_{OH}.

На рисунке 4.7.2 представлена временная диаграмма обращения чтения в микросхемы SDRAM для обоих поддерживаемых значений CAS Latency (2 и 3). На ней приведены упомянутые выше временные параметры.

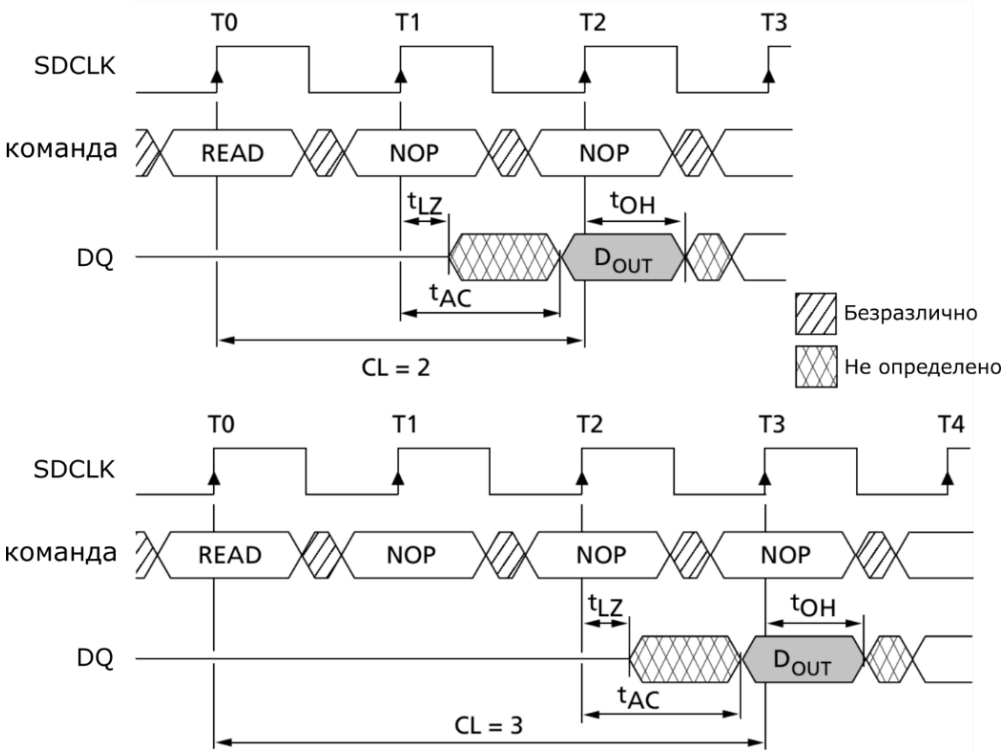


Рисунок 4.7.2 – Временные параметры сигналов микросхем SDRAM относительно SDCLK при обращении чтения (CAS Latency 2 и 3)

На рисунке 4.7.3 приведено положение фронтов тактовых сигналов CLK и SDCLK относительно друг друга.

Положительный фронт SDCLK должен приходиться на момент времени, когда все управляющие выходы SDRAM находятся в стабильном состоянии не менее t_{CMS} нс. После положительного фронта SDCLK сигналы CS#, RAS#, CAS#, WE#, DQM должны удерживаться не менее t_{CMH} нс. С другой стороны, взаиморасположение CLK и SDCLK определяет параметр t_{OH} (время удержания корректных данных на шине).

В техническом описании на выше упомянутые микросхемы указано, что, $t_{CMS} \geq 1{,}5$ нс, $t_{CMH} \geq 0{,}8$ нс, $t_{LZ} \geq 1{,}0$ нс, $t_{AC} \leq 5{,}4$ нс, $t_{OH} \geq 3{,}0$ нс.

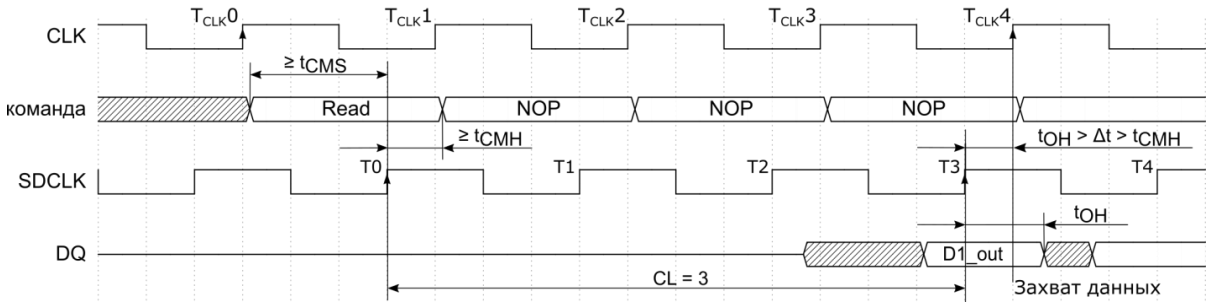


Рисунок 4.7.3 – Взаимное положение тактовых сигналов CLK и SDCLK при обращении чтения (CAS Latency = 3)

Захват данных, в случае CAS Latency равном 3, осуществляется контроллером в момент времени $T_{CLK}4$ (для CAS Latency = 2, в такт $T_{CLK}3$). Он должен приходиться на период удержания микросхемами корректных данных на шине DQ (t_{OH}). С учетом приведенных значений временных параметров схем MT48LC16M16A2, итоговая задержка CLK относительно SDCLK должна лежать в пределах от 3 до 0,8 нс. Поскольку управляющие сигналы SDRAM формируются по положительному фронту тактового сигнала CLK с некоторой задержкой, реальная правая граница рабочего диапазона может быть несколько меньше.

4.7.4 Блок управления трактом задержки SDCLK

Положительный фронт сигнала тактирования на микросхемы SDRAM должен приходить с опережением порядка 2 нс (в зависимости от параметров используемых чипов) относительно положительного фронта CLK. В процессоре выходной сигнал SDCLK является производным от входного сигнала CLK, поэтому данное соотношение не будет выдерживаться без использования дополнительной внешней или внутренней схемы манипулирования тактовым сигналом.

Как один из возможных вариантов организации требуемого взаимного расположения фронтов CLK и SDCLK на заданной частоте можно использовать блок управления трактом задержки SDCLK. Его можно использовать для программной подстройки фронта тактового сигнала, подаваемого на микросхемы SDRAM памяти. Схема блока приведена на рисунке 4.7.4, все конфигурирование осуществляется через регистр SDCLK_CTRL контроллера конфигураций.

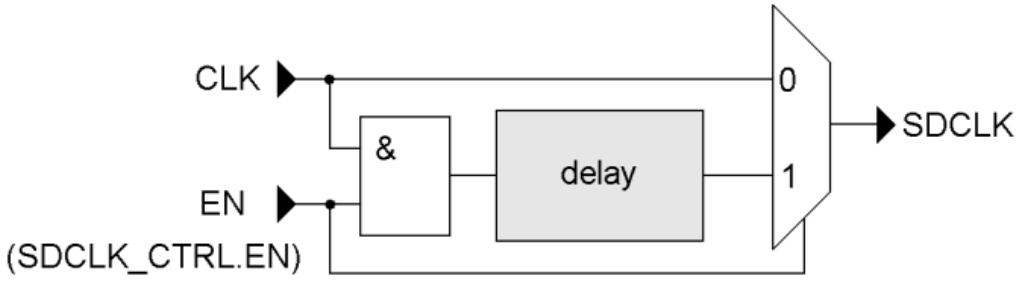


Рисунок 4.7.4 – Схема организации управляемого тракта задержки сигнала SDCLK

Регистр управления трактом задержки SDCLK (SDCLK_CTRL) состоит из следующих бит:

1. EN – Если бит установлен, то активирована ветвь с управляемым трактом задержки SDCLK. В противном случае на данный тракт не подается тактирование;

2. SDCI – Если бит установлен, то выходной сигнал тракта управляемой задержки будет инвертирован перед тем как попадет наружу;
3. SDCD – В зависимости от значения поля определяется путь, по которому будет следовать сигнал в блоке, обозначенном «delay». При нулевом значении поля обеспечивается минимальная задержка (без повторителей). При увеличении SDCD увеличивается задержка SDCLK относительно CLK с шагом близким к линейному, за счет большего количества повторителей задействованных в тракте. Диапазон доступных значений SDCD от 0 до 255 позволяет изменять задержку от 1 до 32 нс (задержку порядка 1 нс дает сам выбор более длинного плеча тракта управляемой задержки через установку бита EN).

В состав схемы входят 255 элементов задержки (на рисунке 4.7.5 обозначены квадратами), 4 уровня четырехвходовых мультиплексоров (1, 4, 16, 64 штуки на уровень соответственно), инвертор, мультиплексоры, определяющие выбор пути распространения сигнала в зависимости от значения битов управления EN и SDCI.

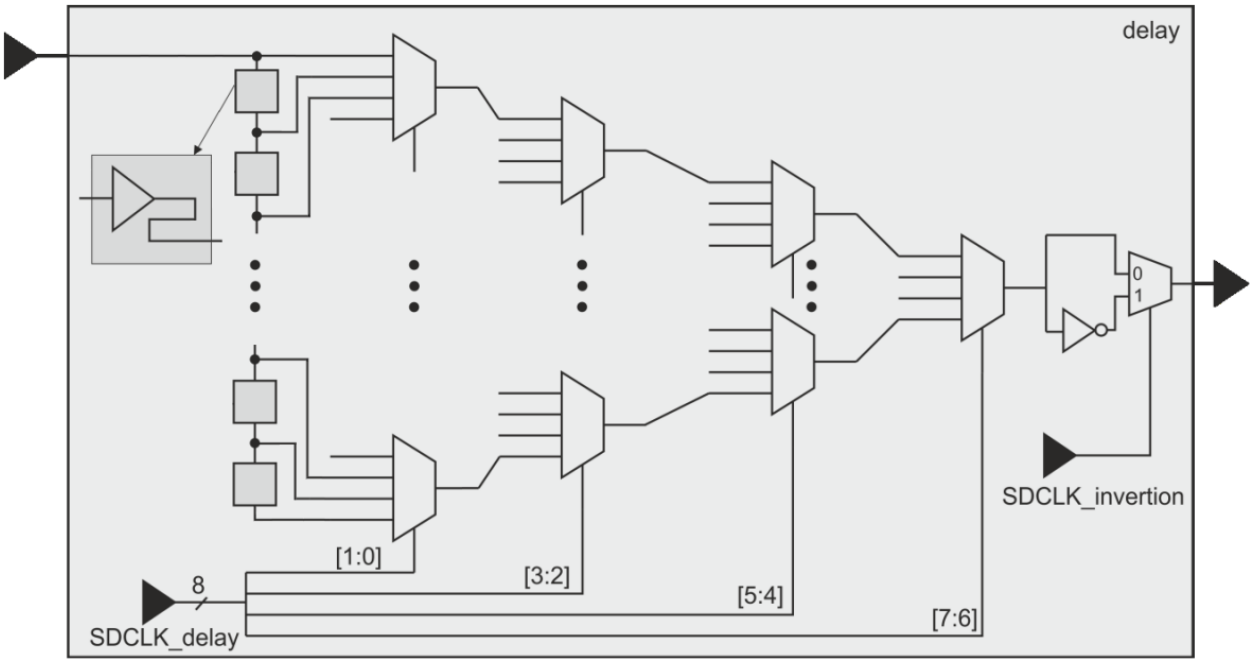


Рисунок 4.7.5 – Внутренняя структура тракта задержки сигнала SDCLK

Значение по сбросу всех полей регистра SDCLK_CTRL равно 0. Т.е. функционал управляемого тракта запрещен, тактирование на него не подается, что позволит избежать дополнительного потребления мощности, в случае использования SRAM в качестве ОЗУ.

На рисунке 4.7.6 приведена ориентировочная форма зависимости дополнительной задержки в диапазоне значений поля SDCD при установленном бите EN регистра SDCLK_CTRL, при штатном напряжении питания и температуре окружающей среды 20 °C.

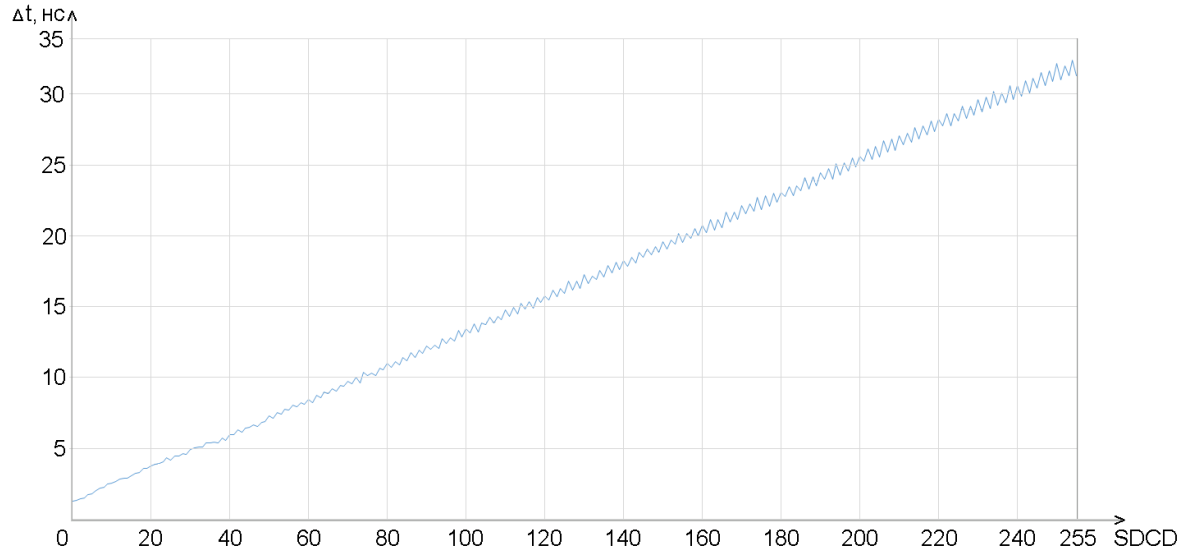


Рисунок 4.7.6 – График зависимости задержки сигнала SDCLK от значения поля SDCD

4.7.5 Инициализация микросхем динамической памяти

Старт процесса инициализации осуществляется по записи в регистр управления SDRAM_CFG_1 (если установлен бит RS регистра FT_CTRL). Микросхемы SDRAM требуют перед подачей инициализационной последовательности выдержать интервал времени, отведенный на стабилизацию напряжения питания и тактовой частоты. Первая запись в регистр SDRAM_CFG_1 организует упомянутый выше период ожидания, определяемый значением поля IP, в конце которого будет сгенерирована последовательность команд PRECHARGE (для всех банков данных) – 8 x AUTO REFRESH – LOAD MODE REGISTER. Если к модулю подключено несколько микросхем памяти (для организации 64-битной разрядности шины данных, либо, если используются выводы SDCSN[1], SDCKE[1] второго блока памяти SDRAM), то все они инициализируются параллельно.

Каждая последующая запись в регистр SDRAM_CFG_1 вызывает только последовательность команд Precharge (для всех банков данных) – LOAD MODE REGISTER. Подача команды LOAD MODE REGISTER при перезаписи данного регистра обязательна, поскольку при изменении CAS Latency (поле CL) требуется отправить данное значение и в микросхемы памяти.

Значение поля IP выбирается исходя из формулы:

$$N \times CF_{MHz} + 5 + tRP + 8 \times tRFC + tMRD,$$

где N – интервал инициализации динамической памяти в мкс (как правило, 100 мкс или 200 мкс),

CF_{MHz} – тактовая частота процессора в МГц,

tRP, tRFC, tMRD – временные характеристики динамической памяти.

На рисунках 4.7.7 и 4.7.8 приведены примеры временных диаграмм инициализационной последовательности, с учетом подбора оптимальных временных параметров контроллера для тактовых частот 100 МГц и 50 МГц соответственно (в качестве исходных данных используются временные параметры микросхем MT48LC16M16A2). Используемые обозначения: P – PRECHARGE, AR – AUTO REFRESH, LMR – LOAD MODE REGISTER, MR – значение, загружаемое в Mode Register микросхем памяти, A10 обозначает установленный бит SA[10] (PRECHARGE всех банков данных). Упомянутые стадии выполнения инициализации: 0 – завершающий такт

интервала ожидания, задаваемого значением поля IP регистра SDRAM_CFG_1; 1 – PRECHARGE всех банков данных; 2 – 8 последовательных AUTO REFRESH; 3 – LOAD MODE REGISTER.

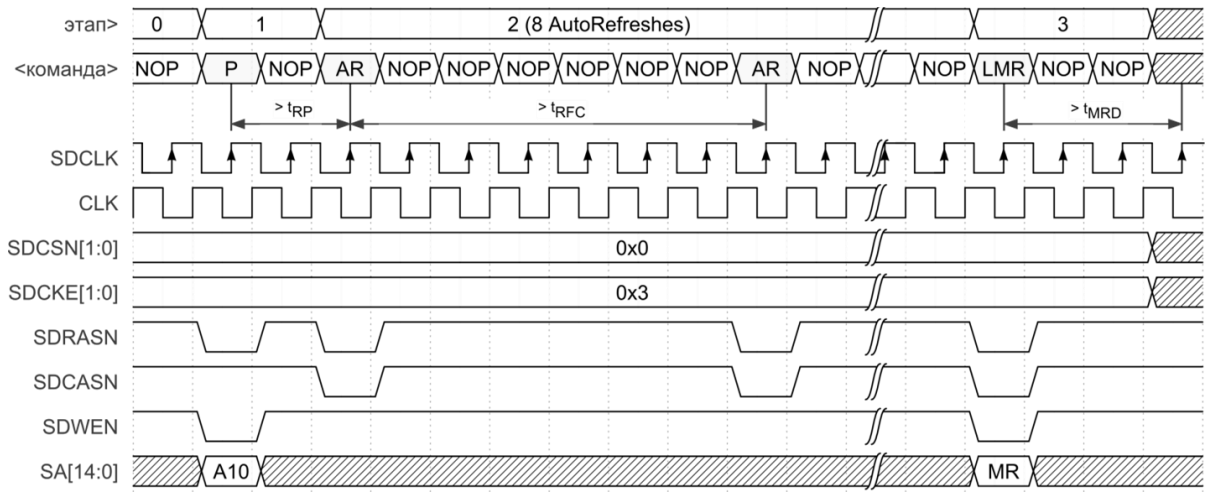


Рисунок 4.7.7 – Пример инициализационной последовательности (CF_{MHz} = 100 МГц)

При периоде тактового сигнала 10 нс для соблюдения временных ограничений микросхем (t_{RP} > 16 нс, t_{RFC} > 66 нс, t_{MRD} – 3 такта до следующей команды) достаточно tRP = 1, tRFC = 6, tMRD = 2.

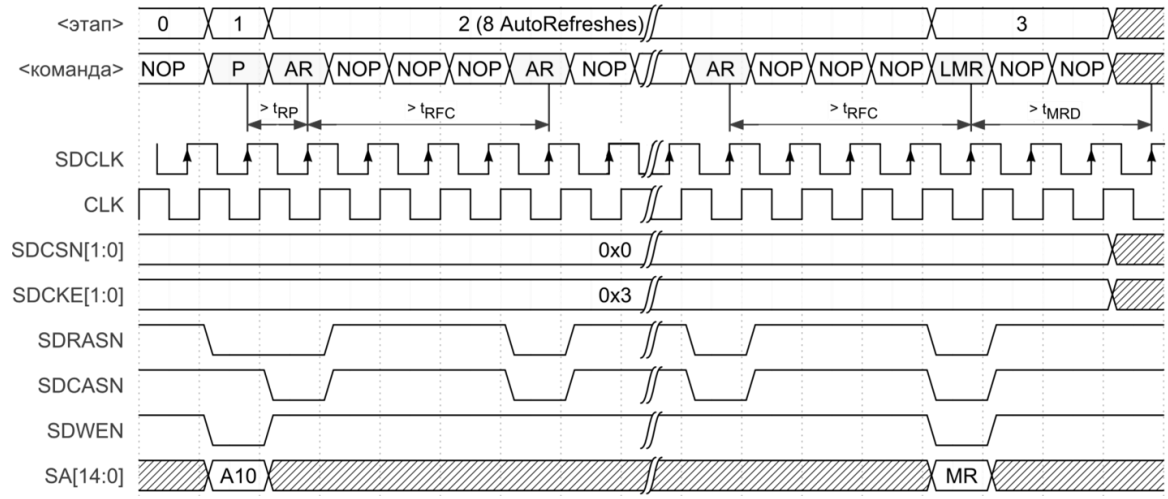


Рисунок 4.7.8 – Пример инициализационной последовательности (CF_{MHz} = 50 МГц)

При периоде тактового сигнала 20 нс для соблюдения временных ограничений микросхем (t_{RP} > 16 нс, t_{RFC} > 66 нс, t_{MRD} – 3 такта до следующей команды) достаточно tRP = 0, tRFC = 3, tMRD = 2.

Более подробная информация по выбору tRP, tRFC, tMRD находится в пункте «4.7.7 Управление параметрами конфигурации SDRAM».

4.7.6 Режимы выполнения обращений к памяти

Каждый банк данных микросхемы SDRAM является двумерным массивом. За выбор конкретной ячейки памяти отвечает адрес ряда и адрес столбца. Для осуществления доступа предварительно требуется активировать нужный ряд (выполнить команду ACTIVE). В команде WRITE / READ выставляется только адрес столбца в уже

активированном ряду. Прежде чем активировать другой ряд в пределах банка данных необходимо деактивировать уже открытый ряд (выполнить команду Precharge). В контроллере реализовано несколько режимов выполнения обращений в память SDRAM. Принципиально друг от друга они отличаются только механизмом активации и деактивации рядов. Режим подачи команд PRECHARGE задается значением поля PM регистра SDRAM_CFG_2, по умолчанию используется режим с одним активным рядом (00_2).

Режим с одним активным рядом

Обращение к памяти (чтение или запись) состоит из следующей последовательности:

• PRECHARGE (если активирован ряд, но обращение идет к другому ряду);
• ACTIVE (если требуемый ряд не активирован);
• WRITE / READ.

Режим подразумевает минимизацию временных накладных расходов на активацию (ACTIVE) и деактивацию рядов микросхем SDRAM (PRECHARGE) при обращениях в пределах одного ряда. Использование данного режима не отменяет подачу PRECHARGE на все блоки перед выполнением AUTO REFRESH, поскольку это является неотъемлемым условием выполнения самой команды.

Если память SDRAM используется только для хранения данных и к ней осуществляется только эпизодический доступ, то выбор подобного режима будет не оправдан с точки зрения энергопотребления (ряд будет оставаться активирован до наступления очередного AUTO REFRESH).

В технической документации на микросхемы SDRAM задается промежуток времени между подачей команд ACTIVE и PRECHARGE – t_{RAS} (для MT48LC16M16A2 42 нс < t_{RAS} < 120000 нс). В данном режиме его верхняя граница косвенным образом накладывает ограничение на период между выполнением автоматической регенерации рядов (задается значением поля RP регистра SDRAM_CFG_2), поскольку ряд будет гарантированно закрыт только перед выполнением AUTO_REFRESH.

Режим с PRECHARGE после чтения/записи

Обращение к памяти (чтение или запись) состоит из следующей последовательности:

• ACTIVE;
• WRITE / READ;
• PRECHARGE (для всех банков данных микросхемы).

После каждой операции выполняется деактивация ряда, что переводит микросхемы памяти в состояние IDLE. В зависимости от электрических параметров используемых микросхем может наблюдаться выигрыш по энергопотреблению относительно режима с одним активным рядом.

Режим чтения/записи с автоматическим PRECHARGE

Аналогично предыдущему режиму перед каждой операцией записи или считывания производится активация требуемого ряда, а после её завершения – деактивация.

Обращение к памяти (чтение или запись) состоит из следующей последовательности:

• ACTIVE;
• WRITE с автоматическим PRECHARGE / READ с автоматическим PRECHARGE.

При использовании данного режима следует помнить, что параметр t_{WR} динамической памяти, задающий количество тактов между командами WRITE и

PRECHARGE рассчитывается исходя из формулы вида (1 такт + несколько нс), а не опирается исключительно на значение интервала времени в нс.

4.7.7 Управление параметрами конфигурации SDRAM

Для корректной работы выбранных микросхем памяти SDRAM через регистр управления SDRAM_CFG_1 следует запрограммировать следующие параметры микросхем SDRAM: задержку (CAS Latency), размер столбцов и рядов, период инициализации. Информация о расчете периода инициализации приведена в пункте «4.7.5 Инициализация микросхем динамической памяти».

Поле CL (CAS Latency) определяет время задержки строб-импульса адреса столбца, соответствует двум младшим битам поля «CAS Latency» стандартного регистра управления динамической памятью (MODE REGISTER). Старший бит всегда равен нулю. От данного параметра зависит задержка между запросом на операцию чтения и появлением данных на шине SD. В ходе инициализации в MODE REGISTER будет установлено значение «3». Запись в регистр вызовет выполнение команды LOAD MODE REG с новым значением MODE REGISTER.

Ширина адреса столбцов (поле CAW) определяет количество столбцов в динамической памяти, вычисляемое по формуле $2^{(CAW + 8)}$. Допустимые значения лежат в диапазоне от 0 до 4, соответственно поддерживаются микросхемы с количеством столбцов от 256 до 4096.

Ширина адреса ряда (поле RAW) определяет количество рядов в динамической памяти, вычисляемое по формуле $2^{(RAW + 8)}$. Допустимые значения лежат в диапазоне от 0 до 5, соответственно поддерживаются микросхемы с количеством рядов от 256 до 8192.

Для обеспечения оптимальных циклов доступа для различных устройств SDRAM (различных тактовых частот) через регистр конфигурации памяти следует запрограммировать следующие параметры микросхем SDRAM: tMRD, tWR, tRCD, tRFC, tRP, период обновления. Значения по умолчанию данных полей см. в регистре управления SDRAM_CFG_2.

Поле tMRD задает количество тактов NOP между загрузкой MODE_REGISTER и последующими командами ACTIVE или REFRESH. По умолчанию установлено значение «3», большинству современных микросхем достаточно 2 тактов.

Поле tWR (WRITE recovery time) задает количество тактов NOP между командами WRITE и PRECHARGE.

Поле tRCD (ACTIVE-to-READ or WRITE delay) задает количество тактов NOP между командой ACTIVE и командами READ или WRITE.

Поле tRFC (AUTO REFRESH period) задает количество тактов NOP после команды AUTO REFRESH.

Поле tRP (PRECHARGE command period) задает количество тактов неактивности после команды PRECHARGE.

Значение полей tWR, tRCD, tRFC, tRP должно быть выбрано исходя из формулы:

$$\left[\frac{\langle t \text{ параметр в нс}\rangle \times \langle \text{тактовая частота в МГц}\rangle}{1000}\right],$$

где [ ] – операция взятия целой части от числа;

t – соответствующий временной параметр микросхемы (t_{WR}, t_{RCD}, t_{RFC}, t_{RP}).

Операция взятия целой части, реализующая округление вниз до ближайшего целого, используется, поскольку управляющие сигналы выполняемой команды удерживается на выводах процессора 1 такт системной частоты. Поэтому, если временной параметр микросхемы (t_{WR}, t_{RCD}, t_{RFC} или t_{RP}) укладывается в период системной тактовой

частоты, то нет никакой потребности в дополнительных тактах NOP между соответствующими командами SDRAM.

В качестве примера на рисунке 4.7.9 приведено выполнение операции записи, деактивации текущего ряда и активация нового ряда в режиме с Precharge после чтения/записи (или в режиме с одним активным рядом, если обращение идет в другой ряд или банк данных) для тактовой частоты 100 МГц. Для соблюдения временных ограничений микросхем (t_{WR} > 14 нс, t_{RP} > 15 нс) достаточно установить tWR = 1 и tRP = 1.

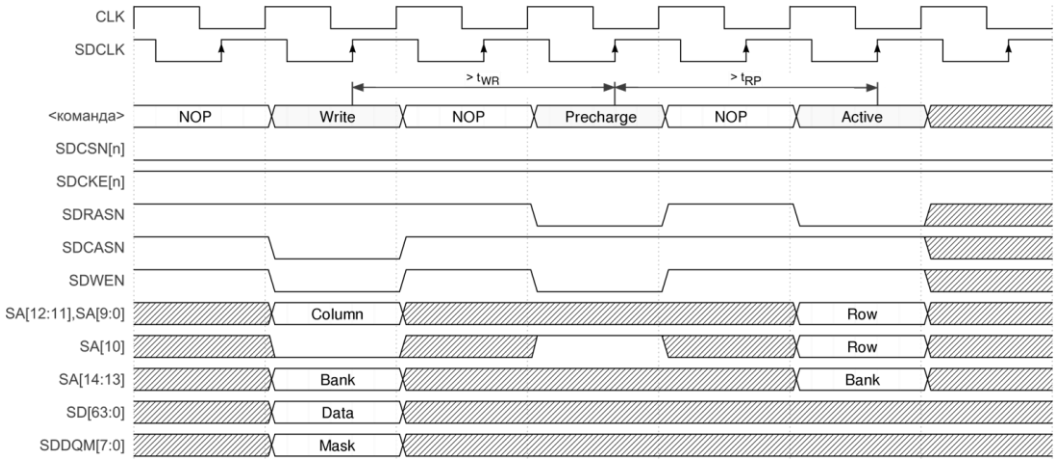


Рисунок 4.7.9 – Закрытие ряда после записи и открытие нового ряда (CF_{MHz} = 100 МГц)

Период обновления (RP) задает значение периода (в тактах системной шины) между последовательными командами AUTO REFRESH. Команда AUTO REFRESH инициирует регенерацию одного ряда в каждом из банков данных, номер обновляемого ряда задает внутренний счетчик микросхем памяти. Значение поля RP должно быть выбрано исходя из следующей формулы:

$$\left[\frac{\langle t_{REF} \text{ в мкс}\rangle \times \langle \text{тактовая частота в МГц}\rangle}{\langle \text{количество рядов в банке данных}\rangle}\right],$$

где [ ] – операция взятия целой части от числа;

t_{REF} в мкс – верхняя граница периода обновления всех рядов банка данных в микросекундах.

Т.е. если в спецификации на микросхемы памяти указано, что период обновления всех рядов банка данных (t_{REF}) не должен превышать64 мс, в банке данных 8192 рядов, а тактовая частота системы – 75 МГц, то значение периода обновления составляет $[64000 \times 75/8192] = [585{,}9375] = 585 = 0x249$. Если значение частоты системы изменяется, то требуется изменить RP так, чтобы обновления динамической памяти проходили не реже, чем этого требует спецификация микросхем памяти. Например, если тактовая системная частота – 50 МГц, то $RP = [64000 \times 50/8192] = 390 = 0x186$.

Команда AUTO REFRESH может быть выполнена, только если все банки данных деактивированы. Если есть не закрытые ряды, то контроллер SDRAM автоматически выполняет PRECHARGE для всех банков данных. В связи с этим не стоит задавать значение RP меньше чем (1 + tRP + 1 + tRFC).

Период обновления косвенно ограничивает минимальную допустимую частоту работы контроллера с микросхемами SDRAM. Для частоты порядка 20 МГц существенную часть времени контроллер будет занят выполнением автоматической регенерации рядов SDRAM. Использование механизма AUTO REFRESH не является обязательным для сохранения данных в микросхемах SDRAM. Если доступ к каждой ячейке памяти, хранящей используемые данные, осуществляется не реже указанного в

документации на микросхему промежутка времени (обычно 64 мс на обновление всех рядов банка данных), то можно отказаться от использования автоматической регенерации рядов (выставить максимальное доступное значение RP).

4.7.8 Коррекция ошибок

Активация режима коррекции ошибок для контроллера SDRAM происходит, только если на вход CFG_FT_ENA подается высокий логический уровень сигнала и установлен бит FT_CTRL.SDFT контроллера конфигураций. В противном случае контроль проверочных бит не осуществляется, при выполнении операций записи на шине FT_CHECKBITS[31:0] выставляется значение 0x00000000.

Код Хэмминга

Если режим коррекции ошибок разрешен для контроллера SDRAM, а бит FT_CTRL.SDC сброшен, то для обеспечения помехоустойчивости используется код Хэмминга(96,64). Он позволяет обнаружение до двух ошибок и исправление одной ошибки на кодовое слово.

В контроллере SDRAM кодовое слово состоит из исходных 64 бит данных и 8 проверочных бит. Каждый из них вычисляется, используя операции ИСКЛЮЧАЮЩЕГО ИЛИ над соответствующими битами данных. Ниже представлены формулы для вычисления проверочных бит:

```
CB0 = D0  ^ D1  ^ D2  ^ D3  ^ D4  ^ D5  ^ D6  ^ D7  ^ D8  ^ D9  ^ D10 ^
      D11 ^ D12 ^ D13 ^ D14 ^ D15 ^ D16 ^ D17 ^ D18 ^ D19 ^ D20 ^ D21 ^
      D22 ^ D23 ^ D24 ^ D25 ^ D26 ^ D27 ^ D28 ^ D29 ^ D30 ^ D31 ^ D32 ^
      D33 ^ D34 ^ D35 ^ D36 ^ D37 ^ D38 ^ D39 ^ D40 ^ D41 ^ D42 ^ D43 ^
      D44 ^ D45 ^ D46 ^ D47 ^ D48 ^ D49 ^ D50 ^ D51 ^ D52 ^ D53 ^ D54 ^
      D55 ^ D56 ^ D57 ^ D58 ^ D59 ^ D60 ^ D61 ^ D62 ^ D63 ^ CB1 ^
      CB2 ^ CB3 ^ CB4 ^ CB5 ^ CB6 ^ CB7;
CB1 = D0  ^ D1  ^ D3  ^ D4  ^ D6  ^ D8  ^ D10 ^ D11 ^ D13 ^ D15 ^ D17 ^
      D19 ^ D21 ^ D23 ^ D25 ^ D26 ^ D28 ^ D30 ^ D32 ^ D34 ^ D36 ^ D38 ^
      D40 ^ D42 ^ D44 ^ D46 ^ D48 ^ D50 ^ D52 ^ D54 ^ D56 ^ D57 ^ D59 ^
      D61 ^ D63;
CB2 = D0  ^ D2  ^ D3  ^ D5  ^ D6  ^ D9  ^ D10 ^ D12 ^ D13 ^ D16 ^ D17 ^
      D20 ^ D21 ^ D24 ^ D25 ^ D27 ^ D28 ^ D31 ^ D32 ^ D35 ^ D36 ^ D39 ^
      D40 ^ D43 ^ D44 ^ D47 ^ D48 ^ D51 ^ D52 ^ D55 ^ D56 ^ D58 ^ D59 ^
      D62 ^ D63;
CB3 = D1  ^ D2  ^ D3  ^ D7  ^ D8  ^ D9  ^ D10 ^ D14 ^ D15 ^ D16 ^ D17 ^
      D22 ^ D23 ^ D24 ^ D25 ^ D29 ^ D30 ^ D31 ^ D32 ^ D37 ^ D38 ^ D39 ^
      D40 ^ D45 ^ D46 ^ D47 ^ D48 ^ D53 ^ D54 ^ D55 ^ D56 ^ D60 ^ D61 ^
      D62 ^ D63;
CB4 = D4  ^ D5  ^ D6  ^ D7  ^ D8  ^ D9  ^ D10 ^ D18 ^ D19 ^ D20 ^ D21 ^
      D22 ^ D23 ^ D24 ^ D25 ^ D33 ^ D34 ^ D35 ^ D36 ^ D37 ^ D38 ^ D39 ^
      D40 ^ D49 ^ D50 ^ D51 ^ D52 ^ D53 ^ D54 ^ D55 ^ D56;
CB5 = D11 ^ D12 ^ D13 ^ D14 ^ D15 ^ D16 ^ D17 ^ D18 ^ D19 ^ D20 ^ D21 ^
      D22 ^ D23 ^ D24 ^ D25 ^ D41 ^ D42 ^ D43 ^ D44 ^ D45 ^ D46 ^ D47 ^
      D48 ^ D49 ^ D50 ^ D51 ^ D52 ^ D53 ^ D54 ^ D55 ^ D56;
CB6 = D26 ^ D27 ^ D28 ^ D29 ^ D30 ^ D31 ^ D32 ^ D33 ^ D34 ^ D35 ^ D36 ^
      D37 ^ D38 ^ D39 ^ D40 ^ D41 ^ D42 ^ D43 ^ D44 ^ D45 ^ D46 ^ D47 ^
      D48 ^ D49 ^ D50 ^ D51 ^ D52 ^ D53 ^ D54 ^ D55 ^ D56;
CB7 = D57 ^ D58 ^ D59 ^ D60 ^ D61 ^ D62 ^ D63;
```

Контроллер SDRAM при обнаружении исправимой ошибки восстановит корректные данные и сигнализирует о произошедшем событии установкой пары битов CE и NE в регистре статуса AHB. Если выявлено больше ошибок, чем модуль в состоянии исправить, то будет установлен только бит NE. Помимо отображения факта неисправимой ошибки в регистрах статуса AHB, в процессоре генерируется исключение instruction_access_error (для операций выборки инструкций), data_access_error (для запросов на считывание данных) или data_store_error (при обращениях вида чтение-модификация-запись). Появление исключения приведет к переходу в таблицу прерываний (помимо выделения нового регистрового окна и сохранения значений PC и nPC в его регистры %l1 и %l2) или к переводу процессора в состояние ошибки, в зависимости от значения бита ET регистра %psr.

Если установлен бит FT_CTRL.WBSD (по сбросу данный режим активен для области SDRAM) при обнаружении исправимой ошибки контроллер автоматически перезапишет исправленное кодовое слово во внешнюю память.

Код Рида-Соломона

Код Рида-Соломона является недвоичным кодом, т.е. он работает с символами, а не с битами. В процессорах 1906BM016, 1906BM01A6 выбран Код Рида-Соломона (96,64) с размером символа в 8 бит, который в состоянии исправить до двух некорректных символов (от 2 до 16 бит на 64 бит данных и 32 проверочных бита, если они локализованы в двух символах). Для кодирования и декодирования данных используется примитивный полином $x^8 + x^4 + x^3 + x^2 + 1$.

Модуль кодирования является полностью комбинаторной схемой, выполняющей последовательные сдвиг и суммирование по модулю 2, на выходе которого помимо исходных 64 бит данных генерируются и 32 проверочных бита.

Модуль декодирования является конвейеризованным. В случае обнаружения исправимой ошибки, спустя 5 тактов на выходе декодера появятся скорректированные данные, и будет установлена пара битов CE и NE в регистре статуса AHB. Если в одном из символов данных содержится больше ошибок, чем модуль в состоянии исправить, то будет установлен только бит NE. Помимо отображения факта неисправимой ошибки в регистрах статуса AHB, в процессоре генерируется исключение instruction_access_error (для операций выборки инструкций), data_access_error (для запросов на считывание данных) или data_store_error (при обращениях вида чтение-модификация-запись). Появление исключения приведет к переходу в таблицу прерываний (помимо выделения нового регистрового окна и сохранения значений PC и nPC в его регистры %l1 и %l2) или к переводу процессора в состояние ошибки, в зависимости от значения бита ET регистра %psr. Если ошибок детектировано не было, то данные с входа модуля могут быть использованы в схеме без необходимости декодирования.

Если установлен бит WBSD регистра FT_CTRL (по сбросу данный режим активен для области SDRAM) при обнаружении исправимой ошибки контроллер автоматически перезапишет исправленное кодовое слово во внешнюю память.

Процесс декодирования осуществляется по следующему алгоритму:

1) Поиск синдрома ошибки. Если ошибок не обнаружено, то данные, минуя остальные этапы, передаются по шине AHB.
2) Вычисление полинома ошибок.
3) Вычисление вектора локализации ошибки.
4) Коррекция ошибок.
5) Дополнительная стадия, используемая для ретайминга.

4.7.9 Регистры

Регистры управления контроллерами памяти PROM / IO, SRAM, SDRAM вынесены в модуль контроллера конфигураций (их непосредственное описание приведено в пункте «4.3.4 Регистры»).

4.8 Контроллер прерываний

4.8.1 Общие сведения

Система AMBA в GRLIB обеспечивает схему прерывания, в которой линии прерывания шин AHB/APB обрабатываются совместно, формируя шину прерывания. Прерывания AHB и APB проходят через шину вместе и передаются обратно ко всем модулям. Контроллер прерываний (IRQMP) подсоединен к шине AMBA, как ведомое устройство APB, и управляется общими сигналами прерывания.

Прерывания, генерируемые на шине прерывания, передаются контроллеру прерываний. Контроллер прерываний определяет приоритет, маскирует и передает прерывание с самым высоким приоритетом в процессор, что проиллюстрировано на рисунке 4.8.1.

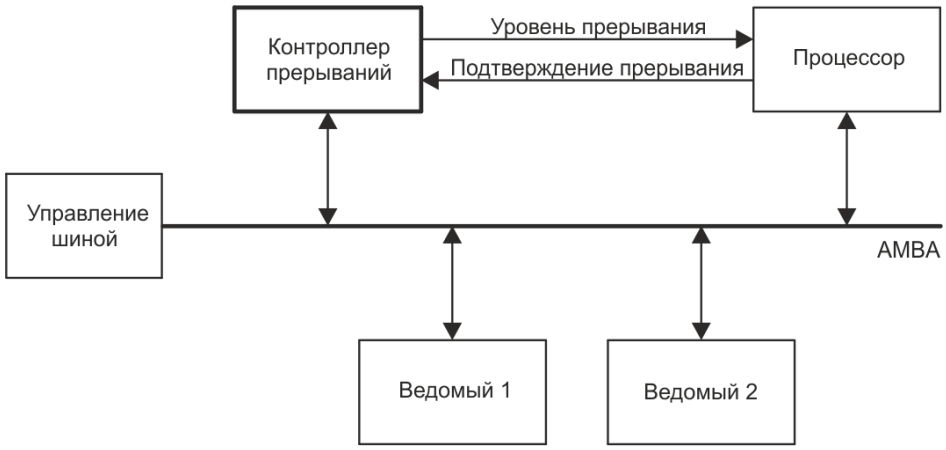


Рисунок 4.8.1 – Процессор LEON с контроллером прерываний

4.8.2 Принцип работы

Приоритет прерываний

Контроллер прерываний контролирует прерывания (1 – 15) шины прерываний (APBI.PIRQ(15:1)). Если линии находятся в высоком логическом состоянии, устанавливается соответствующий бит в регистр ожидающих обработки прерываний. Даже если линия PIRQ находится в сброшенном состоянии, ждущие обработки биты остаются в установленном состоянии до программного сброса или до получения подтверждения прерывания от процессора.

Каждое прерывание закреплено за одним из двух уровней (0 или 1), как запрограммировано в регистре уровней прерываний. Уровень 1 имеет более высокий приоритет, чем уровень 0. Приоритет прерываний распределен на каждом уровне, причем прерывание 15 имеет самый высокий приоритет, а прерывание 1 – самый низкий приоритет. Прерывание уровня 1 с самым высоким приоритетом передается процессору. При отсутствии ожидающих обработки немаскированных прерываний уровня 1

процессору передается немаскированное прерывание уровня 0 с самым высоким приоритетом. PIRQ(31:16) не используются системой IRQMP.

Приоритет прерываний распределен на системном уровне (см. таблицу 4.8.1). В процессоре имеются отдельные регистры маски прерываний и вынужденных прерываний. При получении сигнала о прерывании на шине контроллер прерываний выставляет приоритет прерывания, выполняет маскирование прерываний и передает прерывание в процессор.

Блок-схема контроллера прерываний представлена на рисунке 4.8.2.

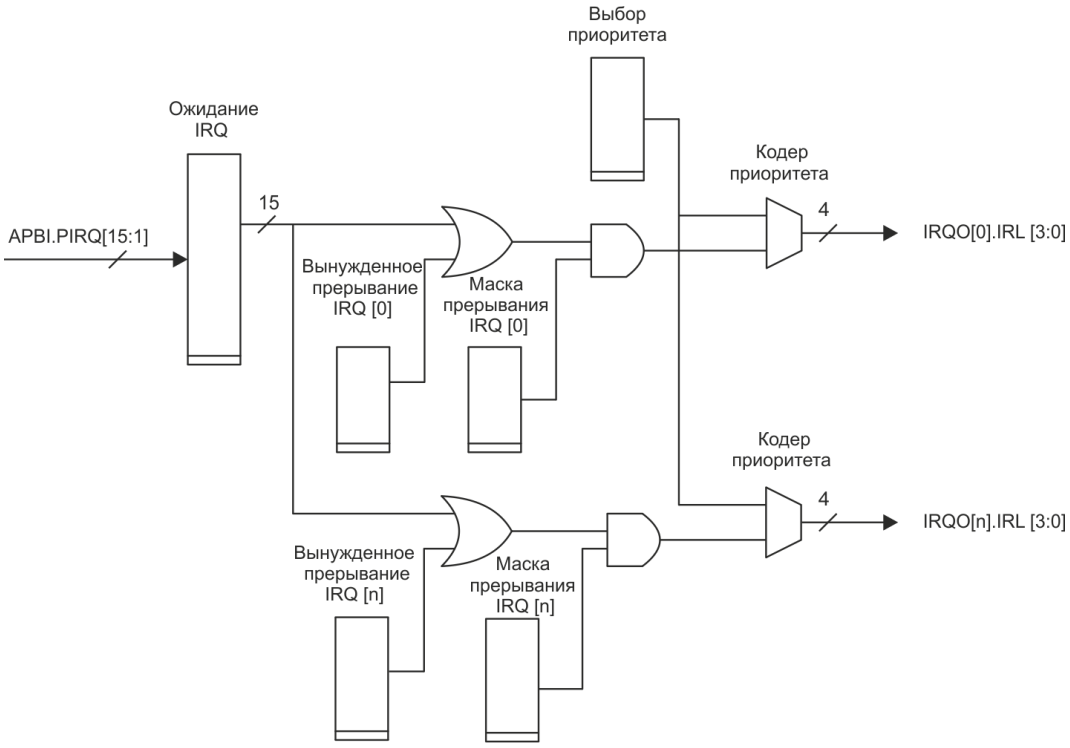


Рисунок 4.8.2 – Блок-схема контроллера прерываний

Когда процессор подтверждает прерывание, соответствующий ожидающий обработки бит автоматически сбрасывается. Прерывание можно вызвать принудительно через установку бита регистра принудительного прерывания. В этом случае после подтверждения процессором сбрасывается бит принуждения, но не бит ожидания обработки. После сброса регистр маски прерываний сбрасывается, в то время как значения остальных регистров управления не определены. Следует обратить внимание на то, что прерывание 15 не маскируется процессором LEON и должно использоваться с осторожностью – большинство операционных систем не обрабатывают это прерывание в безопасном режиме.

Расширенные прерывания

Поскольку процессор имеет только 15 уровней прерываний (interrupt_level_1 – interrupt_level_15, см. подпункт «Исключения» пункта «4.4.2»), расширенные прерывания генерируются одним из регулярных прерываний. Для ИС 1906ВМ016, 1906ВМ01А6 это прерывание – номер 9**,** в соответствии с содержимым таблицы 4.8.1.

Если прерывание принимается и подтверждается процессором, биты регулярного (9-го) и расширенного прерывания в регистре ожидающих обработки прерываний автоматически сбрасываются.

Регистр подтверждения приема расширенного прерывания позволяет определить номер подтвержденного расширенного прерывания. Этот регистр может использоваться

программным обеспечением для того, чтобы вызвать соответствующий обработчик для расширенных прерываний.

Контроль состояния процессора

Состояние процессора можно контролировать через регистр статуса процессорной системы. Поле STATUS в этом регистре указывает остановку процессора «1» или его работу «0». Остановленный процессор можно сбросить и перезапустить при записи «1» в поле статуса.

4.8.3 Таблица прерываний

В микропроцессорах 1906ВМ016, 1906ВМ01А6 используются расширенные прерывания, что приведено в таблице 4.8.1.

Таблица 4.8.1 – Таблица прерываний

Модули	Номер прерывания	Функция
DSU	0	Прерывание модуля отладки
AHBSTAT	1	Ошибка шины AHB
APBUART_1	2	Прерывания UART1 RX/TX
APBUART_2	3	Прерывания UART2 RX/TX
CAN_MC_1	4	Прерывания интерфейса CAN_1
CAN_MC_2	5	Прерывания интерфейса CAN_2
GRPCI2	6	Прерывания PCI Error и DMA
GPTIMER1_1	7	Прерывание по переполнению Таймера 1_1
GPTIMER1_2 (WATCHDOG)	8	Прерывание по переполнению Таймера 1_2
IRQMP	9	Расширенные прерывания
GRSPW2_1	10	Прерывания SpaceWire1
GRSPW2_2	11	Прерывания SpaceWire2
GRSPW2_3	12	Прерывания SpaceWire3
GRSPW2_4	13	Прерывания SpaceWire4
GRETH	14	Прерывания Ethernet
–	15*	Не зарезервировано ни за одним интерфейсом
GRUSBHC	16**	Прерывания расширенного контроллера USB
GRUSBHC	17**	Прерывания универсального контроллера USB
GR1553B_1	18**	Прерывание MIL-STD-1553 1
GR1553B_2	19**	Прерывание MIL-STD-1553 2
GPTIMER2_1	20**	Прерывание по переполнению Таймера 2_1
GPTIMER2_2	21**	Прерывание по переполнению Таймера 2_2
GPTIMER2_3	22**	Прерывание по переполнению Таймера 2_3
GPIO	1–15	Внешние прерывания

* Линия 15 является немаскируемой, следует использовать с осторожностью.
** Относятся к расширенной области прерываний.

4.8.4 Регистры

Система управляется через регистры, отображенные в адресуемом пространстве APB. Количество реализованных регистров зависит от количества процессоров в многопроцессорной системе, см. таблицы 4.8.2 – 4.8.10.

Таблица 4.8.2 – Регистры контроллера прерываний (базовый адрес 0x80000200)

Адрес смещения APB	Регистр
0x00	Регистр уровней прерываний
0x04	Регистр ожидающих обработки прерываний
0x08	Регистр принудительной установки прерываний
0x0C	Регистр сброса прерываний
0x10	Регистр статуса процессорной системы
0x14 – 0x3C	Зарезервировано
0x40	Регистр маски прерывания процессора
0x80	Регистр принудительной установки и сброса прерываний
0xC0	Регистр подтверждения приема расширенного прерывания

Регистр уровней прерываний

Таблица 4.8.3 – Регистр уровней прерываний

31 16	15 1	0
Зарезервировано	IL(15:1)	–
0	n/r	0
r	rw	r

31–16 Зарезервировано
15–1 Уровень прерывания n (IL(n)) – Уровень прерывания для прерывания n
0 Зарезервировано

Регистр ожидающих обработки прерываний

Таблица 4.8.4 – Регистр ожидающих обработки прерываний

31 16	15 1	0
EIP(31:16)	IP(15:1)	–
0	0	0
rw	rw	r

31–16 Расширенное прерывание n ожидает рассмотрения (EIP(n))
15–1 Прерывание с порядковым номером n ожидает рассмотрения (IP(n))
0 Зарезервировано

Регистр принудительной установки прерываний

Таблица 4.8.5 – Регистр принудительной установки прерываний

31 16	15 1	0
Зарезервировано	IF(15:1)	–
0	0	0
r	rw	r

31–16 Зарезервировано
15–1 Принудительно установить прерывание n (IF(n))

0 Зарезервировано

Регистр сброса прерываний

Таблица 4.8.6 – Регистр сброса прерываний

31 16	15 1	0
EIC(31:16)	IC(15:1)	–
0	0	0
wc	wc	r

31–16 Сброс расширенного прерывания n (EIC(n))
15–1 Сброс прерывания n (IC(n)) – Запись «1» в IC(n) сбрасывает прерывание n
0 Зарезервировано

Регистр статуса процессорной системы

Таблица 4.8.7 – Регистр статуса процессорной системы

31 28	27	26 20	19 16	15 1	0
NCPU	BA	–	EIRQ	STATUS(15:1)	STATUS(0)
0	0	0	0x9	0	0
r	r	r	r	r	r/wc

31–28 Количество CPU (NCPU) – Количество CPU в системе «минус» 1
27 Доступно транслирование (BA) – Сброшено, поскольку NCPU = 0
26–20 Зарезервировано
19–16 Расширенный запрос на прерывание (EIRQ) – Номер прерывания (1–15), используемый для расширенных прерываний (в соответствии с данными таблицы 4.8.1).
15–0 Статус пониженного потребления энергии CPU(n) (STATUS(n)) – «1» = пониженное потребление энергии, «0» = рабочий режим. Запись «1» в STATUS(n) запускает процессор n

Регистр маски прерывания процессора

Таблица 4.8.8 – Регистр маски прерывания процессора

31 16	15 1	0
EIM(31:16)	IM(15:1)	–
0	0	0
rw	rw	r

31–16 Маска расширенных прерываний (EIM(n)) – Если EIM(n) = «0», прерывание n из расширенной области прерываний маскируется, в противном случае оно разрешено
15–1 Маска прерывания n (IM(n)) – Если IM(n) = «0», прерывание n маскируется, в противном случае оно разрешено
0 Зарезервировано

Регистр принудительной установки и сброса прерываний

Таблица 4.8.9 – Регистр принудительной установки и сброса прерываний

31 17	16	15 1	0
IFC(15:1)	–	IF(15:1)	–
0	0	0	0
wc	r	rw	r

31–17 Принудительно сбросить прерывание n (IFC(n))
16 Зарезервировано
15–1 Принудительно установить прерывание n (IF(n))
0 Зарезервировано

Регистр подтверждения приема расширенного прерывания

Таблица 4.8.10 – Регистр подтверждения приема расширенного прерывания

31 5	4 0
Зарезервировано	EID[4:0]
0	0
r	r

31–5 Зарезервировано
4–0 Номер последнего принятого расширенного прерывания (16–31). Если данное поле имеет нулевое значение, то прерывание 9 (IRQMP) возникло не в результате срабатывания расширенного прерывания. Если прерывание 9 установлено принудительно, то данный регистр будет оставаться сброшенным до тех пор, пока одно или более из расширенных прерываний не установится в регистре ожидающих рассмотрения прерываний

4.9 Отладочный модуль LEON4

4.9.1 Общие сведения

Для упрощения отладки целевого аппаратного оборудования процессор LEON4 реализует режим отладки, во время которого конвейер неактивен и управление процессором осуществляется через специальный интерфейс отладки. Отладочное устройство (блок DSU4) LEON4 используется для управления процессором в режиме отладки. DSU работает как подчиненное устройство AHB, причем доступ к нему можно получить через любое ведущее устройство AHB. Поэтому, через внешний узел отладки можно получить доступ к DSU посредством нескольких различных интерфейсов, что видно из рисунка 4.9.1. Таким интерфейсом может быть UART (RS232), JTAG или Ethernet.

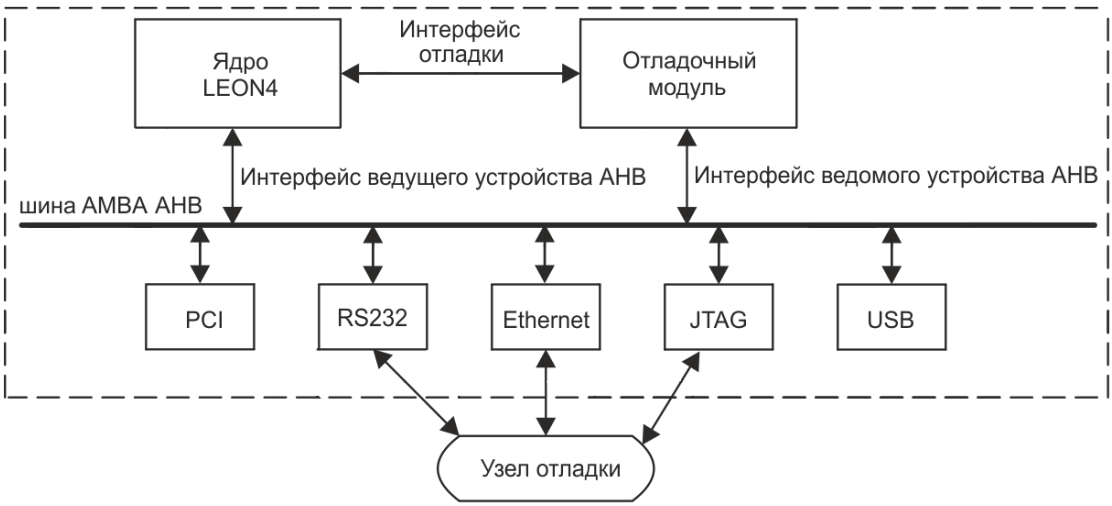


Рисунок 4.9.1 – Соединение LEON4/DSU

4.9.2 Принцип работы

Через интерфейс ведомого устройства AHB DSU любое ведущее устройство AHB может получить доступ к регистрам процессора и содержанию буфера трассировки команды. Доступ к регистрам команд DSU можно получить в любое время, в то время как доступ к регистрам процессора, кэшу и буферу трассировки можно получить, только если процессор находится в режиме отладки. В режиме отладки работа конвейера процессора приостанавливается, и доступ к статусу процессора можно получить через DSU. Вход в режим отладки может осуществляться при возникновении следующих событий:

- выполнение инструкции программного прерывания (ta 1);
- срабатывание аппаратной точки останова/контрольной точки данных (системное прерывание 0x0B);
- прохождение переднего фронта внешнего сигнала прерывания (DSUBRE);
- установка бита немедленного прерывания (BN) в регистре прерывания и пошагового режима DSU;
- системное прерывание, которое вводит процессор в режим ошибки;
- возникновение любого или одного из заданных в регистре управления DSU прерываний;
- выполнение инструкции в пошаговом режиме;
- срабатывание точки останова или контрольной точки данных AHB DSU.

Вход в режим отладки возможен, только если установлен внешний сигнал разрешения отладочного режима (DSUEN).

Чтобы прерывание DSU и бит немедленного прерывания (BN) возымели эффект, в регистре управления DSU должен быть установлен бит остановки в контрольной точке блока целочисленной арифметики (BW). Этот бит устанавливается, когда сигнал DSUBRE активен после сброса, а также должен быть установлен отладочным программным обеспечением во время инициализации DSU. При входе в режим отладки выполняются следующие действия:

- сохраняются во временных регистрах PC и nPC (доступны через отладочное устройство);
- устанавливается выходной сигнал (DSUACT) для индикации статуса отладки;
- останавливаются модули таймеров (если бит DF регистра конфигураций соответствующего блока таймеров сброшен), фиксируя значения таймеров LEON и сторожевого таймера.

Инструкция, которая привела к вводу процессора в режим отладки, не выполняется и состояние процессора остается неизменным. Выполнение возобновляется при сбросе

бита BN в регистре прерывания и пошагового режима DSU или при сбросе внешнего сигнала DSUEN. Блок таймера повторно разрешен, при этом продолжается выполнение операции с сохранённых значений PC и nPC. В режим отладки можно также войти после входа процессора в режим ошибок, например, если приложение завершило свою работу и остановило работу процессора. Режим ошибки можно сбросить, а процессор перезапустить с любого адреса.

Когда процессор находится в режиме отладки, доступ к области диагностики ASI передается в IU, предоставляется доступ к пространству с идентификатором ASI, определенным в регистре ASI DSU (см. подпункт «4.9.5»), в качестве адреса используется 20 младших битов исходного адреса.

4.9.3 Буфер трассировки AHB

Буфер трассировки AHB состоит из кольцевого буфера, в котором хранится информация о передаче данных AHB. Адрес, данные и разные управляющие сигналы шины AHB сохраняются, и могут быть считаны при последующем анализе. Объем буфера трассировки AHB – 256 записей. Одна запись буфера трассировки инструкций занимает 160 бит, но адрес начала каждой записи имеет выравнивание по границе в 8 слов (1 слово – 32 бита). Сохраняемая информация приводится в таблице 4.9.1.

Таблица 4.9.1 – Размещение данных в буфере трассировки AHB

Биты	Имя	Определение
159 – 128	Загруженные/сохраненные данные	HRDATA/HWDATA(63:32) AHB
127	Срабатывание точки останова AHB	Бит установлен, если сработала точка останова AHB DSU
126	–	Не используется
125 – 96	Метка времени	Значение счетчика меток времени DSU
95	–	Не используется
94 – 80	Hirq	HIRQ (15 – 1) шины AHB
79	Hwrite	HWRITE шины AHB
78, 77	Htrans	HTRANS шины AHB
76 – 74	Hsize	HSIZE шины AHB
73 – 71	Hburst	HBURST шины AHB
70 – 67	Hmaster	HMASTER шины AHB
66	Hmastlock	HMASTLOCK шины AHB
65 – 64	Hresp	HRESP шины AHB
63 – 32	Загруженные/сохраненные данные	HRDATA/HWDATA (31 – 0) шины AHB
31 – 0	Адрес загрузки/сохранения	HADDR шины AHB

Размещение записей буфера трассировки AHB приведено в таблице 4.9.6.

В дополнение к сигналам AHB в буфере трассировки сохраняются показания счетчика меток времени DSU.

Для разрешения буфера трассировки необходимо установить бит разрешения (EN) в регистре управления трассировкой. Каждая передача AHB сохраняется в буфере кольцевым способом. Адрес, за которым закрепляется следующая передача, сохраняется в регистре индексов буфера трассировки и автоматически инкрементируется после каждой передачи. Трассировка останавливается при сбросе бита EN или при срабатывании точки останова AHB. Если процессор входит в режим отладки, трассировка временно приостанавливается при условии, что в регистре управления буфером трассировки AHB не установлен бит принудительной трассировки (TF). Если бит принудительной трассировки установлен, буфер трассировки разрешен при условии, что установлен бит разрешения. Бит TF сбрасывается при срабатывании точки останова AHB, а также может

быть сброшен программным способом. Следует обратить внимание на то, что ни буферную память, ни регистр нельзя прочитать (записать) программным способом, когда буфер трассировки разрешен (см. ниже).

В DSU имеется внутренний счетчик меток времени, который фиксирует своё значение при входе процессора в режим отладки. Если трассировка AHB выполняется в режиме отладки (с использованием бита принудительной трассировки), может оказаться желательным также включить счетчик меток времени. Это можно сделать с помощью бита разрешения таймера (TE). Следует обратить внимание на то, что метка времени также используется для буфера трассировки инструкций. Бит разрешения таймера должен быть установлен только при использовании DSU как буфера трассировки AHB, но не при обработке или отладке программного обеспечения. Бит разрешения таймера сбрасывается в тех же случаях, что и бит принудительной трассировки.

4.9.4 Буфер трассировки инструкций

Буфер трассировки инструкций состоит из кольцевого буфера, в котором сохраняются выполняемые инструкции. Буфер трассировки инструкций находится в процессоре и считывается через DSU. Объем буфера трассировки инструкций – 1 Кбайт (64 записи), по 128 бит (16 байт) на каждую запись. Сохраняемая информация приводится в таблице 4.9.2.

Таблица 4.9.2 – Размещение данных в буфере трассировки инструкций

Биты	Имя	Определение
126	Многотактная инструкция	Устанавливается на втором этапе многотактной инструкции
125 – 96	Метка времени	Значение счетчика метки времени DSU
95 – 64	Полученный или сохраненный адрес/данные	Результат инструкции, сохраненный адрес или сохраненные данные
63 – 34	Счетчик команд	Программный счетчик (2 самых младших бита удаляются, поскольку они всегда сброшены)
33	Системное прерывание инструкции	Установлено при системном прерывании трассируемой инструкции
32	Режим ошибки процессора	Установлено, если трассируемая инструкция привела к входу в режим ошибки процессора
31 – 0	Код операции (Opcode)	Код операции инструкции

Размещение записей буфера трассировки инструкций приведено в таблице 4.9.5.

Во время трассировки в буфере трассировки сохраняется одна инструкция на строку за исключением атомарных инструкций загрузки/сохранения, которые занимают две записи (одна для операции загрузки и одна для операции сохранения). Биты (95 – 64) в буфере соответствуют сохраненному адресу и загруженным данным для инструкций загрузки. Бит 126 устанавливается при второй записи.

Когда процессор входит в режим отладки, трассировка приостанавливается. Буфер трассировки и регистр управления буфером трассировки AHB можно прочесть и в него можно записать, пока процессор находится в режиме отладки. Во время трассировки инструкций (процессор в нормальном режиме) доступ к буферу трассировки и регистру управления буфером трассировки отсутствует. Инструкции трассировки можно дополнительно отфильтровать по типам инструкций. Инструкции, подлежащие трассировке, определены в регистре управления трассировкой инструкций (31 – 28) и представлены в таблице 4.9.3.

Таблица 4.9.3 – Работа фильтра трассировки

Фильтр трассировки	Трассируемые инструкции
0x0	Все инструкции
0x1	Инструкции SPARC формата 2
0x2	Изменения потока управления. Все инструкции вызова процедуры, передачи и системные прерывания, включая цели передачи
0x4	Инструкции SPARC формата 1 (CALL)
0x8	Инструкции SPARC формата 3 кроме LOAD или STORE
0xC	Инструкции SPARC формата 3 LOAD или STORE
0xD	Инструкции SPARC формата 3 LOAD или STORE в альтернативное адресное пространство
0xE	Инструкции SPARC формата 3 LOAD или STORE в альтернативное адресное пространство 0x80 – 0xFF

4.9.5 Карта памяти DSU

Карта памяти DSU представлена в таблице 4.9.4. Некоторые области приведены в более детальном виде в таблицах 4.9.5 – 4.9.7.

Таблица 4.9.4 – Карта памяти DSU (базовый адрес 0x90000000)

Смещение адреса	Регистр
1	2
0x000000	Регистр управления DSU
0x000008	Счетчик метки времени буферов трассировки
0x000020	Регистр прерывания и пошагового режима DSU
0x000024	Регистр маски режима отладки (функционал актуален только для мультипроцессорных систем)
0x000040	Регистр управления буфером трассировки AHB
0x000044	Регистр индекса буфера трассировки AHB
0x000048	Регистр управления фильтром буфера трассировки AHB (функционал не поддерживается)
0x00004C	Регистр маски фильтра буфера трассировки AHB (функционал не поддерживается)
0x000050	Регистр адреса точки останова AHB 1
0x000054	Регистр маски точки останова AHB 1
0x000058	Регистр адреса точки останова AHB 2
0x00005C	Регистр маски точки останова AHB 2
0x000070	Регистр счетчика инструкций
0x000080	Регистр управления контрольными точками AHB
0x000090 – 0x00009C	Регистры данных контрольной точки 1 AHB
0x0000A0 – 0x0000AC	Регистры масок контрольной точки 1 AHB
0x0000B0 – 0x0000BC	Регистры данных контрольной точки 2 AHB
0x0000C0 – 0x0000CC	Регистры масок контрольной точки 2 AHB
0x100000 – 0x10FFFC	Буфер трассировки инструкций (…0000_2: Биты трассировки 127 – 96, …0100_2: Биты трассировки 95 – 64, …1000_2: Биты трассировки 63 – 32, …1100_2: Биты трассировки 31 – 0)
0x110000	Регистр управления трассировкой инструкций 0
0x110004	Регистр управления трассировкой инструкций 1

Окончание таблицы 4.9.4

1	2
0x200000 – 0x20FFFC	Буфер трассировки AHB (…00000_2: Биты трассировки 127 – 96, …00100_2: Биты трассировки 95 – 64, …01000_2: Биты трассировки 63 – 32, …01100_2: Биты трассировки 31 – 0, …10000_2: Биты трассировки 159 – 128)
0x300000 – 0x3007FC	Регистровый файл блока целочисленной арифметики. NWINDOWS = 8 %o*n*: 0x300000 + (((%psr.CWP × 64) + 0x20 + *n* × 4) mod 0x200) %l*n*: 0x300000 + (((%psr.CWP × 64) + 0x40 + *n* × 4) mod 0x200) %i*n*: 0x300000 + (((%psr.CWP × 64) + 0x60 + *n* × 4) mod 0x200) %g*n*: 0x300000 + 0x200 + *n* × 4
0x301000 – 0x30107C	Регистровый файл FPU %f*n*: 0x301000 + *n* × 4
0x400000	Регистр Y
0x400004	Регистр PSR
0x400008	Регистр WIM
0x40000C	Регистр TBR (поле tt доступно для записи)
0x400010	Регистр PC
0x400014	Регистр nPC
0x400018	Регистр FSR
0x40001C	Регистр CPSR (не реализован)
0x400020	Регистр системного прерывания DSU
0x400024	Регистр ASI DSU
0x400040	%asr16 (см. «Регистр управления защитой»)
0x400044	%asr17 (см. «Регистр конфигурации процессора»)
0x400048	%asr18 (см. «Инструкции умножителя с накоплением»)
0x40004C	%asr19 (см. «Режим пониженного потребления энергии»)
0x400050 – 0x400054	%asr20, %asr21 (не реализовано)
0x400058 – 0x40005C	%asr22, %asr23 (см. «Счетчик тактов»)
0x400060 – 0x40006C	%asr24 – %asr27 (см. «Аппаратные точки останова/контрольные точки данных»)
0x400060 – 0x40006C	%asr28 – %asr31 (не реализовано)
0x700000 – 0x7FFFFC	Доступ для диагностического контроля ASI (ASI = значение в регистре ASI DSU, адрес = адрес[19:0]) ASI = 0x09: Локальная память инструкций ASI = 0x0B: Локальная память данных ASI = 0x0C: Метки кэша инструкций ASI = 0x0D: Данные кэша инструкций ASI = 0x0E: Метки кэша данных ASI = 0x0F: Данные кэша данных ASI = 0x1E: Раздельные метки отслеживания (snoop tags)

Таблица 4.9.5 – Доступ к данным буфера трассировки инструкций

Адрес	Смещение 0x00	Смещение 0x04	Смещение 0x08	Смещение 0x0C
0x901000000	record_00[127:96]	record_00[95:64]	record_00[63:32]	record_00[31:0]
0x901000010	record_01[127:96]	record_01[95:64]	record_01[63:32]	record_01[31:0]
…	…	…	…	…
0x9010003F0	record_63[127:96]	record_63[95:64]	record_63[63:32]	record_63[31:0]

Таблица 4.9.6 – Доступ к данным буфера трассировки AHB

Адрес	Смещение 0x00	Смещение 0x04	Смещение 0x08	Смещение 0x0C
0x902000000	record_000[127:96]	record_000[95:64]	record_000[63:32]	record_000[31:0]
0x902000010	record_000[159:128]	0x000000000	0x000000000	0x000000000
0x902000020	record_001[127:96]	record_001[95:64]	record_001[63:32]	record_001[31:0]
0x902000030	record_001[159:128]	0x000000000	0x000000000	0x000000000
…	…	…	…	…
0x902001FE0	record_255[127:96]	record_255[95:64]	record_255[63:32]	record_255[31:0]
0x902001FF0	record_255[159:128]	0x000000000	0x000000000	0x000000000

Таблица 4.9.7 – Доступ к данным регистрового файла блока целочисленной арифметики

Адреса	Описание
0x90300000 – 0x9030001C	Окно_7 %l0 – %l7 *
0x90300020 – 0x9030003C	Окно_0 %o0 – %o7 / Окно_7 %i0 – %i7 *
0x90300040 – 0x9030005C	Окно_0 %l0 – %l7
0x90300060 – 0x9030007C	Окно_1 %o0 – %o7 / Окно_0 %i0 – %i7
0x90300080 – 0x9030009C	Окно_1 %l0 – %l7
0x903000A0 – 0x903000BC	Окно_2 %o0 – %o7 / Окно_1 %i0 – %i7
0x903000C0 – 0x903000DC	Окно_2 %l0 – %l7
0x903000E0 – 0x903000FC	Окно_3 %o0 – %o7 / Окно_2 %i0 – %i7
0x90300100 – 0x9030011C	Окно_3 %l0 – %l7
0x90300120 – 0x9030013C	Окно_4 %o0 – %o7 / Окно_3 %i0 – %i7
0x90300140 – 0x9030015C	Окно_4 %l0 – %l7
0x90300160 – 0x9030017C	Окно_5 %o0 – %o7 / Окно_4 %i0 – %i7
0x90300180 – 0x9030019C	Окно_5 %l0 – %l7
0x903001A0 – 0x903001BC	Окно_6 %o0 – %o7 / Окно_5 %i0 – %i7
0x903001C0 – 0x903001DC	Окно_6 %l0 – %l7
0x903001E0 – 0x903001FC	Окно_7 %o0 – %o7 / Окно_6 %i0 – %i7 *
0x90300200 – 0x9030021C	%g0 – %g7 (%g0 всегда считывается как «0»)
0x90300220 – 0x903007FC	Не используется

* Регистровый файл закольцован, реализовано 8 окон (каждое окно состоит из 16 собственных регистров %l0–%l7, %o0–%o7 и доступа к выходным регистрам предыдущего окна через %i0–%i7). Подробное описание приведено в пункте «5.3.1»

4.9.6 Регистры DSU

Регистр управления DSU

Управление DSU осуществляется через регистр управления DSU, что видно из таблицы 4.9.8.

Таблица 4.9.8 – Регистр управления DSU

31 – 12	11	10	9	8	7	6	5	4	3	2	1	0
–	PW	HL	PE	EB	EE	DM	BZ	BX	BS	BW	BE	TE
0	0	0	0	*	*	*	0	0	0	0	0	0
r	r	rw	uc	r	r	r	rw	rw	rw	rw	rw	rw

31 – 12 Зарезервировано

11 Power down (PW) – Устанавливается, если процессор находится в режиме пониженного потребления энергии

10 Остановка процессора (HL) – При считывании данный бит будет установлен, если остановлена работа процессора. Если процессор находится в режиме отладки, то при установке этого бита процессор входит в режим остановки (если этот режим активирован, то при выходе из режима отладки процессор не возобновит выполнение инструкций)

9 Режим ошибки процессора (PE) – Устанавливается, если процессор находится в режиме ошибки; в противном случае сброшен. При записи «1» в данное поле, сбрасывается режим ошибки и остановки

8 Внешнее прерывание (EB) – Значение внешнего сигнала DSUBRE

7 Внешнее разрешение (EE) – Значение внешнего сигнала DSUEN

6 Режим отладки (DM) – Указывает, когда процессор вошел в режим отладки. В этом режиме устанавливается сигнал DHALT, который служит для приостановки работы всех таймеров, включая и сторожевой

5 Остановка в случае системного прерывания ошибки (BZ) – Если бит установлен, процессор входит в режим отладки по всем системным прерываниям, кроме: privileged_instruction, fp_disabled, window_overflow, window_underflow, asynchronous_interrupt, ticc_trap

4 Остановка в случае системного прерывания (BX) – Если бит установлен, процессор входит в режим отладки при возникновении системного прерывания

3 Остановка в программных точках останова (BS) – Если бит установлен, режим отладки принудительно разрешается при выполнении инструкции точки останова (ta 1)

2 Остановка в контрольной точке блока целочисленной арифметики (BW) – Если бит установлен, режим отладки принудительно разрешается в контрольной точке данных блока целочисленной арифметики (системное прерывание 0xb)

1 Остановка при ошибке (BE) – Если бит установлен, процессор принудительно входит в режим отладки, так же, как в случае нахождения процессора в состоянии ошибки (вложенное системное прерывание)

0 Разрешение трассировки (TE) – Трассировка инструкций разрешена. Если бит установлен, инструкции сохраняются в буфере трассировки. Остается установленным, когда процессор входит в режим отладки или ошибок

Регистр прерывания и пошагового режима DSU

Этот регистр используется для осуществления прерывания или пошагового выполнения инструкций процессором, как показано в таблице 4.9.9.

Таблица 4.9.9 – Регистр прерывания и пошагового режима DSU

31 – 17	16	15 – 1	0
–	SS	–	BN
0	0	0	0
r	rw	r	rw

31 – 17 Зарезервировано

16 Пошаговый режим (SS) – Если поле установлено, процессор выполняет одну инструкцию и возвращается в режим отладки. Поле остается установленным после входа процессора в режим отладки

15 – 1 Зарезервировано

0 Немедленное прерывание (BN) – Принудительно вводит процессор в режим отладки, если в регистр управления DSU процессора установлен бит прерывания в контрольной точке (BW). При сбросе процессор возобновляет выполнение операции

Регистр системного прерывания DSU

Регистр системного прерывания DSU предназначен только для чтения и указывает тип системного прерывания SPARC, что показано в таблице 4.9.10, которое принудительно вводит процессор в режим отладки. При принудительном вводе в режим отладки при установке бита BN в регистре управления DSU тип системного прерывания – 0xb (аппаратное прерывание контрольной точки).

Таблица 4.9.10 – Регистр системного прерывания DSU

31 – 13	12	11 – 4	3 – 0
–	EM	TRAPTYPE	–
0	n/r	n/r	0
r	r	r	r

31 – 13 Зарезервировано

12 Режим ошибки (EM) – Устанавливается, если системное прерывание вызвало вход процессора в режим ошибки

11 – 4 Тип системного прерывания (TRAPTYPE) – 8-битовый тип системного прерывания SPARC

3 – 0 Зарезервировано

Счетчик метки времени буферов трассировки

Счетчик метки времени буферов трассировки инкрементируется на каждом такте, пока работает процессор. Счетчик останавливается, когда процессор входит в режим отладки (если не установлен бит разрешения таймера регистра управления буфером трассировки AHB), и перезапускается при возобновлении выполнения операции. Данный регистр представлен в таблице 4.9.11.

Таблица 4.9.11 – Счетчик метки времени буферов трассировки

31 30	29 0
–	TIMETAG
0	0
r	rw

31, 30 Зарезервировано
29 – 0 Значение метки времени DSU

Значение используется как метка времени в буфере трассировки AHB и буфере трассировки инструкции.

Регистр ASI DSU

DSU может обеспечивать доступ для диагностического контроля различных областей ASI (см таблицу 4.9.12). Значение в регистре доступа для диагностического контроля ASI используется как идентификатор ASI при адресации из DSU.

Таблица 4.9.12 – Регистр доступа для диагностического контроля ASI

31 8	7 0
–	ASI
0	n/r
r	rw

31 – 8 Зарезервировано
7 – 0 ASI (ASI) – Идентификатор ASI, используемый для диагностического доступа к области ASI

Регистр управления буфером трассировки AHB

Управление буфером трассировки AHB осуществляется через регистр управления буфером трассировки AHB, что показано в таблице 4.9.13.

Таблица 4.9.13 – Регистр управления буфером трассировки AHB

31 24	23 16	15 9	8	7	6	5	4 3	2	1	0
–	DCNT	–	DF	SF	TE	TF	BW	BR	DM	EN
0	0	0	0	0	0	0	0x1	0	0	0
r	rw	r	rw	rw	rw	rw	r	rw	rw	rw

31 – 24 Зарезервировано
23 – 16 Счетчик для формирования задержки буфера трассировки (DCNT)
15 – 9 Зарезервировано
8 Приостановить таймер в отладочном режиме (DF) – Счетчик тактов продолжает декрементироваться каждый такт, пока в процессоре активирован хотя бы один внутренний таймер. Если данный бит установлен, то счетчик тактов приостанавливается при входе в режим отладки

7	Принудительная выборка (SF) – Если этот бит установлен, он действует на буфер трассировки AHB так же, как установленный на шине сигнал HREADY во время последовательной или непоследовательной передачи. Это означает, что при установке этого бита значения в регистре тестирования буфера трассировки записываются в буфер трассировки, и в регистрах тестируются новые значения. Этот бит автоматически сбрасывается после одного цикла тактовой частоты. Для записи в буфер трассировки также требуется, чтобы он был разрешен (бит EN установлен) и чтобы CPU не находился в режиме отладки или чтобы велась принудительная трассировка (бит TF установлен). Данная функциональность в первую очередь востребована, когда с помощью буфера трассировки осуществляется отслеживание отдельной шины, насчет которой есть подозрение, что она зависла (не происходит штатной установки HREADY по завершению выполнения транзакции AHB)
6	Разрешение таймера (TE) – Счетчик меток времени разрешен, в том числе и в режиме отладки
5	Принудительная трассировка (TF) – Буфер трассировки разрешен, в том числе и в режиме отладки. Следует обратить внимание на то, что буфер трассировки должен быть отключен при считывании данных буфера трассировки через интерфейс регистра ядра
4, 3	Ширина шины (BW) – Значение соответствует $\log_2$ (<Ширина шины> / 32). Аппаратно запрограммированное значение – 0x1 (64-битная шина)
2	Остановка (BR) – Если установлено, процессор входит в режим отладки при остановке буфера трассировки AHB ввиду срабатывания точки останова на шине AHB
1	Режим счетчика задержки (DM) – Указывает, что буфер трассировки находится в режиме счетчика для формирования задержки (более подробное описание приведено в пункте «Регистры точек останова буфера трассировки AHB»)
0	Разрешение трассировки (EN) – Буфер трассировки шины AHB разрешен

Регистр индекса буфера трассировки AHB

Регистр индекса содержит индекс следующей строки буфера трассировки, в которую будет сделана запись, что показано в таблице 4.9.14. Значение данного регистра нельзя напрямую использовать для определения смещения записи в пределах буфера (для этого само значение требуется предварительно увеличить вдвое). Так происходит, поскольку начало одной записи отстоит от начала следующей на 32 байта.

Таблица 4.9.14 – Регистр индексирования буфера трассировки AHB

31 – 12	11 – 4	3 – 0
–	INDEX	–
0x00000	n/r	0
r	rw	r

31 – 12	Зарезервировано
11 – 4	Счетчик индексов буфера трассировки (INDEX) – Следует обратить внимание на то, что приведены только фактически реализованные биты для буфера трассировки AHB размером 256 записей
3 – 0	Зарезервировано

Регистры точек останова буфера трассировки AHB

DSU содержит два регистра точек останова для сопоставления адресам на шине AHB, что проиллюстрировано в таблицах 4.9.15 и 4.9.16. Точка останова используется для фиксации состояния буфера трассировки с помощью автоматического сброса бита разрешения. Остановка может быть задержана путём записи ненулевого значения в поле DCNT регистра управления буфером трассировки AHB. В этом случае значение DCNT будет уменьшаться для каждой дополнительной трассировки, пока не окажется нулевым, после чего произойдет фиксация состояния буфера трассировки. Регистр масок взаимодействует с каждой точкой останова, обеспечивая возможность генерирования прерывания при соответствии блоку адресов. Для детектирования срабатывания точки останова сравниваются только те биты адресов, для которых установлен соответствующий бит маски. Для генерации прерывания при попытке осуществления загрузки или сохранения данных по шине AHB необходимо установить биты LD и/или ST.

Таблица 4.9.15 – Регистр адреса прерывания буфера трассировки AHB

31 – 2	1, 0
BADDR[31 – 2]	–
n/r	0
rw	r

31 – 2 Адрес точки останова (BADDR)
1, 0 Зарезервировано

Таблица 4.9.16 – Регистр маски прерывания буфера трассировки AHB

31 – 2	1	0
BMASK[31 – 2]	LD	ST
n/r	0	0
rw	rw	rw

31 – 2 Маска точки останова (BMASK)
1 Загрузка (LD) – Прерывание на адресе при загрузке данных
0 Сохранение (ST) – Прерывание на адресе при сохранении данных

Регистр управления трассировкой инструкций 0

Регистр управления трассировкой инструкций 0 содержит указатель на следующую запись буфера трассировки инструкций, в которую будет сохранена отладочная информация. Структура данного регистра представлена в таблице 4.9.17.

Таблица 4.9.17 – Регистр управления трассировкой инструкций 0

31 – 28	27 – 6	5 – 0
TFILT	–	ITPOINTER
0	0	0
rw	r	rw

31 – 28 Конфигурация фильтрации трассировки (TFILT)
27 – 6 Зарезервировано
5 – 0 Указатель на трассируемую инструкцию (ITPOINTER) – Следует обратить внимание на то, что приведены только фактически реализованные биты для буфера трассировки инструкций размером 1 Кбайт (на одну запись приходится по 128 байт)

Регистр управления трассировкой инструкций 1

Регистр управления трассировкой инструкций 1 содержит конфигурационную информацию, которая может быть использована для контроля за переполнением буфера трассировки. Регистр можно перезаписывать в процессе функционирования процессора. Его структура представлена в таблице 4.9.18.

Таблица 4.9.18 – Регистр управления трассировкой инструкций 1

31 – 28	27	26 – 24	23	22 – 0
–	WO	TLIM	TOV	–
0	0	0	0	0
r	rw	rw	rw	r

31 – 28 Зарезервировано

27 Контрольная точка при переполнении (WO) – Если данный бит установлен и остановка в контрольной точке блока целочисленной арифметики (BW) разрешена в регистре управления DSU, то контрольная точка будет выставлена, когда будет обнаружено переполнение буфера трассировки (бит TOV данного регистра установится)

26 – 24 Граница трассировки (TLIM) – TLIM сравнивается со старшими битами ITPOINTER регистра управления трассировкой инструкций 0 для генерации значения бита TOV, приведенного ниже

23 Переполнение при трассировке (TOV) – Бит устанавливается, когда DSU обнаруживает, что значение поля TLIM равно трем старшим битам ITPOINTER

22 – 0 Зарезервировано

Регистр счетчика инструкций

DSU содержит регистр счетчика инструкций, позволяющий осуществлять профилирование приложений или генерирование режима отладки после определенного количества тактов или инструкций. Регистр счетчика инструкций представлен в таблице 4.9.19. Регистр счетчика инструкций состоит из 29-разрядного вычитающего счетчика, значение которого уменьшается на каждом такте (IC = 0) или после каждой выполненной инструкции (IC = 1). В режиме профилирования (PE = 1) после обратного переполнения все разряды счетчика устанавливаются, генерации прерывания процессора не происходит. В этом режиме счетчик может периодически опрашиваться в определенном порядке, при этом в отношении CPI (количество тактов на одну инструкцию) формируется статистика. При отключенном профилировании (PE = 0) процессор входит в режим отладки после обратного переполнения счетчика. Это позволяет отладчику выполнять определенное количество инструкций или работать заданное число тактов.

Таблица 4.9.19 – Регистр счетчика инструкций

31	30	29	28 – 0
CE	IC	PE	ICOUNT[28 – 0]
0	0	0	n/r
rw	rw	rw	rw

31 Разрешение счетчика (CE)

30 Счет инструкций (IC) – счет инструкций (1) или тактов (0)

29 Разрешение профилирования (PE)

28 – 0 Счетчик инструкции (ICOUNT)

Регистр управления контрольными точками AHB

DSU содержит две контрольные точки данных AHB, которые могут использоваться для фиксации состояния буфера трассировки AHB или перевода процессора в режим отладки в случае возникновения определенных данных на шине AMBA. Регистр управления контрольными точками AHB представлен в таблице 4.9.20. Эти контрольные точки данных могут быть связаны с двумя точками останова AHB для того, чтобы не происходило срабатывание контрольной точки данных, пока не сработала точка останова AHB. Это также означает, что если контрольная точка данных связана с точкой останова AHB, точка останова не фиксирует состояние буфера трассировки AHB или не вводит процессор в режим отладки, если вместе с точкой останова не сработала контрольная точка данных.

Таблица 4.9.20 – Регистр управления контрольной точкой AHB

31 – 7	6	5	4	3	2	1	0
–	IN	CP	EN	–	IN	CP	EN
0	0	0	0	0	0	0	0
r	rw	rw	rw	r	rw	rw	rw

31 – 7 Зарезервировано

6 Инвертирование (IN) – Инвертирование контрольной точки AHB 2. При установленном бите контрольная точка срабатывает, если данные по шине AHB не соответствуют определенному образцу данных (обычно используется только, если контрольная точка связана с адресом при установке поля CP)

5 Связывание (CP) – Связывание контрольной точки AHB 2 с точкой останова AHB 2

4 Разрешение (EN) – Разрешение контрольной точки AHB 2

3 Зарезервировано

2 Инвертирование (IN) – Инвертирование контрольной точки AHB 1. При установленном бите контрольная точка срабатывает, если данные по шине AHB не соответствуют определенному образцу данных (обычно используется если контрольная точка связана с адресом при установке поля CP)

1 Связывание (CP) – Связывание контрольной точки AHB 1 с точкой останова AHB 1

0 Разрешение (EN) – Разрешение контрольной точки AHB 1

Регистры данных и масок контрольных точек AHB

Регистры данных и масок контрольных точек AHB определяют шаблон данных для контрольных точек AHB (таблицы 4.9.21 и 4.9.22). Контрольная точка данных используется для фиксации состояния буфера трассировки при помощи автоматического сброса бита разрешения. Контрольная точка данных может также использоваться для перевода процессора в режим отладки.

Каждый регистр маски ассоциирован со своим регистром данных. В процессе сравнения участвуют только те биты данных, для которых установлен соответствующий бит маски.

Таблица 4.9.21 – Регистр данных контрольной точки AHB

31	0
DATA[127-n×32 – 96-n×32]	
n/r	
rw	

31–0 Данные контрольной точки AHB (DATA) – Определяют шаблон данных одного слова для контрольной точки AHB. Нижняя часть адреса регистра определяет часть шины, с которой сравнивается значение регистра: Смещение 0x0 определяет значение данных битов шины AHB 127 – 96, 0x4 – для битов 95 – 64, 0x8 – для 63 – 32 и смещение 0xC – для битов 31 – 0

Таблица 4.9.22 – Регистр масок контрольной точки AHB

31	0
MASK[127-n×32 – 96-n×32]	
n/r	
rw	

31–0 Маска контрольной точки AHB (MASK) – Определяет шаблон данных одного слова для контрольной точки AHB. Нижняя часть адреса регистра определяет часть шины, с которой сравнивается значение регистра: Смещение 0x0 определяет значение маски битов шины AHB 127 – 96, 0x4 – для битов 95 – 64, 0x8 – для 63 – 32 и смещение 0xC – для битов 31 – 0

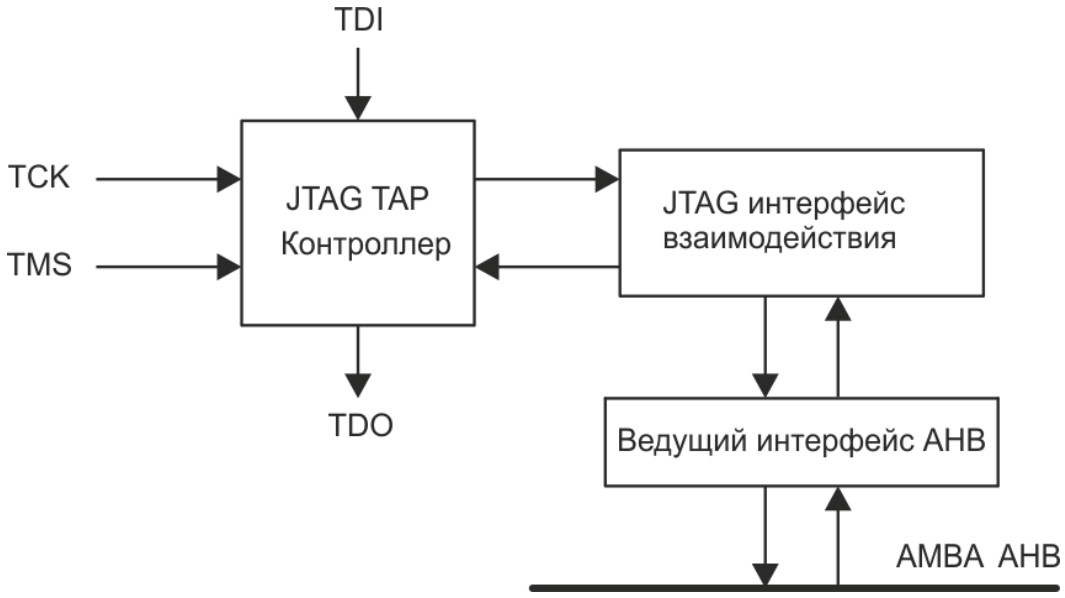
В системе с 64-разрядной шиной записи делаются только в половине регистров данных и масок. Для контрольной точки данных AHB 1 значение данных размером 64 бита записывается в регистры данных контрольной точки AHB при смещениях 0x98 и 0x9C. Соответствующие биты масок устанавливаются в регистрах масок при смещениях 0xA8 и 0xAC.

При осуществлении доступа к данным, размер которых меньше полного размера шины, производится дублирование для заполнения всей шины. Например, 32-разрядный доступ для записи процессора LEON на 64-разрядной шине размещает данные в битах шины 64–32 и 31–0.

4.10 Отладочный интерфейс JTAG

4.10.1 Общие сведения

Через интерфейс системы отладки JTAG (AHBJTAG) обеспечивается доступ к шине AMBA AHB, согласно рисунку 4.10.1. Интерфейс системы отладки JTAG реализует простой протокол, который переводит инструкции JTAG в передачу данных по шине AHB. Через эту систему можно реализовать чтение и запись по любому адресу на шине AHB.



Принятые условные обозначения:
- TDI (Test Data Input) – вход тестовых данных;
- TCK (Test Clock) – тестовая синхронизация;
- TMS (Test Mode Select) – выбор тестового режима;
- TDO (Test Data Output) – выход тестовых данных;
- TAP (Test Access Port) – порт тестового доступа.

Рисунок 4.10.1 – Блок-схема системы отладки JTAG

4.10.2 Принцип работы

Протокол передачи

Система отладки JTAG распознает две инструкции и представлена двумя регистрами: регистром команд/адреса и регистром данных, как показано в таблицах 4.10.1 и 4.10.2. Чтение разрешается подачей команды, состоящей из бита «чтение/запись», данных о размере и адресе AHB-доступа в командно-адресном регистре. Когда чтение AHB выполнено, данные готовы к выдаче из регистра данных. Запись разрешается подачей команды, размера и адреса AHB в командно-адресный регистр и сопровождается записью данных в регистр данных. Последовательная передача выполняется подачей команды и адреса для передачи стартового адреса и подачей бита «SEQ» в регистр данных для последующего доступа. Бит «SEQ» будет инкрементировать AHB адрес для последующего доступа. Последовательные передачи не должны превышать границу в 1 Кбайт. Последовательные передачи всегда выполняются пословно.

Таблица 4.10.1 – Регистр команд/адреса системы отладки JTAG

34	33 32	31 0
W	РАЗМЕР	АДРЕС AHB

34	Запись (W): «0» – чтение, «1» – запись
33, 32	Размер передачи AHB: «00» – байт, «01» – полуслово, «10» – слово, «11» – зарезервировано
31 – 0	Адрес AHB – Адрес на шине AHB

Таблица 4.10.2 – Регистр данных системы отладки JTAG

32	31 0
SEQ	ДАННЫЕ AHB

32	Последовательная передача (SEQ) – Если «1» записывается в эту битовую позицию, когда данные чтения выдаются из регистра данных или данные записи подаются в регистр данных, то последующая передача будет выполнена для адреса следующего слова. При считывании этот разряд будет установлен, если доступ AHB завершен, и сброшен – если доступ AHB не завершен
31 – 0	Данные AHB – данные чтения/записи AHB. Для передачи байта и полуслова упорядочивание данных происходит в порядке следования «сначала старший байт», при этом данные с адресом смещения 0 помещаются в старшие биты

Ядро сигнализирует о завершении доступа AHB, устанавливая разряд 32 в регистре данных. Отладчик может просмотреть разряд 32 из полученных данных для определения успешности доступа. Если разряд 32 установлен, доступ завершен и данные достоверны. Если разряд 32 сброшен, доступ AHB не завершен, когда отладчик начинает считывать данные. В этом случае отладчик может повторять чтение регистра данных, пока разряд 32 не будет установлен, сигнализируя, что данные достоверны и что доступ AMBA AHB завершен.

Следует отметить, что пока разряд 32 читается как «0», новые данные не подаются в регистр данных. Поэтому, чтобы увидеть завершилась ли предыдущая команда, отладчик должен проверить разряд 32 при подаче данных для последовательного доступа AHB. Если разряд 32 сброшен, данные чтения не достоверны и только что загруженная команда сбрасывается ядром.

4.10.3 Тактовая синхронизация

Временная диаграмма и параметры синхронизации показаны на рисунке 4.10.2 и в таблице 4.10.3.

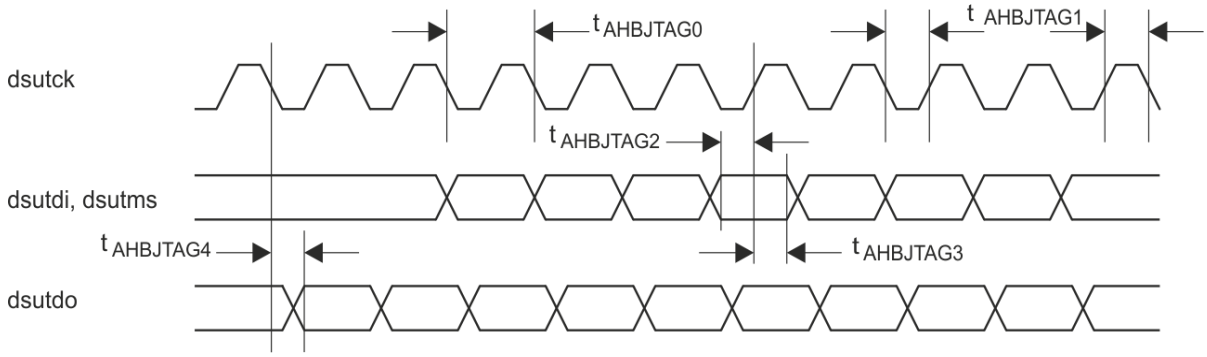


Рисунок 4.10.2 – Временная диаграмма

Таблица 4.10.3 – Параметры тактовой синхронизации

Имя	Параметр	Опорный фронт	Минимум	Максимум	Единица измерения
$t_{AHBJTAG0}$	Тактовый период	–	100	–	нс
$t_{AHBJTAG1}$	Длительность высокого/низкого состояния тактового сигнала	–	40	–	нс
$t_{AHBJTAG2}$	Предустановка данных	Передний фронт dsutck	15	–	нс
$t_{AHBJTAG3}$	Удержание данных	Передний фронт dsutck	0	–	нс
$t_{AHBJTAG4}$	Задержка выдачи данных	Задний фронт dsutck	–	25	нс

4.10.4 Контроллер TAP JTAG

Контроллер TAP JTAG, представленный на рисунке 4.10.3, обеспечивает порт доступа к средствам тестирования согласно стандарту IEEE-1149 (JTAG).

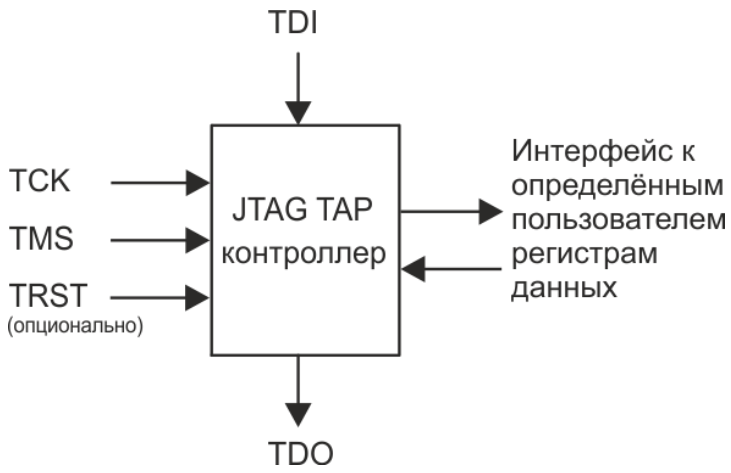


Рисунок 4.10.3 – Блок-схема контроллера TAP

Система реализует сигналы порта доступа к средствам тестирования, синхронную машину конечных состояний TAP, несколько регистров данных JTAG и интерфейс к регистрам данных JTAG, определяемых пользователем.

Контроллер TAP реализует интерфейс порта доступа к средствам тестирования JTAG с сигналами TCK, TMS, TDI и TDO, синхронную машину конечных состояний, соответствующую стандарту IEEE-1149, регистр инструкций JTAG и два регистра данных JTAG: регистр обхода и регистр идентификационного кода устройства. Устройство обеспечивает сдвиг и обновление регистра инструкций JTAG, перевод JTAG в режим обхода (инструкция BYPASS) и выдачу идентификационного номера (инструкция IDCODE). Доступ к определяемым пользователем тестовым регистрам JTAG осуществляется через интерфейс регистра данных, определяемых пользователем.

Доступ к определяемым пользователем тестовым регистрам данных осуществляется через интерфейс регистра данных, определяемых пользователем. В регистр инструкций контроллера TAP загружается инструкция, в регистр данных подаются данные и устанавливаются сигналы, указывающие, что контроллер TAP находится в регистре сбора данных, сдвиговом регистре данных или регистре обновления данных. Логика, управляющая регистрами данных, определяемых пользователем, отображает значение в регистре инструкций и сигналы статуса контроллера TAP для сбора данных, сдвига данных или обновления данных в регистрах.

4.10.5 Регистры TAP контроллера

Реализуется три регистра: регистр инструкций, регистр обхода и регистр идентификационного кода устройства.

4.11 Отладочный интерфейс UART

4.11.1 Общие сведения

Отладочный интерфейс UART (AHBUART) – интерфейс универсального асинхронного приемопередатчика (UART), подключенный к шине AHB AMBA в качестве ведущего устройства. Простой протокол взаимодействия обеспечивает доступ к параметрам и данным. Через линию связи может быть сгенерировано обращение чтения или записи по любому адресу на шине AHB AMBA. На рисунке 4.11.1 представлена блок-схема модуля.

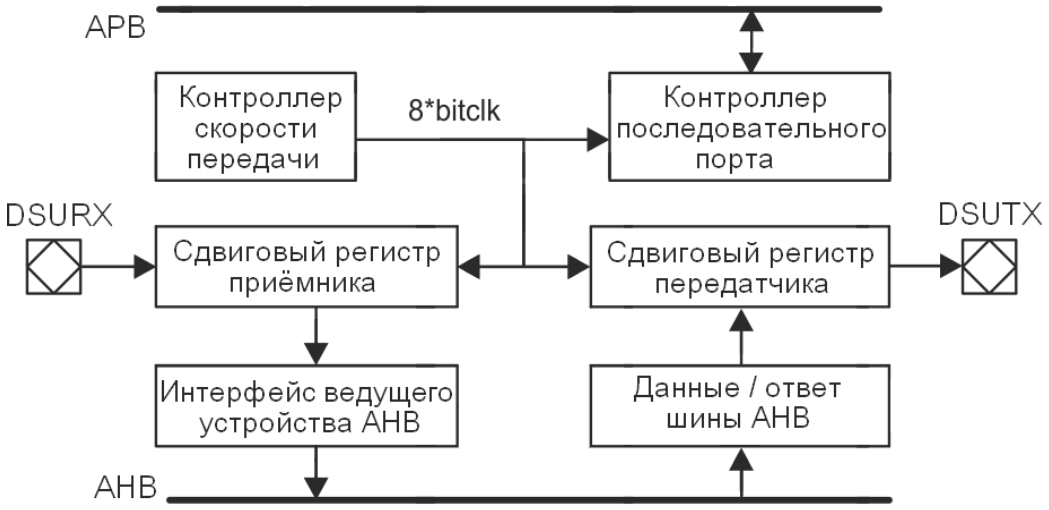


Рисунок 4.11.1 – Блок-схема отладочного интерфейса UART

4.11.2 Принцип работы

Протокол передачи

Интерфейс поддерживает простой протокол передачи, в котором команды состоят из управляющего байта, за которыми следует 32-битный адрес, далее опционально могут располагаться записываемые данные. Данные отправляются в 8-битном базисе, как показано на рисунке 4.11.2.

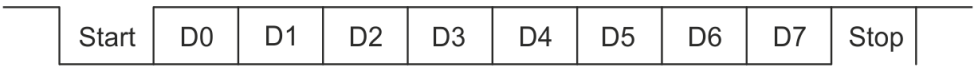


Рисунок 4.11.2 – Кадры данных

Формат команд приведен на рисунке 4.11.3.

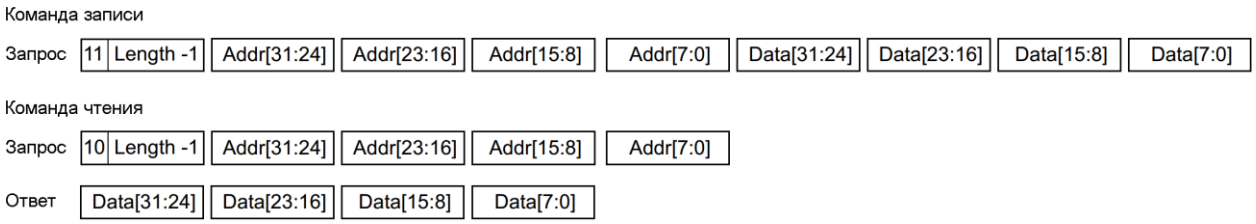


Рисунок 4.11.3 – Формат команд

Поблочная передача данных может быть организована установкой в поле длины значения n – 1, где n определяет количество передаваемых слов. Для запроса записи управляющий байт и адрес посылаются единожды, за ними следует заданное количество записываемых слов. После каждого слова данных адрес автоматически инкрементируется. Для обращений чтения управляющий байт и адрес также посылаются единожды, соответствующее количество слов данных будет передано в ответ.

Генерация скорости передачи

Отладочный интерфейс UART содержит 18-разрядный блок предделителя с декрементом, который задает требуемую скорость передачи. Блок предделителя синхронизируется системным тактовым сигналом и при каждом обратном переполнении генерирует такт UART. Предделитель при обратном переполнении перезагружается значением регистра предделителя UART. Тактовая частота UART должна в 8 раз превышать требуемую скорость передачи.

Если скорость передачи в бодах не задана программным путем, то она будет определена в автоматическом режиме. Это осуществляется путем поиска наименьшего периода между двумя спадающими фронтами полученных данных (соответствующего двум битовым периодам). Когда три идентичных 2-битовых периода обнаружены, соответствующее значение предделителя фиксируется в регистре перезагрузки, в регистре управления AHB UART устанавливается бит BL. Если бит BL сброшен программным путем, то процесс поиска скорости передачи будет перезапущен. Аналогичное поведение также будет наблюдаться, если приемником будет зарегистрирован BREAK или ошибка синхронизации кадров, позволяя изменить скорость передачи внешнему передатчику. Для корректного определения скорости передачи, значение 0x55 должно быть передано в приемник после сброса или после отправки BREAK.

Наиболее подходящее значение предделителя для ручного программирования скорости передачи можно рассчитать по следующей формуле:

$$scaler\ value = \left(\frac{system\_clock\_frequency \times 10}{baud\_rate \times 8} - 5\right) / 10$$

4.11.3 Регистры

Управление интерфейсом осуществляется через регистры, отображенные в адресуемом пространстве APB, согласно таблицам 4.11.1 – 4.11.4.

Таблица 4.11.1 – Регистры AHB UART (базовый адрес 0x80000700)

Смещение адреса APB	Регистр
0x4	Регистр статуса AHB UART
0x8	Регистр управления AHB UART
0xC	Регистр предделителя AHB UART

Регистр статуса AHB UART

Таблица 4.11.2 – Регистр статуса AHB UART

31 – 10	9 – 8	7	6	5	4	3	2	1	0
–	ABC	RX	FE	–	OV	BR	TH	TS	DR
0x000000	0x0	0	0	0	0	0	1	1	0
r	r	r	rw	r	rw	rw	r	r	r

31 – 10 Зарезервировано

9 – 8 Счетчик системы автоматического определения скорости передачи (ABC). Отображает количество последовательных идентичных 2-битовых периодов, используемых для автоматического определения скорости передачи (см. подпункт «Генерация скорости передачи»)

7 Принимаемый бит (RX). Текущее значение принимаемого разряда данных после буфера фильтрации

6 Ошибка синхронизации (FE). Указывает на обнаружение ошибки синхронизации кадров (ошибки кадрирования)

5 Зарезервировано

4 Переполнение (OV). Показывает, что один или несколько символов данных потеряно из-за переполнения

3 Получен символ остановки (BR). Показывает, что получен символ BREAK

2 Регистр удержания передатчика пуст (TH). Показывает, что регистр передатчика, в котором хранятся данные перед отправкой в сдвиговый регистр, пуст

1 Сдвиговый регистр передатчика пуст (TS). Показывает, что сдвиговый регистр передатчика пуст

0 Данные готовы (DR). Указывает на то, что новые данные получены от ведущего интерфейса AHB AMBA

Регистр управления AHB UART

Таблица 4.11.3 – Регистр управления AHB UART

31 – 2	1	0
–	BL	EN
0x00000000	0	0
	r	rw

31 – 2 Зарезервировано

1 Скорость передачи зафиксирована (BL). Бит автоматически устанавливается, когда скорость передачи фиксируется в регистре предделителя

0 Разрешение приема и передачи (EN)

Регистр предделителя AHB UART

Таблица 4.11.4 – Регистр перезагрузки предделителя AHB UART

31 – 19	18 – 0
–	SCALED RELOAD VALUE
0x0000	0x3FFFF
r	rw

31 – 19 Зарезервировано

18–0	Значение перезагрузки предделителя

4.12 Регистры статуса AHB

4.12.1 Общие сведения

Регистры статуса (блок AHBSTAT) хранят информацию об ошибках доступа к AHB AMBA. Существуют регистр статуса AHB и регистр адреса сбоя AHB. Они захватывают управляющие и адресные значения сигналов о сбое транзакции шины AMBA, а также фиксируют присутствие корректируемой ошибки, сигнал о которой поступает от сбоеустойчивого ядра. Доступ к регистру статуса и регистру адреса сбоя осуществляется с шины APB AMBA.

4.12.2 Принцип работы

Ошибки

Регистры осуществляют контроль над транзакциями шины AHB AMBA и хранят во внутренней памяти текущие значения HADDR, HWRITE, HMASTER и HSIZE. Мониторинг всегда активен после запуска и сброса до обнаружения сообщения об ошибке (HRESP = «01»). При обнаружении ошибки содержимое регистра адреса сбоя AHB и регистра статуса AHB фиксируется, устанавливается разряд New Error (NE). Одновременно генерируется прерывание, как описано ниже.

Исправимые ошибки

Не только сообщения об ошибке на шине AHB (HRESP = «01») детектируются. Модули сбоеустойчивых контроллеров внешней и внутренней памяти по выделенным линиям сигнализируют и об исправимых ошибках. Последовательность действий модуля AHBSTAT та же самая, что и для сообщения об ошибке на шине AHB, за исключением дополнительной установки бита корректируемой ошибки (CE) регистра статуса AHB.

Прерывания

Прерывание поступает в контроллер прерывания для информирования процессора об условиях возникновения ошибки. В обычном порядке программа обработки прерывания обрабатывает ошибку при помощи информации в регистрах статуса. После этого она сбрасывает разряд NE, и мониторинг активируется снова. Прерывания генерируются при сообщениях об ошибке AMBA и для корректируемых ошибок, как описано выше.

4.12.3 Регистры

Размещение и формат регистров, отображаемых в адресном пространстве APB, приведены в таблицах 4.12.1 – 4.12.3.

Таблица 4.12.1 – Регистры статуса AHB (базовый адрес 0x80000000)

Смещение адреса APB	Регистры
0x0	Регистр статуса AHB
0x4	Регистр адреса сбоя AHB

Регистр статуса AHB

Таблица 4.12.2 – Регистр статуса AHB

31 – 10	9	8	7	6 – 3	2 – 0
–	CE	NE	HWRITE	HMASTER	HSIZE
0x000000	0	0	n/r	n/r	n/r
r	rw*	rw*	r	r	r

31 – 10 Зарезервировано.
9 Корректируемая ошибка (CE). Устанавливается, если обнаруженная ошибка является корректируемой, иначе ноль. Бит может быть сброшен записью нуля.
8 Новая ошибка (NE). Бит сбрасывается при запуске и после перезагрузки. Устанавливается при обнаружении ошибки. Бит может быть сброшен записью нуля.
7 Сигнал HWRITE транзакции AHB, которая вызвала ошибку.
6 – 3 Сигнал HMASTER транзакции AHB, которая вызвала ошибку.
2 – 0 Сигнал HSIZE транзакции AHB, которая вызвала ошибку.

Регистр адреса сбоя AHB

Таблица 4.12.3 – Регистр адреса сбоя AHB

31 – 0
AHB FAILING ADDRESS
n/r
r

31–0 Сигнал HADDR транзакции AHB, которая вызвала ошибку

4.13 Блок таймеров общего назначения

4.13.1 Общие сведения

Блок таймеров общего назначения (GPTIMER) обеспечивает общее предделение частоты и декремент таймеров. В процессорах 1906ВМ016, 1906ВМ01А6 реализовано два блока таймеров, по два и три таймера для GPTIMER 1 и GPTIMER 2 соответственно. Каждый блок действует как подчиненное устройство на шине APB AMBA. Разрядность делителя частоты – 16 бит. Разрядность таймеров – 32 разряда. Последний таймер блока GPTIMER 1 является сторожевым. Каждый таймер в состоянии генерировать отдельное прерывание при обратном переполнении, в соответствии с пунктом «4.8.3». Блок GPTIMER 2 генерирует прерывания, относящиеся к расширенной области, см. пункт «4.8.2 Принцип работы», подпункт «Расширенные прерывания». Структурная схема блока таймеров общего назначения представлена на рисунке 4.13.1.

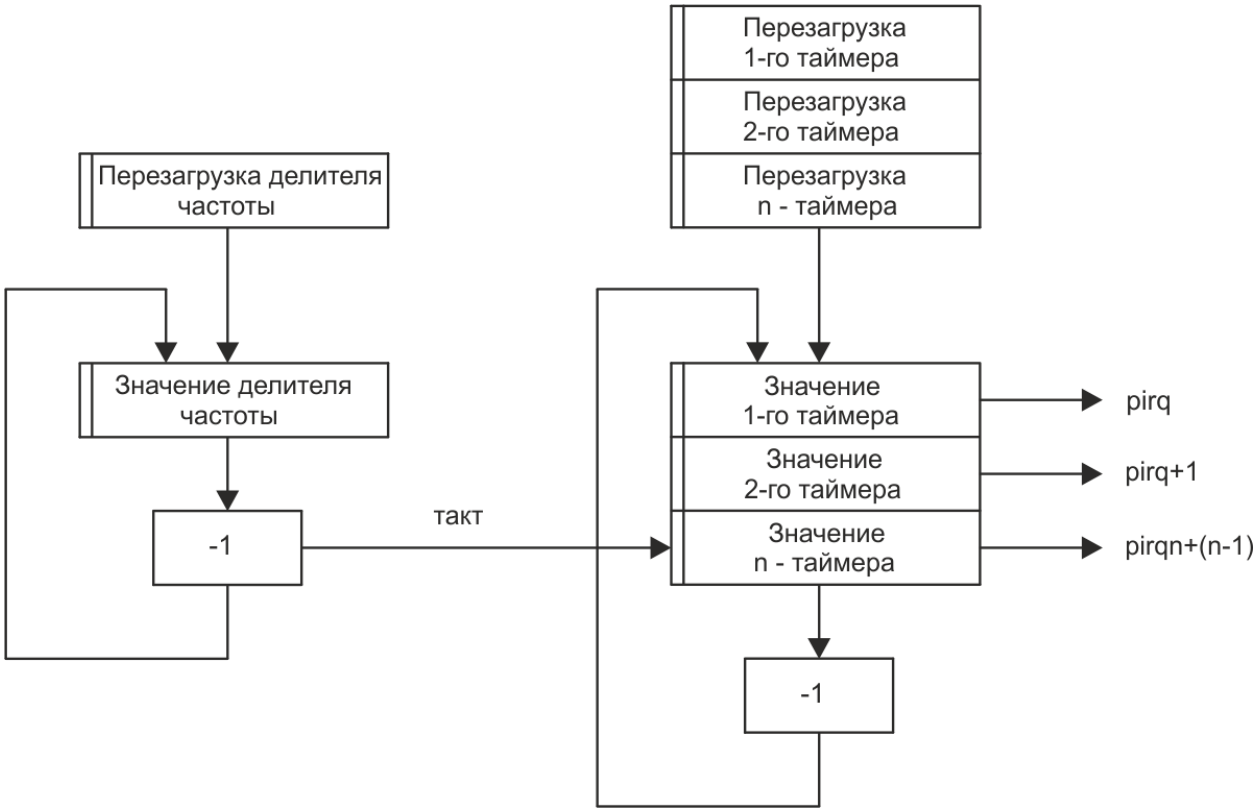


Рисунок 4.13.1 – Структурная схема блока таймеров общего назначения

4.13.2 Принцип работы

Предделитель частоты тактируется системным тактовым сигналом, его значение декрементируется с каждым тактом. При обратном переполнении предделителя частоты он перезагружается из регистра перезагрузки предделителя частоты, и генерируется такт таймера.

Управление работой каждого таймера осуществляется через регистр управления. Таймер разрешен при установке бита разрешения (EN) в регистре управления таймером. Значение таймера декрементируется на каждом такте предделителя частоты. Если в регистре управления установлен бит перезапуска (RS), то при обратном переполнении таймера он автоматически перезагружается значением соответствующего регистра значений перезагрузки таймера. При сброшенном бите RS он останавливается на значении 0xFFFFFFFF, бит EN сбрасывается.

Каждый из таймеров в состоянии сгенерировать прерывания на выделенной ему линии при обратном переполнении счетчика таймера, если установлен бит разрешения прерывания (IE). Бит ожидания обработки прерывания (IP) устанавливается в регистре управления таймером при обратном переполнении и останется в таком состоянии, пока не будет сброшен записью «1» в данное поле.

Чтобы минимизировать сложность, таймеры в одном блоке совместно используют одно устройство декрементирования. Это означает, что минимальный допустимый коэффициент деления предделителя частоты TIMERS + 1 (значение регистра перезагрузки предделителя = TIMERS), где TIMERS – количество реализованных в блоке таймеров (соответствующее значения поля регистра конфигураций блока). При установке в регистре управления бита цепи (CH) таймер n можно объединить в цепь с предыдущим таймером n – 1. Таймер n будет декрементироваться каждый раз при обратном переполнении таймера n – 1.

Каждый таймер можно в любое время перезагрузить со значением из регистра перезагрузки таймера при установке бита загрузки (LD) регистра управления таймером.

Каждый таймер блока GPTIMER2 может быть сконфигурирован на захват данных из регистра значений счетчика таймера при возникновении системного прерывания. Для фильтрации прерываний используется специальный регистр маски.

При сбросе все таймеры, кроме сторожевого, отключаются. Разряды значения предделителя частоты и регистров перезагрузки устанавливаются в состояние логической единицы. Значение сторожевого таймера по сбросу – 0x0000FFFF. Все остальные регистры не инициализируются.

4.13.3 Регистры

Программирование модулей осуществляется через регистры, отображенные в адресуемом пространстве APB, см. таблицы 4.13.1 – 4.13.10. Количество реализованных регистров зависит от количества реализованных таймеров. Базовые адреса блоков таймеров GPTIMER 1 и GPTIMER 2 – 0x80000300 и 0x80000900 (в соответствии с данными таблицы 2.5.1).

Таблица 4.13.1 – Регистры блока таймеров общего назначения (GPTIMER 1, базовый адрес 0x80000300)

Смещение адреса APB	Регистр
0x00	Регистр значения предделителя
0x04	Регистр значения перезагрузки предделителя
0x08	Регистр конфигураций
0x0C	Не используется
0x10	Регистр значений счетчика таймера 1
0x14	Регистр значений перезагрузки таймера 1
0x18	Регистр управления таймером 1
0x1C	Не используется
0x20	Регистр значений счетчика таймера 2 (сторожевой таймер)
0x24	Регистр значений перезагрузки таймера 2 (сторожевой таймер)
0x28	Регистр управления таймером 2 (сторожевой таймер)

Таблица 4.13.2 – Регистры блока таймеров общего назначения (GPTIMER 2, базовый адрес 0x80000900)

Смещение адреса APB	Регистр
0x00	Регистр значения предделителя
0x04	Регистр значения перезагрузки предделителя
0x08	Регистр конфигураций
0x0C	Регистр конфигураций захвата
0x10	Регистр значений счетчика таймера 1
0x14	Регистр значений перезагрузки таймера 1
0x18	Регистр управления таймером 1
0x1C	Регистр захвата 1
0x20	Регистр значений счетчика таймера 2
0x24	Регистр значений перезагрузки таймера 2
0x28	Регистр управления таймером 2
0x2C	Регистр захвата 2
0x30	Регистр значений счетчика таймера 3
0x34	Регистр значений перезагрузки таймера 3
0x38	Регистр управления таймером 3
0x3C	Регистр захвата 3

Регистр значения предделителя

Таблица 4.13.3 – Регистр значения предделителя

31 16	15 0
–	SV
0x0000	0xFFFF
r	rw

31 – 16 Зарезервировано
15 – 0 Значение предделителя (SV)

Регистр значения перезагрузки предделителя

Таблица 4.13.4 – Регистр значения перезагрузки предделителя

31 16	15 0
–	SRV
0x0000	0xFFFF
r	rw

31 – 16 Зарезервировано
15 – 0 Значение перезагрузки предделителя (SRV)

Регистр конфигураций

Таблица 4.13.5 – Регистр конфигураций

31 12	11	10	9	8	7 3	2 0
–	EL	–	DF	SI	IRQ	TIMERS
0x00000	0	0	0	1	*	*
r	rw*	r	rw	r	r	r

31 – 12 Зарезервировано
11 Разрешение захвата (EL). Если бит установлен, то при следующем совпадении прерывания с маской регистра конфигураций захвата в регистры захвата будет отправлено соответствующее значение счетчика таймера. Затем бит автоматически сбрасывается, значение таймера не будет перезагружено до тех пор, пока он не будет установлен вновь. Функционал доступен только для блока таймеров GPTIMER 2, для таймеров GPTIMER 1 данное поле доступно только для чтения
10 Зарезервировано
9 Отключает приостановку таймера (DF). Если установлено, таймер не прерывает свою работу при входе процессора в режим отладки, в противном случае сигнал DHALT останавливает работу блока таймеров
8 Отдельные прерывания (SI). Установлено, если генерируются отдельные прерывания для каждого таймера; в противном случае – сброшено
7 – 3 Прерывание (IRQ): Определяет линию прерывания, которую будет использовать первый таймер данного блока таймеров. Поскольку реализовано использование раздельных линий прерываний, таймер n будет генерировать запрос на прерывание по линии IRQ + n. Для GPTIMER 1 значение равняется 7, для GPTIMER 2 – 20 (см. таблицу 4.8.1)
2 – 0 Количество реализованных таймеров (TIMERS). Для GPTIMER 1 значение равняется 2, для GPTIMER 2 – 3

Регистр конфигураций захвата

Таблица 4.13.6 – Регистр конфигураций захвата

31	0
LCTRL	
0x00000000	
rw	

31 – 0 Конфигурация захвата (LCTRL). Задает, какие биты шины прерывания будут вызывать захват значения счетчика таймера при возникновении прерывания. Функционал доступен только для блока таймеров GPTIMER 2, для таймеров GPTIMER 1 данное поле доступно только для чтения и имеет значение 0

Регистр значений счетчика таймера

Таблица 4.13.7 – Регистр значений счетчика таймера

31	0
TCV	
0x00000000	
rw	

31 – 0 Значение счетчика таймера (TCV). Декрементируется при каждом такте предделителя частоты.

Регистр значений перезагрузки таймера

Таблица 4.13.8 – Регистр значений перезагрузки таймера

31	0
TCRV	
*	
rw	

31 – 0 Значение перезагрузки таймера (TCRV). Это значение загружается в счетчик таймера при записи «1» в бит загрузки (LD) регистра управления таймером либо, при обратном переполнении, если был установлен бит RS. Значение по сбросу для сторожевого таймера – 0x0000FFFF, для остальных таймеров оно равно 0xFFFFFFFF

Регистр управления таймером

Таблица 4.13.9 – Регистр управления таймером

31 – 9	8	7	6	5	4	3	2	1	0
–	WS	WN	DH	CH	IP	IE	LD	RS	EN
0x000000	0	0	0	0	0	0	0	*	*
r	rw*	rw*	r	rw	wc	rw	rw	rw	rw

31 – 9 Зарезервировано

8 Запретить вывод сторожевого таймера (WS): Если бит установлен, то вывод WDOGN# будет не активен (на нем всегда будет выставлен высокий логический уровень). Функционал доступен только для сторожевого таймера, для остальных таймеров данное поле доступно только для чтения

7	Разрешить немаскируемое прерывание (WN): Если бит установлен, то сторожевой таймер также будет генерировать немаскируемое прерывание (по линии 15), когда подается запрос на прерывание. Функционал доступен только для сторожевого таймера, для остальных таймеров данное поле доступно только для чтения
6	Остановка в режиме отладки (DH): Данный бит отображает значение сигнала, используемого для остановки работы счетчика, пока система находится в режиме отладки (внутренний сигнал DHALT от модуля DSU)
5	Цепь (CH): Организует связь по цепочке с предыдущим таймером блока. Если установлено для таймера n, таймер n декрементируется каждый раз при обратном переполнении таймера (n – 1)
4	Ожидается обработка прерывания (IP): модуль устанавливает этот бит, когда подается сигнал прерывания. Этот бит остается установленным, пока не будет сброшен записью «1» в данное поле, запись «0» не имеет никакого воздействия
3	Разрешение прерывания (IE): Если установлено, таймер генерирует прерывание при обратном переполнении
2	Загрузка (LD): Отправить значение из регистра перезагрузки таймера в регистр значений счетчика таймера
1	Перезапуск (RS): Если установлено, регистр значений счетчика таймера перезагружается значением регистра перезагрузки при обратном переполнении таймера. Значение по сбросу для сторожевого таймера – 1, для остальных таймеров – 0
0	Разрешение (EN): Если бит установлен, то таймер разрешен. Значение по сбросу для сторожевого таймера – 1, для остальных таймеров – 0

Регистр захвата

Таблица 4.13.10 – Регистр захвата

31	0
LTCV	
0x00000000	
r	

31 – 0	Захваченное значение счетчика таймера (LTCV). Функционал доступен только для блока таймеров GPTIMER 2, для таймеров GPTIMER 1 данное поле доступно только для чтения и имеет значение 0

4.14 Порт ввода-вывода общего назначения

4.14.1 Общие сведения

Система портов ввода-вывода общего назначения (GRGPIO) является 16-разрядной и поддерживает прерывания.

Каждый разряд порта универсальной системы ввода-вывода может быть отдельно настроен на работу в качестве входа или выхода. Линии с 1 по 15 могут дополнительно генерировать прерывание. Для генерации прерывания вход может фильтроваться по полярности и быть настроенным на обнаружение уровня или фронта сигнала.

На рисунке 4.14.1 представлена схема одной линии области ввода-вывода.

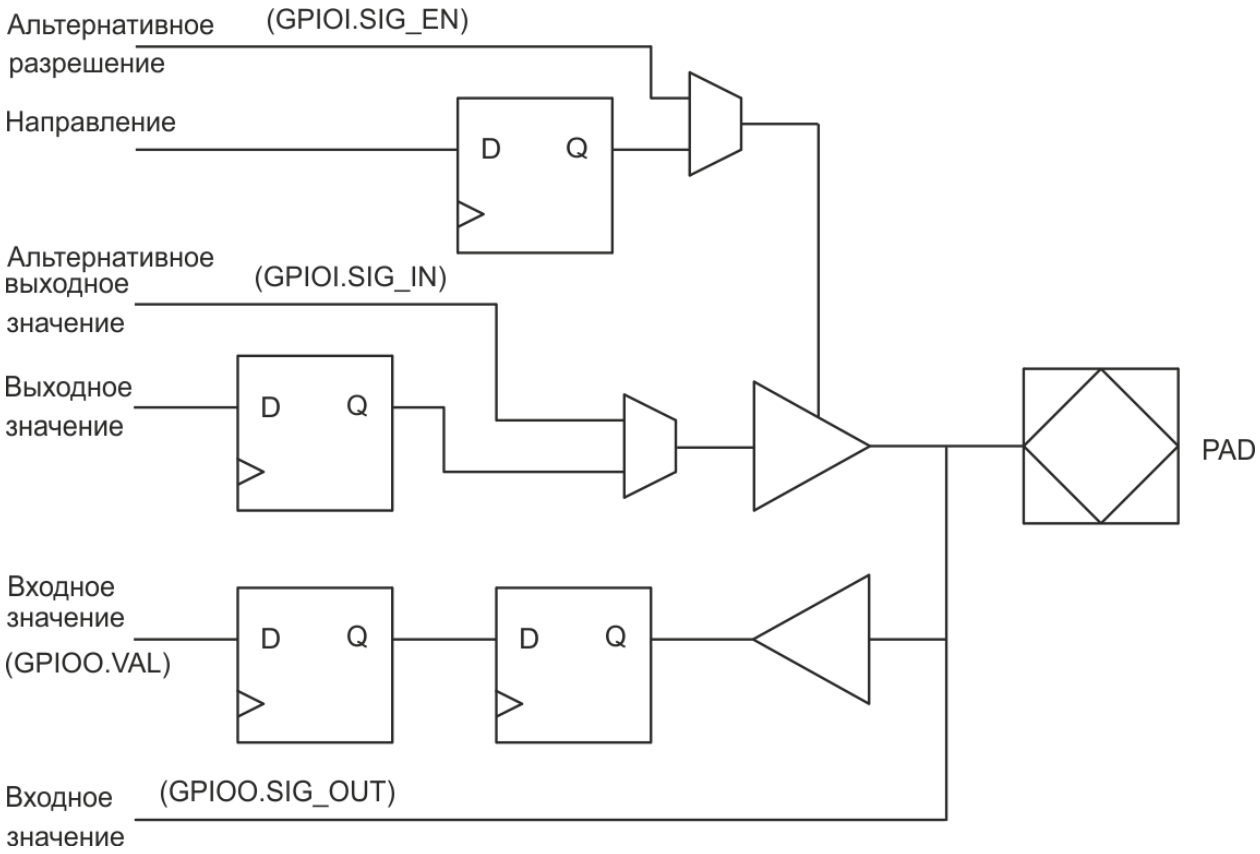


Рисунок 4.14.1 – Схема порта ввода-вывода общего назначения

4.14.2 Принцип работы

Порты ввода-вывода реализованы как двунаправленные буферы с программируемым разрешением выхода. Вход каждого буфера синхронизируется двумя последовательными триггерами для исключения потенциальной метастабильности. Синхронизируемые значения можно считывать из регистра данных порта ввода-вывода. Разрешение выхода порта ввода-вывода управляется регистром направления. Установка «1» в соответствующей битовой позиции разрешает выходной буфер для соответствующей линии области ввода-вывода. Задающее значение берется из выходного регистра порта ввода-вывода.

Реализовано фиксированное назначение линий прерывания. При данной конфигурации для каждой линии области ввода-вывода номер прерывания равен индексу линии области ввода-вывода плюс смещение от первой линии прерывания, pirq (PIO[1] соответствует прерыванию pirq+1, и т.д.). В текущей реализации pirq = 0, соответственно линии с 1 по 15 генерируют прерывания с уровнями от 1-го до 15-го. Нулевое прерывание зарезервировано под системные нужды. Также значение pirq можно считать из регистра области plug & play модуля GPIO, доступного по адресу 0x801FF020 (см. рисунок 4.23.2).

Генерация прерывания управляется тремя регистрами: маски, полярности и фронтов. Чтобы разрешить прерывание, необходимо установить соответствующий бит в регистре маски прерываний. Если значение регистра фронтов прерываний – «0», то прерывание рассматривается как прерывание по уровню. Если значение регистра полярности – «0», активный уровень прерывания низкий. Если значение регистра полярности прерываний – «1», активный уровень прерывания высокий. Если значение регистра фронтов – «1», прерывание запускается фронтом сигнала. Далее с помощью регистра полярности определяется, будет ли осуществляться генерация прерывания по

переднему («1») или по заднему («0») фронту. Стоит отметить, что захват прерывания осуществляется по положительному фронту тактового сигнала процессора вне зависимости от того используется ли прерывание по фронту или по уровню (т.е. не гарантируется генерация прерывания по сигналу с длительностью менее периода системной тактовой частоты). Основное отличие данных режимов проявляется, если после выхода из обработчика предыдущего прерывания на линии остался прежний уровень сигнала. Тогда при генерации по уровню будет повторно сгенерировано прерывание. В той же ситуации для формирования прерывания по фронту требуется явное переключение состояния вывода (в соответствии со значением в регистре полярности).

4.14.3 Регистры

Система программируется через регистры, отображенные в адресуемом пространстве APB, приведенные в таблицах 4.14.1 – 4.14.8

Таблица 4.14.1 – Регистры портов ввода-вывода (базовый адрес 0x80100400)

Смещение адреса APB	Регистр
0x00	Регистр данных порта ввода-вывода
0x04	Выходной регистр порта ввода-вывода
0x08	Регистр направления порта ввода-вывода
0x0C	Регистр маски прерываний
0x10	Регистр полярности прерываний
0x14	Регистр фронтов прерываний
0x18	Зарезервировано
0x1C	Регистр поддерживаемого функционала

Регистр данных порта ввода-вывода

Таблица 4.14.2 – Регистр данных порта ввода-вывода

31 … 16	15 … 0
–	Входное значение порта I/O
0x0000	*
r	rw

31–16 Зарезервировано
15–0 Входное значение порта ввода-вывода

Выходной регистр порта ввода-вывода

Таблица 4.14.3 – Выходной регистр порта ввода-вывода

31 … 16	15 … 0
–	Выходное значение порта I/O
0x0000	0x0000
r	rw

31–16 Зарезервировано
15–0 Выходное значение порта ввода-вывода

Регистр направления порта ввода-вывода

Таблица 4.14.4 – Регистр направления порта ввода-вывода

31 16	15 0
–	Значение направления порта I/O
0x0000	0x0000
r	rw

31–16 Зарезервировано
15–0 Значение направления порта ввода-вывода
(0 = выход отключен, 1 = выход включен)

Регистр маски прерываний

Таблица 4.14.5 – Регистр маски прерываний

31 16	15 0
–	Маска прерываний
0x0000	0x0000
r	rw

31–16 Зарезервировано
15–0 Маска прерываний
(0 = прерывание маскировано, 1 = прерывание включено)

Регистр полярности прерываний

Таблица 4.14.6 – Регистр полярности прерываний

31 16	15 0
–	Полярность прерываний
0x0000	n/r
r	rw

31–16 Зарезервировано
15–0 Полярность прерываний
(0 = низкий уровень/задний фронт, 1 = высокий уровень/передний фронт)

Регистр фронтов прерываний

Таблица 4.14.7 – Регистр фронтов прерываний

31 16	15 0
–	Фронт прерываний
0x0000	0x0000
r	rw

31–16 Зарезервировано
15–0 Фронт прерываний (0 = уровень, 1 = фронт)

Регистр поддерживаемоего функционала

Таблица 4.14.8 – Регистр поддерживаемого функционала

31 19	18	17	16	15 13	12 8	7 5	4 0
–	PU	IER	IFL	–	IRQGEN	–	NLINES
0x0000	0	0	0	0x0	0x0	0x0	0x0F
r	r	r	r	r	r	r	r

31–19 Зарезервировано

18 Реализован регистр импульсов. Бит сброшен, поскольку в модуле отсутствует регистр импульсов.
Поле доступно в модуле GPIO, начиная с версии 2 и выше

17 Реализован регистр разрешений ввода. Бит сброшен, поскольку в модуле отсутствует регистр разрешений входов.
Поле доступно в модуле GPIO, начиная с версии 2 и выше

16 Реализация регистра флагов прерывания. Бит сброшен, поскольку в модуле отсутствуют регистр доступных прерываний и регистр флагов прерываний (смещение 0x40 и 0x44).
Поле доступно в модуле GPIO, начиная с версии 2 и выше

15–13 Зарезервировано

12–8 Настройки генерации прерываний (IRQGEN). Данное поле доступно только для чтения. Если IRQGEN = 0, то линия с номером n будет управлять прерыванием pirq + n. Значение pirq может быть прочитано из области plug & play данного модуля

7–5 Зарезервировано

4–0 Порядковый номер старшей линии GPIO (NLINES) – значение поля на единицу меньше количества линий GPIO (только для чтения)

4.15 Интерфейс UART

4.15.1 Общие сведения

Интерфейс универсального асинхронного приемопередатчика (APBUART) предназначен для последовательной передачи данных. В процессорах 1906BM016, 1906BM01A6 реализованы два UART: UART1 и UART2. Интерфейсом поддерживается размер кадра с 8 битами данных, одним дополнительным битом четности и одним или двумя стоповыми битами. Для генерации скорости передачи у каждого UART имеется программируемый 12-разрядный тактовый предделитель.

Для передачи данных между шиной APB и UART используются два буфера FIFO (обслуживание в порядке поступления).

Глубина каждого буфера FIFO – 4 байта.

Для передачи данных между шиной APB и UART используются два регистра временного хранения информации, как показано на рисунке 4.15.1.

Аппаратное управление потоком не поддерживается.

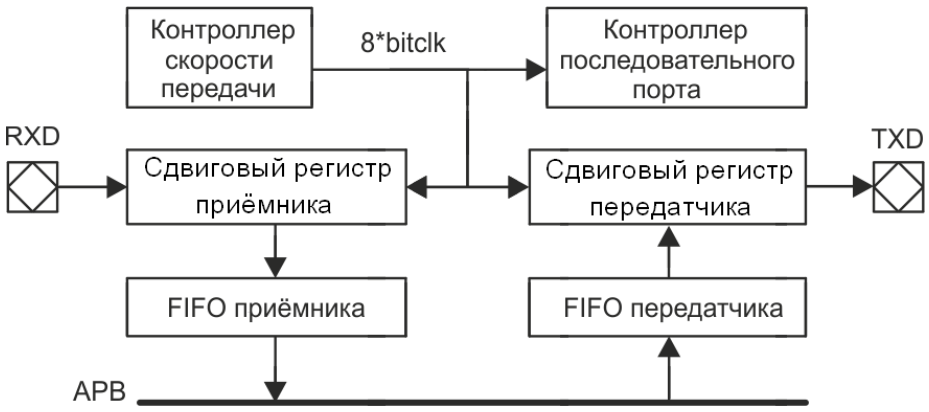


Рисунок 4.15.1 – Блок-схема интерфейса UART

4.15.2 Принцип работы

Работа передатчика

Передатчик активируется при помощи бита TE в регистре управления UART. Данные, которые необходимо передать, сохраняются в FIFO при записи в регистр данных UART (см. рисунок 4.15.2). Данные, готовые для передачи, передаются из FIFO передатчика, в сдвиговый регистр передатчика и преобразуются в последовательный поток на выводе передатчика (TXD). Автоматически передается стартовый бит, за которым следуют 8 бит данных, бит четности (не является обязательным) и один или два стоповых бита. Сначала передается младший значащий бит данных.

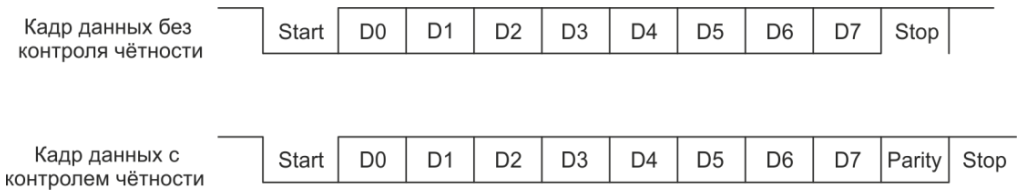


Рисунок 4.15.2 – Кадры данных UART

После передачи стопового бита, если не доступны новые данные в FIFO передатчика, вывод выхода последовательных данных передатчика остается в высоком логическом состоянии и в регистре статуса UART устанавливается бит TS, сообщающий что сдвиговый регистр передатчика пуст. При загрузке нового символа в FIFO передатчика передача данных возобновляется, бит TS сбрасывается. Если FIFO передатчика пуст, в регистре статуса устанавливается бит TE. Отключение передатчика немедленно прекращается любую активную передачу, включая символ, который в текущий момент выдается наружу сдвиговым регистром передатчика. В FIFO передатчика не следует загружать данные, если передатчик отключен или FIFO заполнен. В противном случае данные могут быть перезаписаны, один или несколько кадров потеряны.

Бит статуса TF (не путать с битом управления TF) устанавливается, если FIFO передатчика в данный момент заполнен. Бит TH устанавливается, если FIFO заполнен меньше, чем наполовину (меньше половины записей в FIFO содержат данные). Если установлен бит управления TF, то разрешена генерация прерываний FIFO передатчика. Регистр статуса также содержит счетчик TCNT, который показывает текущее количество записей в FIFO передатчика.

Работа приемника

Приемник активируется для получения данных с помощью бита разрешения приемника (RE) в регистре управления UART. Приемник пытается обнаружить стартовый бит – изменение логического уровня сигнала с высокого на низкий на выводе последовательного входа данных приемника. Если такое переключение было выявлено, то повторный опрос состояния последовательного входа осуществляется спустя половину бита. Если выборка показала, что последовательный вход в состоянии логической единицы, то стартовый бит недействителен, и поиск корректного стартового бита продолжается. Если последовательный вход в состоянии логического нуля, то стартовый бит признается действительным и приемник продолжает осуществлять выборку с последовательного входа с интервалом в 1 бит (в теоретическом центре бита), пока не будет принято надлежащее количество битов данных, бит четности и не будет обнаружен один стоповый бит. Сигнал с последовательного входа пропускается через фильтр мета-стабильности, представляющий из себя схему мажорирования трех последовательных значений, сохраняемых каждый отсчет UART (см. пункт «4.15.3»).

В приемнике имеется собственный FIFO, который идентичен FIFO передатчика.

Первым принимается самый младший бит. Затем данные передаются в FIFO приемника и в регистре статуса UART, как только FIFO будет содержать, по крайней мере, один кадр данных, устанавливается бит готовности данных (DR). Биты ошибок четности, кадрирования и переполнения устанавливаются на границе принимаемого байта, это происходит одновременно с установкой бита готовности данных приемника. При обнаружении ошибки кадр данных не сохраняется в FIFO. Кроме того, новые биты статуса ошибки, прежде чем они будут сохранены в регистре статуса, объединяются по схеме ИЛИ со старыми значениями. Таким образом, они не сбрасываются, пока не будут записаны нули через доступ по шине APB AMBA. Если FIFO приемника и сдвиговые регистры заполнены при обнаружении нового стартового бита, то данные, сохраненные в сдвиговом регистре приемника, будут потеряны и в регистре статуса UART будет установлен бит переполнения (OV). Бит получен символ остановки (BR) устанавливается при получении BREAK, который является разновидностью ошибки синхронизации (линия удерживается в состоянии логического нуля дольше ожидаемой длительности кадра).

Бит статуса RF (не путать с битом управления RF) устанавливается при заполнении FIFO приемника. Бит статуса RH устанавливается, если FIFO заполнен наполовину (по крайней мере, половина записей FIFO занята кадрами данных). Если установлен разряд управления RF, прерывание FIFO приемника разрешено. Регистр статуса также содержит счетчик RCNT, в котором показывается текущее количество кадров данных в FIFO приемника.

4.15.3 Генерация скорости передачи

Каждый UART содержит 12-разрядный блок предделителя с обратным счетом, который задает требуемую скорость передачи. Блок предделителя синхронизируется системным тактовым сигналом и при каждом обратном переполнении генерирует отсчет UART. Предделитель при обратном переполнении перезагружается значением регистра предделителя UART. Частота следования отсчетов UART должна быть в 8 раз больше требуемой скорости передачи.

Формула для вычисления значения предделителя под требуемую скорость передачи с использованием целочисленного деления и отбрасыванием остатка:

$$scaler\ value\ = \frac{system\_clock\_frequency}{baud\_rate\ \times\ 8\ +\ 7}$$

Для вычисления точного значения используется формула:

$$scaler\ value\ = \frac{system\_clock\_frequency}{baud\_rate\ \times\ 8}\ -\ 1$$

Входы внешнего тактирования интерфейсов UART не выведены в качестве выводов процессора. На них постоянно подается уровень логического нуля. Установка бита EC равносильна отключению тактирования интерфейса UART.

4.15.4 Режим обратной петли

Если в регистре управления UART установлен бит LB, UART находится в режиме обратной петли (Loopback mode). В этом режиме выход передатчика подключен к входу приемника. Режим позволяет провести loopback тесты, выполняемые для проверки корректности работы приемника, передатчика и взаимодействующих с ними подпрограмм программного обеспечения. В этом режиме выходы остаются в неактивном состоянии во избежание передачи данных.

4.15.5 Режим отладки FIFO

Чтобы войти в режим отладки FIFO, необходимо в регистре управления UART установить бит режима отладки (DB). В этом режиме можно считать FIFO передатчика и записать FIFO приемника через регистр отладки FIFO UART. Выход передатчика в режиме отладки неактивен. Запись в FIFO приемника генерирует прерывание, если разрешено прерывание приемника.

4.15.6 Генерация прерываний

В интерфейсе UART реализовано две разновидности прерываний: нормальные прерывания и прерывания FIFO.

Нормальные прерывания передатчика генерируются, когда разрешены прерывания передатчика (установлен бит TI), разрешена передача (установлен бит TE в регистре управления UART) и FIFO передатчика перестает содержать данные, становясь пустым (устанавливается бит TE в регистре статуса UART). Прерывания FIFO генерируются, когда разрешены прерывания FIFO передатчика (установлен бит TF), разрешена передача и если UART заполнен меньше чем наполовину (т.е. установлен бит статуса TH). Это прерывание по уровню, сигнал прерывания непрерывно удерживается в состоянии логической единицы, пока выполняются условия.

Прерывания приемника происходят аналогичным образом. Нормальные прерывания приемника генерируются, когда разрешены прерывания приемника (установлен бит RI), разрешен прием (установлен бит RE в регистре управления UART) и либо FIFO приемника содержит данные (устанавливается бит DR), либо получен символ данных с ошибкой по четности (устанавливается бит PE в регистре статуса UART), либо произошли переполнение (устанавливается бит OV) или ошибка синхронизации кадров (устанавливается бит FE). Прерывание FIFO приемника генерируется, когда разрешены прерывания FIFO приемника (RF), разрешен прием (RE) и наполовину заполнен FIFO. Сигнал прерывания удерживается в состоянии логической единицы, пока FIFO приемника наполовину заполнен (по крайней мере, половина записей содержит кадры данных).

Для сокращения количества прерываний, генерируемых интерфейсом, реализован режим отложенных прерываний приемника. Он активируется установкой бита разрешения

отложенных прерываний (DI). Когда бит DI установлен, после приема символа каждый раз запускается внутренний таймер, а прерывание генерируется, только если в течение 384 отсчетов UART (сопоставимо по длительности с четырьмя кадрами) не был получен новый кадр данных (полученным он считается после первого стопового бита). Одновременно с запуском таймера устанавливается внутренний флаг ожидания генерации прерывания в связи с получением символа. Если прерывания FIFO приемника разрешены, и оно происходит, внутренний флаг ожидания генерации прерывания будет сброшен, так как символ, вызывающий состояние ожидание формирования прерывания, уже находится в FIFO и драйвер уведомляется о нем через прерывание FIFO.

Также реализовано отдельное прерывание по символу остановки (BREAK). Если данный режим разрешен (установлен бит BI), прерывание генерируется немедленно, как только получен BREAK, даже если разрешены отложенные прерывания приемника. Если прерывание по BREAK не разрешено, но активирован режим отложенных прерываний, прерывание при приеме BREAK не генерируется.

Если задержка прерываний запрещена, поведение будет тем же самым для бита прерывания остановки, за исключением того, что прерывание будет генерироваться для символов остановки, если разрешены прерывания приемника, даже если прерывание остановки запрещено.

Сдвиговый регистр передатчика также может служить источником прерываний. Интерфейс будет генерировать прерывание каждый раз, когда сдвиговый регистр становится пустым (бит TS переключается в состояние установлен).

4.15.7 Регистры

Управление устройством осуществляется через регистры, отображенные в адресуемом пространстве APB, согласно таблицам 4.15.1 – 4.15.6.

Таблица 4.15.1 – Регистры UART (базовые адреса 0x80000100 и 0x80000600)

Смещение адреса APB	Регистр
0x00	Регистр данных UART
0x04	Регистр статуса UART
0x08	Регистр управления UART
0x0C	Регистр предделителя UART
0x10	Регистр отладки FIFO UART

Регистр данных UART

Таблица 4.15.2 – Регистр данных UART

31 – 8	7 – 0
–	DATA
0x000000	n/r
r	rw

31 – 8 Зарезервировано
7 – 0 FIFO приемника (при выполнении считывания)
7 – 0 FIFO передатчика (при выполнении записи)

Регистр статуса UART

Таблица 4.15.3 – Регистр статуса UART

31 – 26	25 – 20	19 – 17	16 – 11	10	9	8	7	6	5	4	3	2	1	0
RCNT	TCNT	–	FD	RF	TF	RH	TH	FE	PE	OV	BR	TE	TS	DR
0x00	0x00	0x0	0x04	0	0	0	0	0	0	0	0	1	1	0
r	r	r	r	r	r	r	r	rw	rw	rw	rw	r	r	r

31 – 26 Счетчик FIFO приемника (RCNT) – Показывает количество кадров данных в FIFO приемника

25 – 20 Счетчик FIFO передатчика (TCNT) – Показывает количество кадров данных в FIFO передатчика

19 – 17 Зарезервировано

16 – 11 Глубина FIFO (FD) – Количество кадров, которые могут одновременно находиться в FIFO приемника, в FIFO передатчика

10 FIFO приемника заполнен (RF) – Если бит установлен, то FIFO приемника заполнен

9 FIFO передатчика заполнен (TF) – Если бит установлен, то FIFO передатчика заполнен

8 FIFO приемника заполнен наполовину (RH) – Если бит установлен, то, по крайней мере, половина FIFO занято данными

7 FIFO передатчика заполнен наполовину (TH) – Если бит установлен, то FIFO заполнен менее чем наполовину

6 Ошибка кадрирования (FE) – Бит устанавливается, если обнаружена ошибка синхронизации кадров

5 Ошибка четности (PE) – Бит устанавливается, если обнаружена ошибка четности

4 Переполнение (OV) – Бит устанавливается, если некоторые данные были потеряны из-за переполнения

3 Получен символ остановки (BR) – Бит устанавливается при получении символа BREAK

2 FIFO передатчика пуст (TE) – Если бит установлен, FIFO передатчика пуст

1 Сдвиговый регистр передатчика пуст (TS) – Если бит установлен, то сдвиговый регистр передатчика пуст

0 Данные готовы (DR) – Если бит установлен, то в FIFO приемника доступны новые данные

Регистр управления UART

Таблица 4.15.4 – Регистр управления UART

31	30 – 16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FA	–	NS	SI	DI	BI	DB	RF	TF	EC	LB	FL	PE	PS	TI	RI	TE	RE
1	0x0000	n/r	n/r	n/r	n/r	n/r	n/r	n/r	0	n/r	0	n/r	n/r	n/r	n/r	0	0
r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

31 FIFO доступны (FA) – Бит установлен, поскольку вместо пары регистров временного хранения информации реализованы FIFO приемника и FIFO передатчика

30 – 16 Зарезервировано

15 Количество стоп-битов (NS) – Если бит установлен, то будет использоваться два стоп бита, в противном случае – один. Не сбрасывается

14 Разрешение прерывания пустого сдвигового регистра передатчика (SI) – Если бит установлен, то при опустошении сдвигового регистра передатчика генерируется прерывание (подробное описание приведено в пункте «4.15.6 Генерация прерываний»). Не сбрасывается

13 Разрешение отложенных прерываний (DI) – Если бит установлен, активирован режим отложенных прерываний приемника. В данном режиме прерывание при получении данных генерируется, только если в течение интервала времени длительностью в 384 отсчетов UART не будет получен новый кадр данных. Подробное описание приведено в пункте 4.15.6 Генерация прерываний». Данный функционал применим, если установлено разрешение прерывания приемника. Не сбрасывается

12 Разрешение прерывания остановки (BI) – Если бит установлен, генерируется прерывание при каждом получении символа BREAK. Подробное описание приведено в пункте «4.15.6 Генерация прерываний». Не сбрасывается

11 Разрешение режима отладки FIFO (DB) – Если бит установлен, можно считать и записать регистр отладки FIFO. Не сбрасывается

10 Разрешение прерывания FIFO приемника (RF) – Если бит установлен, прерывание по уровню от FIFO приемника разрешено. Не сбрасывается

9 Разрешение прерывания FIFO передатчика (TF) – Если бит установлен, прерывание по уровню FIFO передатчика разрешено. Не сбрасывается

8 Внешнее тактирование (EC) – Внешнее тактирование не реализовано, установка данного бита равносильна отключению тактирования UART

7 Режим обратной петли (LB) – Если бит установлен, режим обратной петли активирован. Не сбрасывается

6 Управление потоком (FL) – Если бит установлен, разрешено управление потоком передачи данных через использование CTS/RTS (не реализовано)

5 Разрешение контроля по четности (PE) – Если бит установлен, разрешена генерация и проверка бита четности. Не сбрасывается

4 Выбор полярности четности (PS) – Выбор полярности контроля по четности (0 = четность, 1 = нечетность). Не сбрасывается

3 Разрешение прерывания передатчика (TI) – Если бит установлен, при передаче символа данных генерируется прерывание (подробное описание приведено в пункте «4.15.6 Генерация прерываний). Не сбрасывается

2 Разрешение прерывания приемника (RI) – Если бит установлен, при приеме символа данных генерируется прерывание (подробное описание приведено в пункте «4.15.6 Генерация прерываний»). Не сбрасывается

1 Разрешение передатчика (TE) – Если бит установлен, передача разрешена

0 Разрешение приемника (RE) – Если бит установлен, прием разрешен

Регистр предделителя UART

Таблица 4.15.5 – Регистр перезагрузки предделителя UART

31 12	11 0
–	SCALED RELOAD VALUE
0x00000	n/r
r	rw

31 – 12 Зарезервировано

11–0 Значение перезагрузки предделителя

Регистр отладки FIFO UART

Таблица 4.15.6 – Регистр отладки FIFO UART

31 8	7 0
–	DATA
0x000000	n/r
r	rw

31 – 8 Зарезервировано
7 – 0 FIFO приемника (при выполнении записи)
7 – 0 FIFO передатчика (при выполнении считывания)

4.16 Интерфейс MIL-STD-1553/AS15531

4.16.1 Общие сведения

В состав микропроцессоров 1906ВМ016 и 1906ВМ01А6 входят два интерфейса Mil-STD-1553B (0, 1) с базовыми адресами 0x80100200 и 0x80100300, соответственно.

Интерфейсы реализованы на блоках GR1553B.

Интерфейс устройства соединяет шину AHB/APB AMBA с шиной MIL-STD-1553 (ГОСТ Р 52070-2003) с одиночным или с двойным резервированием и работает как контроллер шины, удаленный терминал или монитор шины.

MIL-STD-1553 – контроллер шины для передачи данных максимум 32 устройствам через общий дифференциальный проводник, с двойным резервированием. Шина разработана для прогнозируемой работы в реальном времени и обеспечения ошибкоустойчивости. Исходная скорость передачи данных на шине – 1 Мбит/с при максимальной скорости передачи данных полезной нагрузки приблизительно 770 Кбит/с.

Контроллер шины (BC) является одним из терминалов на шине, контролирующим весь поток обмена на шине. Другие терминалы – удаленные терминалы (RTs), которые работают при получении команд от контроллера шины. Каждый RT имеет уникальный адрес между 0 – 30. Кроме того, к шине могут быть присоединены пассивные мониторы шины (BMs), регистрирующие информацию о транзакциях на шине.

На рисунке 4.16.1 представлен вариант топологии шины MIL-STD-1553.

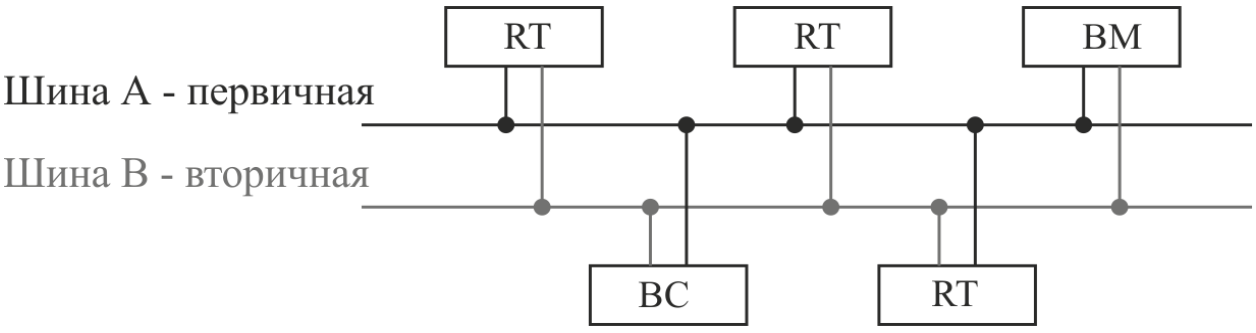


Рисунок 4.16.1 – Топология шины MIL-STD-1553

Существует пять возможных типов передачи данных на шине MIL-STD-1553:
- передача из BC в RT («прием»);
- передача из RT в BC («передача»);
- передача из RT до RT;
- широковещательная трансляция из BC в RTs;
- широковещательная трансляция из RT в RTs.

Каждая передача содержит (1–32) слова данных по 16 битов каждое.

Контроллер шины также может отправлять «коды режима» в RTs для выполнения административных задач, таких как синхронизация и считывание статуса терминала.

4.16.2 Электрический интерфейс

Устройство соединяется с шиной MIL-STD-1553 через одиночный или двойной трансивер, изолирующие трансформаторы и трансформаторную или шлейфовую связь, как показано на рисунке 4.16.2. Если используется однократное резервирование, неиспользуемые сигналы P/N, принимаемые шиной, должны быть притянуты оба либо к высокому логическому уровню, либо к низкому логическому уровню. Сигнал разрешения трансмиттера обычно инвертирован: он называется – «запрещение передатчика» (txinh).

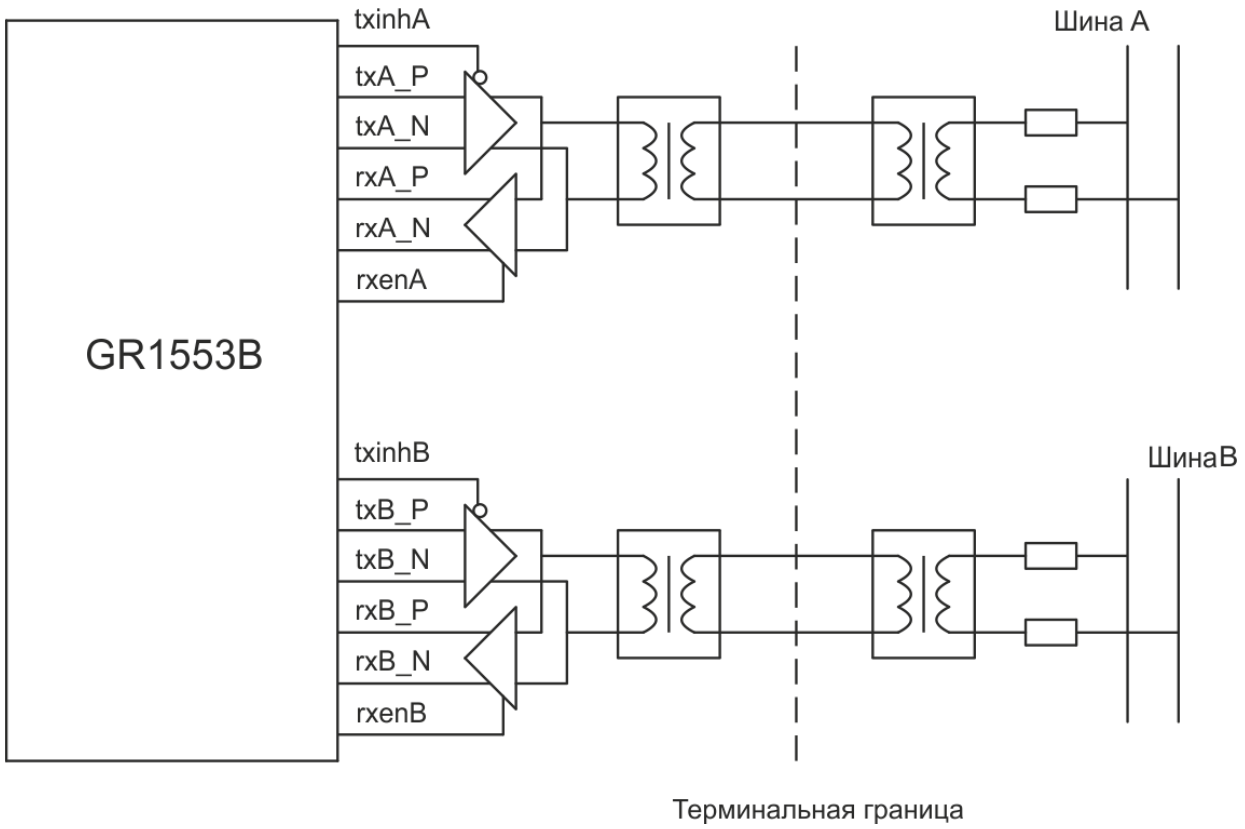


Рисунок 4.16.2 – Интерфейс устройства и шины MIL-STD-1553 (с двойным резервированием, с трансформаторной связью)

4.16.3 Принцип работы

Режимы работы

Устройство содержит три отдельных блока: контроллер шины (BC), удаленный терминал (RT) и монитор шины (BM) с общим кодеком 1553.

Управление режимом работы устройства осуществляется путем запуска и остановки частей BC/RT/BM через регистры записи. В начале работы с устройством работа блоков не разрешена и устройство не активно на шине 1553 и шине AMBA.

Части устройства BC и RT не могут быть одновременно активными на шине 1553. Если BC запущен или приостановлен, доступ к шине 1553 имеет только BC (и возможно BM); RT может только получать и отвечать на команды только при полной остановке схем BC.

Монитор шины только «прослушивает» трафик на шине и поэтому работает независимо от того, разрешены или не разрешены другие части.

Регистр интерфейса

Конфигурация и управление устройства осуществляется через регистры управления, доступ к которым обеспечивается по шине APB. Каждая часть – BC, RT, BM имеет отдельный набор регистров, а также имеется небольшой набор общих регистров.

Некоторые поля регистра управления BC и RT защищены «ключом» – полем данного регистра, в которое записывается определенное значение для обеспечения записи в регистр. Пользователи RT/BM используют ключи в целях обеспечения безопасного доступа к регистрам при разрешении BC или изменении адреса RT. Если программное обеспечение построено без знания ключа к определенному регистру, случайное выполнение записи с правильным ключом к регистру управления маловероятно.

Прерывание

Устройство имеет один выход прерывания, который генерируется несколькими разными первичными событиями. События, которые вызывают прерывание, определяются с помощью регистра разрешения маски прерывания.

Кодек MIL-STD-1553

Внутренний кодек устройства получает и передает слова данных по шине 1553, а также генерирует и проверяет шаблоны и четность синхронизации.

Логика проверки «обратной петли» контролирует, чтобы каждое передаваемое слово было «видимо» на входах приема. Если переданное слово не отображается, передатчик останавливается и передает сигнал ошибки, который направляется пользователю.

4.16.4 Работа контроллера шины

Общие сведения

При работе в качестве контроллера шины устройство выступает ведущим на шине MIL-STD-1553, запускает и выполняет передачу.

Этот режим работает на базе запланированного списка (плана) передач. Программное обеспечение устанавливает в памяти последовательность дескрипторов и переходов передач, буферы для отправленных и принятых данных и кольцевой буфер указателей прерывания. При запуске плана (через запись регистра работы BC) устройство обрабатывает план, выполняет последовательные передачи и записывает статус в план передачи и входящие данные в соответствующие буферы.

Временной контроль

В каждом дескрипторе плана есть поле «слот времени». Если планируемая передача завершается раньше времени, запланированного на эту передачу, устройство будет находиться в состоянии паузы до выполнения следующей команды. Это позволяет пользователю точно осуществлять временной контроль сообщений в коммуникационном потоке.

Если передача использовала больше времени, чем заложенное для нее в плане, это время будет вычитаться из слота времени следующей команды и так далее. Устройство может отслеживать до одной секунды заимствованного времени и не вставляет паузы до

момента установления положительного баланса, кроме пауз между сообщениями согласно требованиям стандарта.

Для максимально быстрого выполнения плана передач необходимо установить все слоты времени в «0». Для группировки передач можно сдвинуть все интервалы ответа к последней передаче.

План передач можно остановить или перевести в состояние ожидания при соответствующей записи в регистре выполнения контроллера шины. При переводе в состояние ожидания отсчет времени плана по-прежнему будет учитываться, чтобы планируемое время при активизации плана не нарушилось. При остановке плана таймеры сбрасываются.

Если в дескрипторе следующей передачи установлен бит «extsync», запуск команды начинается с приходом фронта на внешнем входе синхронизации. Таймер планирования и баланс слота времени сбрасываются, и запускается команда. Если импульс «sync» поступает до передачи, он сохраняется, и команда сразу запускается. Активатор передачи очищается при остановке плана (не при переводе в состояние ожидания). Также активатор устанавливается/очищается программным обеспечением через регистр выполнения контроллера шины.

Выбор шины

Каждый дескриптор передачи имеет бит выбора шины, с помощью которого выбирается одна из двух резервных шин («0» – шина A, «1» – шина B) для передачи.

Другим способом управления шины является регистр перестановки per-RT, в котором содержится по одному биту для каждого адреса RT. Регистр перестановки шины является опциональным. Программное обеспечение может проверить поля регистра BCFEAT с доступом только для чтения на предмет его наличия в устройстве.

При установке бита в регистре перестановки per-RT значение бита выбора шины инвертируется для всех передач для соответствующего RT – при сбросе выбирается шина «B», при установке – шина «A». Это позволяет переключить все передачи в один или набор RT через другую шину с единственным регистром записи без возможности изменения дескрипторов.

Аппаратное обеспечение определяет соответствующую шину для использования, выполняя операцию исключающего ИЛИ бита регистра перестановки шины и бита выбора шины. В обычном режиме для каждого RT выбирается один из следующих способов: бит выбора шины всегда сброшен и используется регистр перестановки или бит регистра перестановки всегда сброшен и используется бит выбора шины.

Если регистр перестановки шины используется для выбора шины, бит указателя резервной шины может быть разрешен для автоматического обновления регистра, в зависимости от итога передачи. Если передача на шине A прошла успешно, бит регистра перестановки шины сбрасывается; если передача успешно прошла на шине B, бит регистра перестановки шины устанавливается. Если передача не прошла, значение в регистре перестановки шины меняется на противоположное.

Второй список передачи

Устройство может быть настроено на работу со вторым «асинхронным» списком передачи в ходе выполнения обычного плана. До окончания интервала ответа команды плана устройство проверяет, чтобы интервал ответа следующей асинхронной передачи был меньше, чем остальное время режима ожидания. В этом случае предусмотрена асинхронная команда.

Если асинхронная команда завершается не по плану, время заимствуется из следующей команды в обычном плане. Чтобы не нарушать обычный план, интервал ответа для асинхронных сообщений должен иметь пессимистичное значение.

Исключающий бит в указателе передачи устанавливается, если не требуется выполнять асинхронную команду в режиме ожидания следующей передачи.

Асинхронные сообщения не планируются во время ожидания импульса синхронизации или при переводе графика в состояние ожидания и при истечении текущего интервала ответа, поскольку старт следующей запланированной команды не определен.

Генерация прерывания

Каждая команда в плане передач может генерировать прерывание после завершения соответствующих передач, выполненных с ошибкой или без нее. Недействительные дескрипторы команд всегда генерируют прерывания и останавливают план. Перед генерацией прерывания, активированного передачей, адрес к соответствующему дескриптору записывается в кольцевой буфер запросов на прерывание ВС, при этом кольцевой регистр позиции запросов на прерывание, активированных передачей ВС, инкрементируется.

Отдельное прерывание ошибки сигнализирует об ошибках DMA. При возникновении ошибки DMA во время считывания/записи указателей работающий план переводится в состояние ожидания. Ошибки DMA в буферах данных приводят к отказу соответствующей передачи с записью кода ошибки (см. 4.16.4).

Какие из этих прерывающих событий в действительности вызывают запрос на прерывание на шине AMBA, контролируется настройками регистра разрешения запросов на прерывание.

Формат списка передачи

Список передачи ВС представляет собой массив дескрипторов передач и переходов, представленных в таблице 4.16.1. Здесь и далее по тексту в таблицах сокращение «R/W» эквивалентно использованию «Считывание/запись». Каждое значение выравнивается по границе 128 бит (16 байт). Два неиспользуемых слова в переходе можно использовать для сохранения арбитражных данных.

Таблица 4.16.1 – Формат дескриптора передачи GR1553B

Смещение	Значение для дескриптора передачи	DMA R/W	Значение для перехода	DMA R/W
0x00	Слово «0» дескриптора передачи (см. таблицу 4.16.2)	R	Условное слово (см. таблицу 4.16.6)	R
0x04	Слово «1» дескриптора передачи (см. таблицу 4.16.3)	R	Адрес команды перехода, выравнивается до 128 бит	R
0x08	Указатель буфера данных, выравнивается до 16 бит. Для буферов записи, если установлен бит 0, полученные данные, не учитываются, и ссылка игнорируется. Это используется для передач из RT в RT, в которых ВС не обрабатывает передаваемые данные	R	Не используется	–
0x0C	Итоговое слово, записываемое устройством (см. таблицу 4.16.4)	W	Не используется	–

Структура слов дескриптора передачи показана в таблицах 4.16.2 – 4.16.4.

Таблица 4.16.2 – Слово «0» дескриптора передачи BC GR1553B (смещение 0x00)

31	30	29	28	27	26	25	24	2322	20	19	18	17	1615	0
0	WTRIG	EXCL	IRQE	IRQN	SUSE	SUSN	RETMD		NRET	STBUS	GAP	–		STIME

31 Должно быть сброшено для идентификации в качестве дескриптора

30 Ожидание внешнего активатора (WTRIG)

29 Исключительный интервал ответа (EXCL) – Асинхронные сообщения не запланированы

28 Запрос на прерывание после передачи с ошибкой (IRQE)

27 Обычный запрос на прерывание (IRQN) – Прерывание после передачи

26 Перевод в режим ожидания при ошибке (SUSE) – Переводит план в режим ожидания (или останавливает список асинхронной передачи) при обнаружении ошибки

25 Обычный перевод в режим ожидания (SUSN) – Перевод в режим ожидания после передачи

24, 23 Режим повторной передачи (RETMD). «00» – повтор на основной шине, «01» – чередующийся повтор на обоих шинах, «10» – повтор сначала на основной шине, затем на альтернативной шине, «11» – зарезервировано, не используется

22–20 Количество повторов (NRET) – Количество автоматических повторов на одну шину.

Общее количество повторов (включая первую попытку) равно NRET + 1 для RETMD = 00 или (2 × (NRET + 1)) для RETMD = 01/10

19 Запоминание шины (STBUS) – Если передача прошла успешно и данный бит установлен, шина, в которой передача прошла успешно (0 для шины A, 1 для шины B), запоминается в регистре перестановки шины per–RT. Если передача не прошла и данный бит установлен, в регистр перестановки запоминается альтернативная шина (только при поддержке маски шины per–RT). Для получения дополнительной информации – подробнее см. выше в подпункте «Выбор шины»

18 Промежуток между сообщениями (GAP) – Если установлено, после передачи добавляется дополнительное время, соответствующее полю RTTO

17, 16 Зарезервировано

15–0 Интервал ответа (STIME) – Выделенное время в интервалах по 4 микросекунды. Если планируемая передача завершается раньше, чем это определено значением STIME, устройство приостанавливает работу перед подачей новой команды до исчерпания времени интервала ответа

Таблица 4.16.3 – Слово «1» дескриптора передачи BC GR1553B (смещение 0x04)

31	30	29 26	25 21	20 16	15 11	10	9 5	4 0
DUM	BUS	RTTO	RTAD2	RTSA2	RTAD1	TR	RTSA1	WCMC

31 Ложная передача (DUM) – Если бит установлен, трафик на шине не генерируется и немедленно передается сообщение – «успешное выполнение передачи». Для ложных передач действуют установки EXCL, IRQN, SUSN, STBUS, GAP, STIME; другие биты и ссылка буфера данных игнорируются

30 Выбор шины (BUS) – Шина для передачи; «0» – шина A, «1» – шина B

29–26 Тайм-аут RT (RTTO) – Дополнительный тайм-аут RT, превышающий номинальный, в интервалах по 4 микросекунды (0000 – 14 мкс, 1111 – 74 мкс).

Примечание – Это дополнительное время также используется, как дополнительный промежуток между сообщениями, если установлен бит GAP

25–21 Адрес второго RT для передачи из RT в RT (RTAD2)

20–16 Подадрес второго RT для передачи из RT в RT (RTSA2)
Примечание – Для получения дополнительной информации по установке RTAD1, RTSA1, RTAD2, RTSA2, WCMC, TR для разных типов передач – см. таблицу 4.16.5
15–11 Адрес первого RT (RTAD1)
10 Передача/прием (TR)
9–5 Подадрес первого RT (RTSA1)
4–0 Счетчик слов/код режима (WCMC)
Примечание – Следует отметить, что биты 15 – 0 соответствуют (первому) слову команды на шине 1553

Таблица 4.16.4 – Итоговое слово дескриптора передачи GR1553B (смещение 0x0C)

31	30–24	23–16	15–8	7–4	3	2–0
0	–	RT2ST	RTST	RETCNT	–	TFRST

31 Всегда сброшено
30–24 Зарезервировано
23–16 Биты статуса RT 2 (RT2ST) – Биты статуса принимающего RT в передаче из RT в RT, в противном случае сброшено. Набор бит статуса такой же, как и в поле RTST
15–8 Биты статуса RT (RTST) – Биты статуса передающего RT в передаче из RT в RT. 15 – ошибка сообщения, 14 – установка аппаратного бита или зарезервированного бита, 13 – запрос на обслуживание, 12 – прием команды транслирования, 11 – бит занятости, 10 – флаг подсистемы, 9 – прием динамического управления шины, 8 – терминальный флаг
7–4 Счетчик повторов (RETCNT) – Количество выполненных повторов
3 Зарезервировано – Массив маски после считывания для обеспечения согласованности
2–0 Статус передачи (TFRST) – Результат последней попытки:
000 – успешно (или был установлен ложный бит);
001 – RT не ответил (передающий RT в передаче из RT в RT);
010 – принимающий RT в передаче из RT в RT не ответил;
011 – принятое слово статуса RT имеет ошибку сообщения, установлен бит занятости или зарезервированный бит;
100 – ошибка протокола (неверно синхронизированные слова данных, ошибка декодера, неверное количество слов данных);
101 – недействительный указатель передачи;
110 – тайм-аут DMA буфера данных или сообщение об ошибке;
111 – передача прервана из-за ошибки проверки «обратной петли».
Примечание – Код ошибки 011 выдается в случае, если количество слов данных совпадает с количеством успешно переданных, в противном случае используется код 100. Код ошибки 011 может быть выдан при правильно выполненной команде «передача последней команды» или «передача последнего слова статуса», поскольку эти команды не сбрасывают слово статуса

Биты конфигурации передачи BC GR1553B для различных типов передачи показаны в таблице 4.16.5.

Таблица 4.16.5 – Биты конфигурации передачи BC GR1553B для различных типов передачи

Тип передачи	RTAD1 (15:11)	RTSA1 (9:5)	RTAD2 (25:21)	RTSA2 (20:16)	WCMC (4:0)	TR (10)	Направление буфера данных
Данные, из BC в RT	Адрес RT (0-30)	Подадрес RT(1-30)	Безразличное состояние	0	Счетчик слов (от 0 до 32)	0	Считывание (2-64 байта)
Данные, из RT в BC	Адрес RT (0-30)	Подадрес RT(1-30)	Безразличное состояние	0	Счетчик слов (от 0 до 32)	1	Запись (2-64 байта)
Данные, из RT в RT	Адрес Recv-RT (0-30)	Подадрес Recv-RT (1-30)	Адрес Xmit-RT (0-30)	Подадрес Xmit-RT (1-30)	Счетчик слов (от 0 до 32)	0	Запись (2-64 байта)
Режим, нет данных	Адрес RT (0-30)	0 или 31*	Безразличное состояние	Безразличное состояние	Код режима (0-8)	1	Не используется
Режим, из RT в BC	Адрес RT (0-30)	0 или 31*	Безразличное состояние	Безразличное состояние	Код режима (16/18/19)	1	Запись (2 байта)
Режим, из BC в RT	Адрес RT (0-30)	0 или 31*	Безразличное состояние	Безразличное состояние	Код режима (17/20/21)	0	Считывание (2 байта)
Данные широко-вещательные, из BC в RTs	31	Подадрес RTs (1-30)	Безразличное состояние	0	Счетчик слов (от 0 до 32)	0	Считывание (2-64 байта)
Данные широко-вещательные, из RT в RTs	31	Подадрес Recv-RTs (1-30)	Адрес Xmit-RT (0-30)	Подадрес Xmit-RT (1-30)	Счетчик слов (от 0 до 32)	0	Запись (2-64 байта)
Режим широко-вещательный, нет данных	31	0 или 31*	Безразличное состояние	Безразличное состояние	Код режима (1, 3-8)	1	Не используется
Режим широко-вещательный, из BC в RT	31	0 или 31*	Безразличное состояние	Безразличное состояние	Код режима (17/20/21)	0	Считывание (2 байта)

* Согласно стандарту для команд режима можно использовать подадрес 0 или 31.

Слово условного перехода организовано в соответствии с таблицей 4.16.6.

Таблица 4.16.6 – Слово условного перехода GR1553B (смещение 0x00)

31	30 – 27	26	25	24	23 – 16	15 – 8	7 – 0
1	–	IRQC	ACT	MODE	RT2CC	RTCC	STCC

31 Должно быть установлено для идентификации в качестве перехода
30–27 Зарезервировано (считывается как 0)
26 Прерывание при выполненном условии (IRQC)

25	Действие (ACT) – Действие при выполненном условии, «0» – перевод плана в режим ожидания, «1» – команда перехода
24	Логический режим (MODE): «0» – режим «ИЛИ» (любой бит, установленный в RT2CC, RTCC, – установлен в RT2ST, RTST или результат находится в маске STCC). «1» – режим «И» (все биты, установленные в RT2CC, RTCC, установлены в RT2ST, RTST и результат находится в маске STCC)
23–16	Код условия RT 2 (RT2CC) – Маска с битами, соответствующими RT2ST в итоговом слове последней передачи
15–8	Код условия RT (RTCC) – Маска с битами, соответствующими RTST в итоговом слове последней передачи
7–0	Код условия статуса (STCC) – Маска с битами, соответствующими значению статуса последней передачи

Следует отметить, что при установке MODE = 0 и STCC = 0xFF создается постоянное истинное условие, а при установке STCC = 0x00 создается постоянное ложное условие. Таким образом, 0x800000FF может использоваться как признак окончания списка.

4.16.5 Работа удаленного терминала

Общие сведения

При работе в качестве удаленного терминала устройство выступает ведомым устройством на шине MIL-STD-1553, выполняя следующие функции: прослушивает запросы на собственный адрес RT (или передачи транслирования), проверяет допустимость запросов и, в случае допустимости, выполняет соответствующую передачу. В случае недопустимости устанавливает флаг ошибки сообщения в слове статуса. Допустимость контролируется с помощью слова управления подадреса для передачи данных и с помощью регистра управления кода режима.

Для старта RT необходимо инициализировать таблицу подадреса и кольцевой буфер регистрации, а затем записать адрес и бит разрешения RT в регистре конфигурации RT.

Обработка передачи данных

Режим удаленного терминала использует трехуровневую структуру для управления передачей данных посредством прямого доступа к памяти (DMA), рисунок 4.16.3. Верхний уровень – строка таблицы подадресов, в которой каждый подадрес имеет слово управления подадреса, указатели на дескрипторы передачи и дескрипторы приема. Каждый дескриптор, в свою очередь, имеет слово управления/статуса дескриптора, указатель на буфер данных и указатель на следующий дескриптор, формируя связанный список или кольцевую схему структур данных. Буферы данных постоянно хранятся в любом месте памяти с выравниванием до 16 бит.

При получении RT запроса на передачу данных, RT проверяет допустимость запроса в таблице подадреса. В случае допустимости передача выполняется с отправлением или приемом данных посредством DMA в/из соответствующего буфера данных. После передачи данных слово управления/статуса указателя обновляется, приобретая статус успешной или неудачной передачи, и указатель таблицы подадреса изменяется на указатель к следующему дескриптору.

Если разрешена регистрация, регистрационная запись будет помещаться в область кольцевого буфера регистрации. Прерывание запущенной передачи также может быть разрешено. Для идентификации передачи, вызвавшей прерывание, значение регистра

позиции прерывания регистрации события RT («RT event log interrupt position») указывает на соответствующую запись регистрации. Поэтому для разрешения прерываний должна быть разрешена регистрация.

Если запрос допустим, но не может быть выполнен, потому что не готов соответствующий дескриптор или потому что данные не могут быть доступны в течение требуемого времени для ответа, устройство отправляет сигнал о прерывании из-за ошибки доступа к таблице RT и не отвечает на запрос. Дополнительно при данных условиях ошибки автоматически устанавливается бит статуса терминального флага.

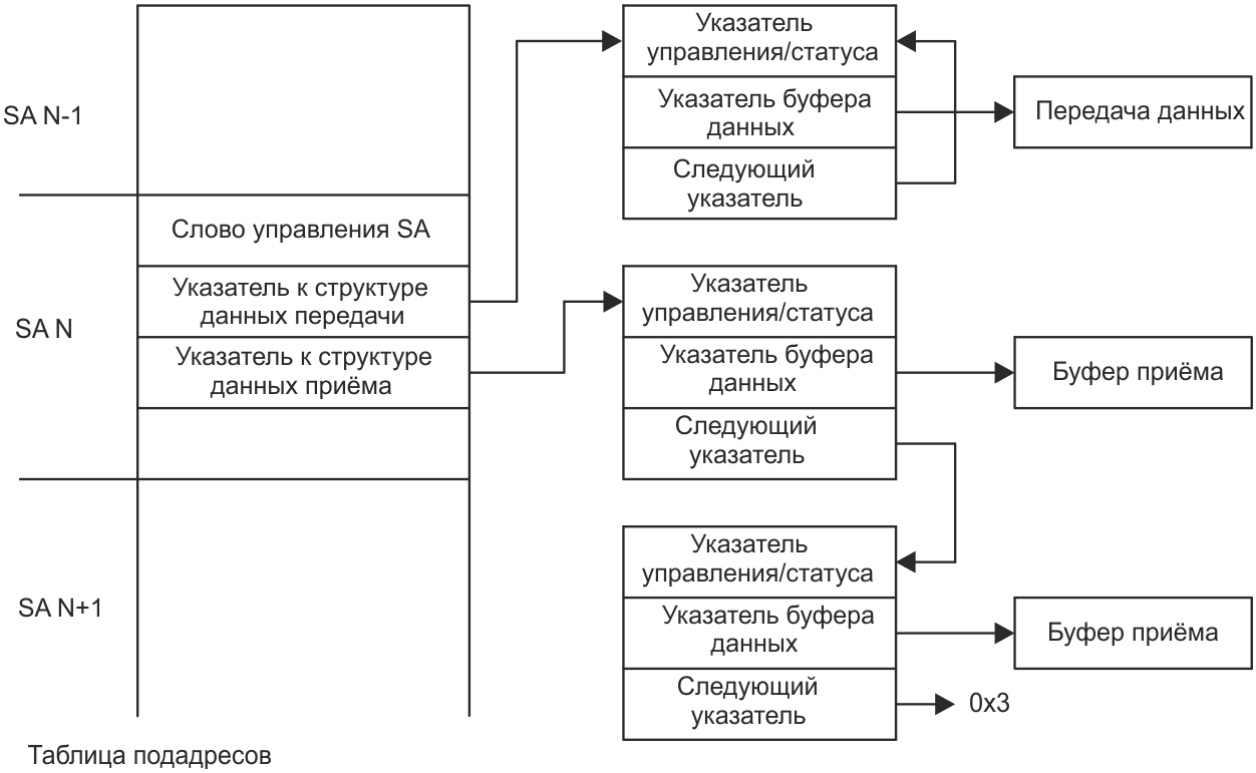


Рисунок 4.16.3 – Пример структуры данных подадреса RT

Коды режимов

Допустимые коды режимов MIL-STD-1553, подлежащих регистрации и генерации прерываний, задаются через регистр управления кодами режима RT. В отношении передачи данных для разрешения прерываний необходимо разрешить регистрацию. Недопустимые коды режимов задаются с помощью тех же полей, что и допустимые, а коды режимов, которые могут выполнять широковещательную трансляцию, имеют два отдельных поля для управления широковещательным и не широковещательным транслированием.

Разные коды режимов и соответствующие действия RT представлены в таблице 4.16.7. Некоторые коды режимов не имеют встроенного действия, поэтому, при необходимости, действие должно быть реализовано через программное обеспечение. В таблице ниже также представлено отношение каждого кода режима к полям регистра управления кодами режимов RT.

Таблица 4.16.7 – Коды режима RT

Код режима		Описание	Встроенное действие при разрешенном коде режима	Регистрация / Запрос на прерывание	Разрешено после сброса	Биты регистра управления
1	2	3	4	5	6	7
0	00000	Динамическое управление шины	Если в регистре статуса шины RT установлен бит DBCA, отправляется ответ принятия динамического управления шины	Есть	Нет	17, 16
1	00001	Синхронизация	Поле времени в регистре синхронизации RT обновлено. Выходной сигнал rtsync выдает логическую «1» длительностью в один цикл AMBA	Есть	Да	3–0
2	00010	Передача слова статуса	Передача статусного слова RT. Всегда разрешено, не может быть регистрировано или не разрешено	Нет	Да	–
3	00011	Запуск самопроверки	Нет встроенного действия	Есть	Нет	21–18
4	00100	Отключение передатчика	RT не будет отвечать на команды на другой шине (не на шине, на которой дана команда)	Есть	Да	11–8
5	00101	Отмена отключения передатчика	Удаление действия кода режима отключения передатчика, полученного ранее на той же шине	Есть	Да	11–8
6	00110	Запрещение терминального флага	Маскирование терминального флага отправленных слов статуса RT	Есть	Нет	25–22
7	00111	Отмена запрещения терминального флага	Удаление действия режима блокировки терминального флага	Есть	Нет	25–22

Окончание таблицы 4.16.7

1	2	3	4	5	6	7
8	01000	Сброс удаленного терминала	Сброс сбоеустойчивых таймеров, отключения передатчика, запрещения терминального флага и запрещения статуса. Биты терминального флага и запроса обслуживания в регистре статуса шины RT очищаются. Выходной сигнал extreset выдает логическую «1» длительностью в один цикл AMBA	Есть	Нет	29–26
16	10000	Передача векторного слова	Ответ с помощью векторного слова из регистра слов статуса RT	Есть	Нет	13, 12
17	10001	Синхронизация со словом данных	Обновление полей времени и данных в регистре синхронизации RT. Выходной сигнал rtsync выдает логическую «1» длительностью в один цикл AMBA	Есть	Да	7–4
18	10010	Передача последней команды	Передача последней команды, отправленной в RT. Всегда разрешено, не может быть регистрировано или не разрешено	Нет	Да	–
19	10011	Передача слова BIT	Ответ словом BIT из регистра слов статуса RT	Есть	Нет	15, 14
20	10100	Отключение выбранного передатчика	Нет встроенного действия	Нет	Нет	–
21	10101	Отмена отключения выбранного передатчика	Нет встроенного действия	Нет	Нет	–

Журнал регистрации событий

Регистрация событий выполняется посредством кольцевой схемы 32-битовых данных в формате, представленном в таблице 4.16.8. Следует обратить внимание на то, что для передачи данных биты (23–0) в журнале регистрации событий аналогичны битам (23–0) в слове статуса указателя.

Таблица 4.16.8 – Формат данных журнала регистрации событий RT GR1553B

31	30 29	28 24	23 10	9	8 3	2 0
IRQSRC	TYPE	SAMC	TIMEL	BC	SZ	TRES

31	Источник запроса на прерывание (IRQSRC) – Установлено, если прерывание вызвала данная передача
30, 29	Тип передачи (TYPE): «00» – передача данных, «01» – прием данных, «10» – код режима
28–24	Подадрес/код режима (SAMC): Если бит TYPE установлен в 0x0 или в 0x1 – это соответствует подадресу передачи, а если в 0x2 – коду режима
23–10	TIMEL – Младшие 14 битов счетчика меток времени
9	Широковещательное транслирование (BC) – Бит установлен, если запрос предназначен для широковещательного транслирования

8–3 Размер передачи (SZ) – Количество 16-битовых слов (0 – 32)

2–0 Результат передачи (TRES):
«000»: успешная передача;
«001»: отмененная передача (т.к. была дана новая команда на другой шине);
«010»: ошибка DMA или тайм-аут памяти;
«011»: ошибка протокола (ошибочно синхронизированные слова данных или ошибка декодера);
«100»: бит занятости или бит ошибки сообщения установлен в переданном слове статуса, данные не отправлены;
«101»: передача прервана из-за ошибки проверки «обратной петли»

Формат таблицы подадреса

Данные таблицы подадреса GR1553B RT, слово управления таблицы подадреса и формат дескриптора RT GR1553B представлены в таблицах 4.16.9 – 4.16.12.

Таблица 4.16.9 – Данные таблицы подадреса GR1553B RT для подадреса N, 0 < N < 31

Смещение	Значение	DMA R/W
0x10×N + 0x00	Слово управления подадреса N (см. таблицу 4.16.10)	R
0x10×N + 0x04	Указатель на дескриптор передачи, выровнено до16 байт (0x3 обозначает недействительный указатель)	R/W
0x10×N + 0x08	Указатель на дескриптор приема, выровнено до 16 байт (0x3 обозначает недействительный указатель)	R/W
0x10×N + 0x0C	Не используется	–
Примечание – Доступ устройства к данным строк таблицы для подадресов кода режима 0 и 31 всегда закрыт.		

Таблица 4.16.10 – Слово управления таблицы подадреса RT GR1553B (смещение 0x00)

31 19	18	17	16	15	14	13	12 8	7 6	5	4 0
–	WRAP	IGNDV	BCRXE	RXEN	RXLOG	RXIRQ	RXSZ	TXEN	TXLOG	TXIRQ	TXSZ

31–19 Зарезервировано

18 Разрешение автоматического циклического перехода (WRAP) – Разрешает режим проверки для данного подадреса, в котором передатчик отправляет последние принятые данные обратно. Это осуществляется копированием указателя на дескриптор завершенной передачи в адрес указателя на дескриптор передачи после каждой успешной передачи.
Примечание – Если WRAP = 1, не допускается устанавливать TXSZ > RXSZ, поскольку это может вызвать считывание за пределами буфера

17 Игнорирование бита действительности данных (IGNDV) – Если бит установлен, прием передачи продолжается (с перезаписью буфера), если при приеме указателя установлен бит действительности данных, вместо отсутствия ответа на запрос.
Это может использоваться для кольцевых схем указателей при безразличном состоянии в отношении перезаписи первоначальных данных

16 Разрешение приема широковещательных сообщений (BCRXE) – Обеспечивает прием передач широковещательных сообщений в данном подадресе

15 Разрешение приема (RXEN) – Обеспечивает прием передач в данном подадресе

14	Регистрация приема передач (RXLOG) – Регистрация всех передач в кольцевой схеме журнала событий (используется в случае, если RXEN=1)
13	Прерывание при приеме (RXIRQ) – Каждый прием данных вызывает прерывание (в случае, если RXEN, RXLOG=1)
12–8	Максимально допустимый размер приема (RXSZ) в данный подадрес в 16-битовых словах (от 0 до 32)
7	Разрешение передачи (TXEN) – Обеспечивает передачи из данного подадреса
6	Регистрация отправки передач (TXLOG) – Регистрация всех передач в кольцевой схеме журнала регистрации событий (в случае, если TXEN=1)
5	Прерывание при передаче (TXIRQ) – Каждая передача данных вызывает прерывание (в случае, если TXEN, TXLOG=1)
4–0	Максимально допустимый размер передачи (TXSZ) из данного подадреса – В 16-битовых словах (от 0 до 32)

Таблица 4.16.11 – Формат дескриптора приема/передачи RT GR1553B

Смещение	Значение	DMA R/W
0x00	Слово управления/статуса, см. таблицу 4.16.12	R/W
0x04	Указатель буфера данных, с выравниванием до 16 бит	R
0x08	Ссылка на следующий дескриптор, с выравниванием до 16 бит или 0x0000003 для обозначения окончания списка	R

Таблица 4.16.12 – Слово управления/статуса дескриптора RT GR1553B (смещение 0x00)

31	30	29 26	25 10	9	8 3	2 0
DV	IRQEN	–	TIME	BC	SZ	TRES

31	Действительные данные (DV) – Бит должен быть сброшен программным обеспечением до передачи и устанавливается аппаратно после передачи. Если DV = 1 в текущем дескрипторе приема до начала приема передачи, активируется ошибка таблицы дескрипторов. Это действие можно отменить установкой бита IGNDV в таблице подадреса
30	Доминирующее разрешение запроса на прерывание (IRQEN) – Регистрирование и запрос на прерывание после передачи, независимо от установок слова управления подадреса SA. Используется для получения прерывания при приближении к концу списка дескрипторов
29–26	Зарезервировано – Записывается «0» и считывается замаскированным для обеспечения согласованности
25–10	Временная метка передачи (TIME) – В поле помещается значение таймера RT после завершения передачи
9	Широковещательный (BC) –Устанавливается устройством в случае, если передача была широковещательной
8–3	Размер передачи (SZ) – Количество 16-битовых слов (0–32)
2–0	Результат передачи (TRES): 000: успешная передача; 001: отмененная передача (т.к. была дана новая команда на другой шине); 010: ошибка DMA или тайм-аут памяти; 011: ошибка протокола (ошибочно синхронизированные слова данных или ошибка декодера); 100: бит занятости или бит ошибки сообщения был установлен в переданном слове статуса, данные не отправлены; 101: передача прервана из-за ошибки проверки возврата к началу цикла

4.16.6 Работа монитора шины

Общие сведения

Монитор шины (BM) разрешается отдельно или одновременно с BC или RT. BM работает как пассивное устройство регистрации, записывающее полученные данные с дополнительной временной информацией в кольцевой буфер.

Фильтрация

Монитор шины также поддерживает фильтрацию. Это дополнительная опция; наличие данной опции проверяется через программное обеспечение, которое выявляет регистры фильтра BM с возможностью записи.

Фильтрация передач выполняется по адресу RT и подадресу, или коду режима. Условия фильтрации проверяются посредством схемы логического «И». Если все биты трех регистров фильтрации и биты 2, 3 регистра управления установлены, BM записывает все слова, полученные на шине.

Для фильтрации по подадресу/коду режима логика BM отслеживает слова 1553, принадлежащие одному сообщению. Поддерживаются все 10 типов сообщений. При выявлении непредусмотренного слова логика фильтра перезапускается. Слова данных, не принадлежащие никакому сообщению, могут быть регистрированы установкой бита в регистре управления.

Логика фильтра может перезапускаться вручную путем установки бита разрешения BM сначала в низкое логическое состояние, а затем опять в высокое логическое состояние. Эта опция позволяет улучшить самотестируемость BM.

Возможности фильтрации можно сконфигурировать извне BM в безопасной области. В этом случае все видимые слова регистрируются, и регистры управления фильтрации становятся предназначенными только для чтения и всегда считываются, как установленные. Однако, можно контролировать регистрацию ошибок манчестерского кода/четности.

Отсутствие обработки ответа

В протоколе MIL-STD-1553 слово команды для кода режима, использующего индикатор «0» или стандартную передачу в подадрес 8, имеет ту же самую структуру, что и допустимое слово статуса. Поэтому при использовании фильтров подадреса или кода режимов может возникнуть неоднозначность, в этом случае RT не отвечает на подадрес. В таком случае BC снова отправляет команду на тот же RT на подадрес 8 или код режима с использованием индикатора 0 на той же шине. В результате слово второй команды интерпретируется как слово статуса и отфильтровывается.

Монитор шины может использовать инструментальный бит или резервные биты для устранения неопределенности, что означает, что этот случай никогда не произойдет, когда используются подадресы 1–7, 9–30 и индикатор кода режима 31. Также эта ситуация никогда не произойдет в случае, если подадрес/код режима не используются, а используется только адрес RT.

Формат записей регистрирования

Каждая запись регистрирования состоит из двух 32-битовых слов, приведенных в таблицах 4.16.13, 4.16.14.

Таблица 4.16.13 – Слово «0» записи регистрирования данных ВМ GR1553B (смещение 0x00)

31	30 – 24	23 – 0
1	–	TIME

31	Всегда установлено
30–24	Зарезервировано – Маскируется при чтении для обеспечения согласованности
23–0	Временная метка передачи (TIME)

Таблица 4.16.14 – Слово «1» записи регистрирования данных ВМ GR1553B (смещение 0x04)

31	30 – 20	19	18 – 17	16	15 – 0
0	–	BUS	WST	WTR	WD

31	Всегда установлено
30–20	Зарезервировано – Маскируется при чтении для обеспечения согласованности
19	Шина приема данных (BUS): «0» – A, «1» – B
18, 17	Статус слова (WST): «00» – слово OK, «01» – ошибка манчестерского кода, «10» – ошибка четности.
16	Тип слова (WTP): «0» – данные, «1» – команда/статус
15–0	Данные (WD)

4.16.7 Тактирование

Устройство работает в двух тактовых доменах: тактовом домене шины AMBA и тактовом домене кодека 1553, с синхронизацией и взаимодействием между доменами.

Тактирование шины AMBA должно осуществляться с частотой не менее 10 МГц. Задержка передачи до одного цикла тактирования с одним кодеком (50 мкс) допустима в каждом сигнале, проходящем через тактовую область.

Тактирование домена кодека 1553 осуществляется через отдельный вход (CLK_1553B) тактовым сигналом с фиксированной частотой 20 МГц (см. рисунок 4.16.4).

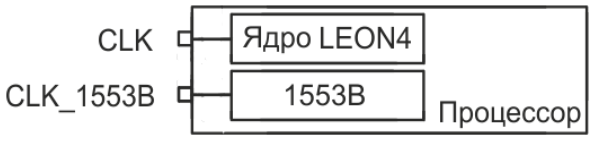


Рисунок 4.16.4 – Схема тактирования ядра микропроцессора и интерфейсов Mil-STD-1553B

Назначение сигналов:
CLK – тактовый вход процессора;
CLK_1553B – тактирование интерфейсов Mil-STD-1553B.

4.16.8 Регистры

Программирование устройства осуществляется через регистры, отображаемые в адресуемом пространстве APB. Если RT, BC или BM ядра исключены из конфигурации, соответствующие регистры станут неактивными и будут возвращать нулевое значение

при считывании, см. таблицу 4.16.15. Зарезервированные поля регистра должны быть сброшены и замаскированы после считывания.

Перечень регистров интерфейса Mil-STD-1553

Таблица 4.16.15 – Регистры интерфейса MIL-STD-1553 (базовые адреса 0x80100200 и 0x80100300)

Смещение адреса APB	Регистр	R/W	Значение сброса
0x00	Регистр прерываний	R/W (запись «1» сбрасывает)	0x00000000
0x04	Регистр разрешений запросов на прерывания	R/W	0x00000000
0x08...0x0F	Зарезервировано	–	–
0x10	Регистр аппаратной конфигурации	R (константа)	0x00000000
0x14...0x3F	Зарезервировано	–	–
0x40...0x7F	Область регистров BC (см. таблицу 4.16.16)	–	–
0x80...0xBF	Область регистров RT (см. таблицу 4.16.17)	–	–
0xC0...0xFF	Область регистров BM (см. таблицу 4.16.18)	–	–

Набор регистров контроллера шины

Таблица 4.16.16 – Специальные регистры интерфейса MIL-STD-1553 для BC

Смещение адреса APB	Регистр	R/W	Значение сброса
0x40	Регистр конфигурации и статуса BC	R/W	0xd0000000
0x44	Регистр действий BC	W	–
0x48	Указатель на следующий список передач BC	R/W	0x00000000
0x4C	Указатель на следующий асинхронный список BC	R/W	0x00000000
0x50	Регистр таймера BC	R	0x00000000
0x54	Регистр выхода таймера из режима ожидания BC	R/W	0x00000000
0x58	Кольцевая позиция запроса на прерывание, инициируемого передачей BC	R/W	0x00000000
0x5C	Регистр перестановки шины BC Per-RT	R/W	0x00000000
0x60...0x67	Зарезервировано	–	–
0x68	Указатель текущего положения в списке передач BC	R	0x00000000
0x6C	Указатель текущего положения в асинхронном списке BC	R	0x00000000
0x70...0x7F	Зарезервировано	–	–

Набор регистров удаленного терминала

Таблица 4.16.17 – Специальные регистры интерфейса MIL-STD-1553 для RT

Смещение адреса APB	Регистр	R/W	Значение сброса
1	2	3	4
0x80	Регистр статуса RT	R	0x80000000
0x84	Регистр конфигурации RT	R/W	0x0000e03e*
0x88	Регистр битов статуса шины RT	R/W	0x00000000
0x8C	Регистр слов статуса RT	R/W	0x00000000
0x90	Регистр синхронизации RT	R	0x00000000
0x94	Базовый адрес таблицы подадреса RT	R/W	0x00000000
0x98	Регистр управления кода режима RT	R/W	0x00000555
0x9C...0xA3	Зарезервировано	–	–

Окончание таблицы 4.16.17

1	2	3	4
0xA4	Регистр управления временной метки RT	R/W	0x00000000
0xA8	Зарезервировано	–	–
0xAC	Маска журнала регистрации событий RT	R/W	0xfffffffc
0xB0	Позиция в журнале регистрации событий RT	R/W	0x00000000
0xB4	Позиция прерывания в журнале регистрации событий RT	R	0x00000000
0xB8…0xBF	Зарезервировано	–	–

_____

* Значение сброса может меняться внешними входными сигналами RTADDR/RTPAR.

Набор регистров монитора шины

Таблица 4.16.18 – Специальные регистры интерфейса MIL-STD-1553 для BM

Смещение адреса APB	Регистр	R/W	Значение сброса
0xC0	Регистр статуса BM	R	0x80000000
0xC4	Регистр управления BM	R/W	0x00000000
0xC8	Регистр фильтрации адреса RT BM	R/W	0xffffffff
0xCC	Регистр фильтрации подадреса RT BM	R/W	0xffffffff
0xD0	Регистр фильтрации кода режима RT BM	R/W	0xffffffff
0xD4	Начало буфера записей регистрации BM	R/W	0x00000000
0xD8	Конец буфера записей регистрации BM	R/W	0x00000007
0xDC	Позиция в буфере записей регистрации BM	R/W	0x00000000
0xE0	Регистр управления метки времени BM	R/W	0x00000000
0xE4...0xFF	Зарезервировано	–	–

В таблицах 4.16.19 – 4.16.50 представлены регистры, отображаемые в адресуемом пространстве APB.

Регистр прерываний

Таблица 4.16.19 – Регистр запросов на прерывание GR1553B

31 18	17	16	15 11	10	9	8	7 3	2	1	0
–	BMTOF	BMD	–	RTTE	RTD	RTEV	–	BCWK	BCD	BCEV
0x0000	0	0	0x00	0	0	0	0x00	0	0	0
r	wc	wc	r	wc	wc	wc	r	wc	wc	wc

Биты считываются как установленные, если произошло прерывание; для подтверждения приема прерывания нужно записать «1».

31–18 Зарезервировано
17 Переполнение таймера BM (BMTOF)
16 Ошибка DMA BM (BMD)
15–11 Зарезервировано
10 Ошибка доступа к таблице RT (RTTE)
9 Ошибка DMA RT (RTD)
8 Прерывание, вызванное передачей RT (RTEV)
7–3 Зарезервировано
2 Прерывание выхода таймера из режима ожидания, BC (BCWK)
1 Ошибка DMA BC (BCD)
0 Прерывание, вызванное передачей BC (BCEV)

Регистр разрешений запросов на прерывания

Таблица 4.16.20 – Регистр разрешений запросов на прерывания GR1553B

31 18	17	16	15 11	10	9	8	7 3	2	1	0
–	BMTOE	BMDE	–	RTTEE	RTDE	RTEVE	–	BCWKE	BCDE	BCEVE
0x0000	0	0	0x00	0	0	0	0x00	0	0	0
r	rw	rw	r	rw	rw	rw	r	rw	rw	rw

31–18 Зарезервировано
17 Разрешение прерывания при переполнении таймера BM (BMTOE)
16 Разрешение прерывания при возникновении ошибки DMA BM (BMDE)
15–11 Зарезервировано
10 Разрешение прерывания при возникновении ошибки доступа к таблице RT (RTTEE)
9 Разрешение прерывания при возникновении ошибки DMA RT (RTDE)
8 Разрешение прерывания при возникновении события, вызванного передачей RT (RTEVE)
7–3 Зарезервировано
2 Разрешение прерывания выхода таймера из режима ожидания, BC (BCWKE)
1 Разрешение ошибки DMA BC (BCDE)
0 Разрешение прерывания, вызванного передачей BC (BCEVE)

Регистр аппаратной конфигурации

Таблица 4.16.21 – Регистр аппаратной конфигурации GR1553B

31	30 12	11	10 9	8	7 0
MOD	–	XKEYS	ENDIAN	SCLK	CCFREQ
0	0x00000	0	0x0	0	0x00
r	r	r	r	r	r

31 Модифицировано (MOD) – Зарезервировано для указания того, что устройство модифицировано/настроено произвольно
30–12 Зарезервировано
11 Установлено, если разрешены ключи безопасности для регистра управления BM и для всех полей регистра управления RT
10, 9 Порядок байт AHB – Установлено в 0x0 – первым следует старший байт
8 Один тактовый сигнал (SCLK) – Зарезервировано для последующих версий для указания того, что ядро модифицировано для работы с одиночным тактовым сигналом
7–0 Частота тактирования кодека (CCFREQ) – Зарезервировано для последующих версий для указания того, что ядро работает с разной частотой тактирования кодека. Значение частоты в МГц, значение 0x0 означает 20 МГц

Описание регистров контроллера шины

Таблица 4.16.22 – Регистр конфигурации и статуса BC GR1553B (BCCS)

31	30 28	27 17	16	15 11	10	9 8	7 3	2 0
BCSUP	BCFEAT	–	BCCHK	ASADL	0	ASST	SCADL	SCST
1	0x5	0x000	0	0x00	0	0x0	0x0	0x0
r	r	r	rw	r	r	r	r	r

31	Поддержка BC (BCSUP) – Бит установлен, поскольку устройство поддерживает режим BC
30–28	Особенности BC (BCFEAT) – Битовое поле, в котором дается описание дополнительных поддерживаемых возможностей («1» – поддерживается): 30 – поддержка таймера планировщика BC; 29 – поддержка прерываний при выходе из режима ожидания планировщика BC (сброшен); 28 – поддержка регистра перестановки шины BC per-RT и бита указателя STBUS
27–17	Зарезервировано
16	Проверка широковещательных сообщений (BCCHK) – Бит для записи. Если установлен, разрешено ожидание и проверка (неожидаемых) ответов на все широковещательные сообщения
15–11	Младшие адресные биты асинхронного списка (ASADL) – Биты 8–4 адреса указателя текущей (если ASST=01) или следующей асинхронной команды
10	Зарезервировано
9, 8	Состояние асинхронного списка (ASST) – Возможные значения поля: «00» – остановлено, «01» – выполнение команды, «10» – ожидание интервала ответа
7–3	Младшие адресные биты плана (SCADL) – Биты 8 – 4 адреса дескриптора текущего (если SCST = «001») или следующего плана
2–0	Состояние плана (SCST) – Биты сброшены, поскольку план находится в остановленном состоянии

Таблица 4.16.23 – Регистр действий BC GR1553B (BCACT)

31 16	15 10	9	8	7 5	4	3	2	1	0
BCKEY	–	ASSTP	ASSRT	–	CLRT	SETT	SCSTP	SCSUS	SCSRT
–	–	–	–	–	–	–	–	–	–
w	–	w	w	–	w	w	w	w	w

31–16	Защитный код (BCKEY) – Необходимо записать значение 0x1552, в противном случае запись в регистр будет игнорироваться
15–10	Зарезервировано
9	Остановка асинхронного плана (ASSTP) – Запись «1» останавливает асинхронный план (после текущей передачи, если выполняется)
8	Старт асинхронного плана (ASSRT) – Запись «1» запускает асинхронный план
7–5	Зарезервировано
4	Сброс внешнего активатора (CLRT) – Запись «1» сбрасывает активатор памяти
3	Установка внешнего активатора (SETT) – Запись «1» устанавливает активатор памяти
2	Остановка плана (SCSTP) – Запись «1» останавливает план (после текущей передачи, если выполняется)
1	Перевод плана в режим ожидания (SCSUS) – Запись «1» переводит план в режим ожидания (после текущей передачи, если выполняется)
0	Старт плана (SCSRT) – Запись «1» запускает план

Таблица 4.16.24 – Регистр следующего указателя передач BC GR1553B (BCNP)

31	0
SCHEDULE TRANSFER LIST POINTER	
0x00000000	
rw	

31–0 Чтение: Текущая (если SCST=001) или следующая передача для выполнения в стандартном плане.
Запись: Изменение адреса. В рабочем режиме после окончания текущей передачи выполняется переход.

Таблица 4.16.25 – Регистр следующего указателя асинхронного списка BC GR1553B (BCNPA)

31	0
SCHEDULE TRANSFER LIST POINTER	
0x00000000	
rw	

31–0 Чтение: Текущая (если ASST = 01) или следующая передача для выполнения в асинхронном плане.
Запись: Изменение адреса. В рабочем режиме после окончания текущей передачи выполняется переход.

Таблица 4.16.26 – Регистр таймера BC GR1553B (BCTMR)

31 24	23 0
–	SCHEDULE TIME (SCTM)
0x00	0x000000
r	r

31–24 Зарезервировано
23:0 Истекшее время «плана передач» в микросекундах (только для считывания). Сброшено при остановке плана или при внешней синхронизации.

Приведенный выше регистр реализует дополнительный функционал, см. регистр конфигурации и статуса BC (таблица 4.16.22) поле BCFEAT бит 30.

Таблица 4.16.27 – Регистр указателя позиции кольцевого буфера указателей запросов на прерывание, запущенных передачей BC GR1553B (BCIP)

31	0
Указатель позиции кольцевого буфера указателей запросов на прерывание BC	
0x00000000	
rw	

31–0 Указатель текущей позиции в кольцевом буфере указателей запросов на прерывание, инициированных передачей. Биты 1, 0 всегда сброшены (выравнивание до 4 байт).
Циклический переход выполняется на границе 64 байт, поэтому биты 31–6 меняются только пользователем

Таблица 4.16.28 – Регистр перестановки шины BC GR1553B per-RT (BCSW)

31	0
BC PER-RT регистр перестановки шины	
0x00000000	
rw	

31–0 Операция «исключающее ИЛИ» будет выполняться между значением выбора шины и битами данной маски, соответствующей адресуемому RT (принимающий RT для передач из RT в RT). Этот регистр обновляется устройством при использовании бита указателя STBUS.

Приведенный выше регистр реализует дополнительный функционал, см. регистр конфигурации и статуса BC поле BCFEAT бит 28. Более подробная информация по использованию данной опции приведена в подпункте «Выбор шины» пункта «4.16.4».

Таблица 4.16.29 – Указатель текущей позиции в списке передач BC GR1553B (BCPP)

31	0
Указатель текущей передачи BC	
0x00000000	
r	

31–0 Указатель на дескриптор передачи, соответствующей текущему временному интервалу (только для считывания, допустимо только в рабочем режиме списка передач).
Биты 3–0 всегда сброшены (выравнивание до 128 бит /16 байт)

Таблица 4.16.30 – Указатель текущей позиции в асинхронном списке BC GR1553B (BCPPA)

31	0
BC Указатель текущей асинхронной передачи	
0x00000000	
r	

31–0 Указатель на дескриптор передачи, соответствующий текущему временному интервалу асинхронного плана (только для считывания, допустимо только в рабочем режиме асинхронного списка).
Биты 3:0 всегда сброшены (выравнивание до 128 бит/16 байт)

Описание регистров удаленного терминала

Таблица 4.16.31 – Регистр статуса RT GR1553B, только для считывания (RTST)

31	30 – 4	3	2	1	0
RTSUP	–	ACT	SHDA	SHDB	RUN
1	0x0000000	0	0	0	0
r	r	r	r	r	r

31 Поддержка RT (RTSUP) – Бит считывается как установленный, если устройство поддерживает режим RT
30–4 Зарезервировано
3 Активный RT (ACT) – Бит установлен, если RT в данный момент обрабатывает передачу

2 Отключение шины A (SHDA) – Бит считывается как установленный, если шина A отключена BC (с помощью команды отключения передатчика на шине B)

1 Отключение шины B (SHDB) – Бит считывается как установленный, если шина B отключена BC (с помощью команды отключения передатчика на шине A)

0 RT в рабочем режиме (RUN) – Бит считывается как установленный, если RT принимает команды

Таблица 4.16.32 – Регистр конфигурации RT GR1553B (RTCFG)

31 16	15	14	13	12 7	6	5 1	0
RTKEY	SYS	SYDS	BRS	–	RTEIS	RTADDR	RTEN
0x0000	1	1	1	0x00	0	0x1F	0
w	rw	rw	rw	r	r	rw	rw

31–16 Защитный код (RTKEY) – Необходимо записать значение 0x1553 для того, чтобы изменить адрес RT, в противном случае поле адреса не меняется при записи. При считывании данного регистра это поле считывается, как 0x0000. Если разрешены дополнительные защитные ключи (см. регистр конфигурации аппаратного оборудования), младшая половина ключа используется для защиты других полей регистра

15 Разрешение сигнала синхронизации (SYS) – Устанавливается для вывода импульса на выход rtsync после приема кода режима синхронизации (без данных)

14 Разрешение сигнала синхронизации с данными (SYDS) – Устанавливается для вывода импульса на выход rtsync после получения кода режима синхронизации со словом данных

13 Разрешение сигнала сброса шины (BRS) – Устанавливается для выдачи импульса на выход busreset после получения кода режима сброса удаленного терминала

12–7 Зарезервировано

6 Считывается как «1», если текущий адрес установлен через внешние входы. После установки адреса из программного обеспечения данное поле сбрасывается

5–1 Адрес RT (RTADDR)

0 Разрешение RT (RTEN) – Устанавливается для разрешения прослушивания запросов

Таблица 4.16.33 – Регистр статуса шины RT GR1553B (RTBST)

31 9	8	7 5	4	3	2	1	0
–	TFDE	–	SREQ	BUSY	SSF	DBCA	TFLG
0x000000	0	0x0	0	0	0	0	0
r	rw	rw	rw	rw	rw	rw	rw

31–9 Зарезервировано

8 Автоматически устанавливает терминальный флаг в случае обнаружения ошибок DMA и ошибок таблицы дескрипторов (TFDE)

7–5 Зарезервировано

4–0 Данные биты отправляются в ответ статуса RT по шине 1553:

4 - запрос на обслуживание (SREQ);

3 - бит занятости (BUSY).

Примечание – Если установлен бит занятости, ответ RT содержит только слово статуса, при этом передача не выполняется.

2 - флаг подсистемы (SSF);

1 - получение динамического управления шины (DBCA);

Примечание – Этот бит отправляется только в ответ на код режима динамического управления шины.

0 - терминальный флаг (TFLG).

ВС маскирует данный флаг при помощи команды «блокировка терминального флага», если разрешено

Таблица 4.16.34 – Регистр слов статуса RT GR1553B (RTW)

31 16 15 0

BIT WORD (BITW)	VECTOR WORD (VECW)
0x0000	0x0000
rw	rw

31–16 Слово BIT – Передается в ответ на команду «передача слова BIT», если разрешено

15–0 Векторное слово – Передается в ответ на команду «передача векторного слова», если разрешено

Таблица 4.16.35 – Регистр синхронизации RT GR1553B (RTS)

31 16 15 0

SYNC TIME (SYTM)	SYNC DATA (SYD)
0x0000	0x0000
r	r

31–16 Значение таймера RT при последней синхронизации или при команде синхронизации со словом данных, если разрешено

15–0 Данные, полученные при последней синхронизации со словом данных, если разрешено

Таблица 4.16.36 – Регистр базового (основного) адреса таблицы подадресов RT GR1553B (RTBA)

31 9 8 0

SUBADDRESS TABLE BASE (SATB)	0
0x000000	0x00
rw	r

31–9 Базовый адрес, биты 31–9 для таблицы подадреса

8–0 Всегда считываются как сброшенные, при записи – без изменений

Таблица 4.16.37 – Регистр управления кода режима RT GR1553B (RTCM)

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16

–	RRTB	RRT	ITFB	ITF	ISTB	IST	DBC
0	0x0	0x0	0x0	0x0	0x0	0x0	0x0
r	rw	rw	rw	rw	rw	rw	rw

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

TBW	TVW	TSB	TS	SDB	SD	SB	S
0x0	0x0	0x1	0x1	0x1	0x1	0x1	0x1
rw	rw	rw	rw	rw	rw	rw	rw

Для каждого кода режима: «00» – недопустимо, «01» – допустимо, «10» – допустимо, разрешена регистрация, «11» – допустимо, разрешены регистрация и прерывание.

31, 30	Зарезервировано
29, 28	Сброс широковещательного удаленного терминала (RRTB)
27, 26	Сброс удаленного терминала (RRT)
25, 24	Запрещение и отмена запрещения широковещательного бита терминального флага (ITFB)
23, 22	Запрещение и отмена запрещения терминального флага (ITF)
21, 20	Запуск широковещательного самотестирования (ISTB)
19, 18	Запуск самотестирования (IST)
17, 16	Динамическое управление шины (DBC)
15, 14	Передача слова BIT (TBW)
13, 12	Передача векторного слова (TVW)
11, 10	Широковещательное отключение, отмена отключения передатчика (TSB)
9, 8	Отключение, отмена отключения передатчика (TS)
7, 6	Широковещательная синхронизация со словом данных (SDB)
5, 4	Синхронизация со словом данных (SD)
3, 2	Широковещательная синхронизация (SB)
1, 0	Синхронизация (S)

Таблица 4.16.38 – Регистр управления метки времени RT GR1553B (RTTT)

31 16	15 0
TIME RESOLUTION (TRES)	TIME TAG VALUE (TVAL)
0x0	0x0
rw	rw

31–16	Разрешение метки времени (TRES) – Единица времени счетчика меток времени RT в микросекундах, минус 1
15–0	Значение метки времени (TVAL) – Текущее значение счетчика меток времени в рабочем режиме

Таблица 4.16.39 – Регистр маски журнала регистрации событий RT GR1553B (RTM)

31 21	20 2	1 0
–	EVENT LOG SIZE MASK	–
0x7FF	0x7FFFF	0x0
r	rw	r

31–21	Биты должны быть установлены
20–2	Маска, определяющая размер и выравнивание кольцевого буфера регистрации событий RT
1–0	Биты должны быть сброшены

Таблица 4.16.40 – Регистр позиции в журнале регистрации событий RT GR1553B (RTP)

31 0
EVENT LOG WRITE POINTER
0x00000000
rw

31–0	Адрес первой неиспользуемой/наиболее ранней записи буфера регистрации событий, с выравниванием до 32 бит

Таблица 4.16.41 – Регистр позиции прерывания в журнале регистрации событий RT GR1553B (RTIP)

31	0
EVENT LOG IRQ POINTER	
0x00000000	
r	

31–0 Адрес записи регистрации события, соответствующего прерыванию, с выравниванием до 32 бит

Описание регистров монитора шины

Таблица 4.16.42 – Регистр статуса BM GR1553B (BMST)

31	30	29 0
BMSUP	KEYEN	–
1	*	0
r	r	r

31 Поддержка BM (BMSUP) – Бит считывается как установленный поскольку устройство поддерживает BM

30 Разрешение ключа (KEYEN) – Бит считывается как установленный, если BM подтверждает правильность поля BMKEY при записи в регистр управления

29–0 Зарезервировано

Таблица 4.16.43 – Регистр управления BM GR1553B (BMCR)

31 16	15 6	5	4	3	2	1	0
BMKEY	–	WRSTP	EXST	IMCL	UDWL	MANL	BMEN
0x0000	0x000	0	0	0	0	0	0
rw	r	rw	rw	rw	rw	rw	rw

31–16 Ключ безопасности – Если разрешено использование ключа безопасности (KEYEN), данное поле должно быть установлено в 0x1543 для успешной записи информации в регистр управления

15–6 Зарезервировано

5 Остановка при циклическом переходе (WRSTP) – Если данный бит установлен, бит BMEN будет сброшен и BM остановлен, когда будет осуществлен автоматический перевод позиции буфера регистрации BM с последней записи на начальную

4 Старт внешней синхронизации (EXST) – Если в бит записывается логическая единица, будет установлен бит BMEN и BM начнет работать, когда придет внешний импульс синхронизации BC

3 Регистрация недопустимого кода режима (IMCL) – Устанавливается для регистрации недопустимых или зарезервированных кодов режима

2 Регистрация неожидаемых слов данных (UDWL) – Устанавливается для регистрации слов данных, которые не являются частью какой-либо команды

1 Регистрация ошибки манчестерского кода/четности (MANL) – Устанавливается для регистрации ошибок декодирования битов

0 Разрешение BM (BMEN) – Должно быть установлено для разрешения регистрации BM

Таблица 4.16.44 – Регистр фильтрации адреса RT BM GR1553B (BMMA)

31	30 … 0
	Регистр маски фильтра адреса
1	0x7FFFFFF
rw	rw

31 Разрешение ведения регистрации широковещательных передач
30–0 Каждая битовая позиция, установленная в «1», разрешает ведение регистрации передач с соответствующим адресом RT

Таблица 4.16.45 – Регистр фильтрации подадреса RT BM GR1553B (BMMSA)

31	30 … 1	0
	Регистр маски фильтра подадреса	
1	0x3FFFFFF	1
rw	rw	rw

31 Разрешение ведения регистрации команд режима на подадресе 31
30–1 Каждая битовая позиция, установленная в «1», разрешает запись передач с соответствующим подадресом RT
0 Разрешение ведения регистрации команд режима на подадресе 0

Таблица 4.16.46 – Регистр фильтрации кода режима RT BM GR1553B (BMFCM)

31 … 19	18	17	16
–	STSB	STS	TLC
0x1FFF	1	1	1
r	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSW	RRTB	RRT	ITFB	ITF	ISTB	IST	DBC	TBW	TVW	TSB	TS	SDB	SD	SB	S
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Каждый установленный бит разрешает регистрацию кода режима:

31–19 Зарезервировано
18 Отключение широковещательного транслирования передатчика и отмена отключения широковещательного транслирования передатчика (STSB)
17 Отключение выбранного передатчика и отмена отключения выбранного передатчика (STS)
16 Передача последней команды (TLC)
15 Передача слова статуса (TSW)
14 Сброс широковещательно транслирующего удаленного терминала (RRTB)
13 Сброс удаленного терминала (RRT)
12 Запрещение и отмена запрещения терминального флага широковещательного транслирования (ITFB)
11 Запрещение и отмена запрещения терминального флага (ITF)
10 Запуск самопроверки широковещательного транслирования (ISTB)
9 Запуск самопроверки (IST)
8 Динамическое управление шины (DBC)
7 Передача слова BIT (TBW)
6 Передача векторного слова (TVW)

5 Отключение передатчика, отмена отключения передатчика широковещательного транслирования (TSB)
4 Отключение передатчика, отмена отключения передатчика (TS)
3 Синхронизация широковещательного транслирования со словом данных (SDB)
2 Синхронизация со словом данных (SD)
1 Синхронизация широковещательного транслирования (SB)
0 Синхронизация (S)

Таблица 4.16.47 – Начало буфера регистрации BM GR1553B (BMSLB)

31 0
BM LOG BUFFER START
0x00000000
rw

31–0 Указатель на младший адрес буфера регистрации BM (с выравниванием до 8 байт). Из-за выравнивания биты 2–0 всегда сброшены

Таблица 4.16.48 – Окончание буфера регистрации BM GR1553B (BMELB)

31 22	21 3	2 0
–	BM LOG BUFFER END	1
0x000	0x00000	0x7
r	rw	r

31–0 Указатель на старший адрес буфера записи BM. Устанавливаются только биты 21–3, т.е. буфер не может превышать 4 Мбайт. Считывание битов 31–22 осуществляется так же, как в стартовом адресе буфера. Из-за выравнивания биты 2–0 всегда установлены

Таблица 4.16.49 – Позиция буфера регистрации BM GR1553B (BMPLB)

31 22	21 3	2 0
–	BM LOG BUFFER POSITION	0
0x000	0x00000	0x0
r	rw	r

31–0 Указатель следующей позиции буфера регистрации BM, в которую будет выполняться запись. Устанавливаются только биты 21–3, т. е. буфер не может превышать 4 Мбайт. Считывание битов 31–22 осуществляется так же, как в стартовом адресе буфера. Из-за выравнивания биты 2–0 всегда сброшены

Таблица 4.16.50 – Регистр управления метки времени BM GR1553B (BMTT)

31 24	23 0
TIME TAG RESOLUTION	TIME TAG VALUE
0x00	0x000000
rw	rw

31–24 Разрешение метки времени (TRES) – Единица времени счетчика меток времени BM в микросекундах, минус 1
23–0 Значение метки времени (TVAL) – Текущее значение счетчика меток времени в рабочем режиме

4.17 Интерфейс SpaceWire с поддержкой RMAP

4.17.1 Общие сведения

В составе процессоров 1906ВМ016 и 1906ВМ01А6 имеется четыре модуля SpaceWire (GRSPW2) с базовыми адресами регистров 0x80100500, 0x80100600, 0x80100700, 0x80100800 (в соответствии с подразделом «2.5»). В каждый модуль SpaceWire интегрирован блок физического уровня (PHY) с резистором согласования 100 Ом. На рисунке 4.17.1 приведена упрощенная схема, показывающая подключение внутреннего согласующего резистора. Следует обратить внимание на тот факт, что драйверы всех четырех каналов SpaceWire по сбросу находятся в режиме PowerDown. Для перевода их в рабочий режим следует использовать регистр управления драйверами LVDS контроллера конфигурации.

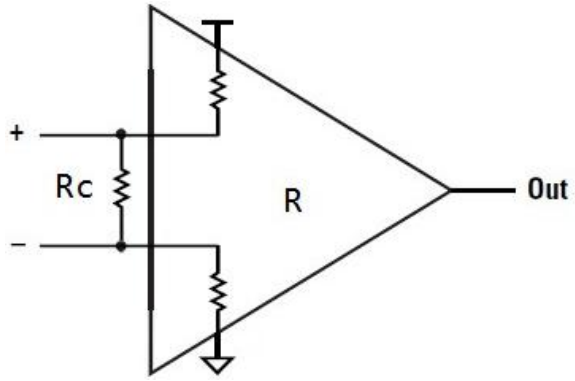


Рисунок 4.17.1 – Упрощенная схема подключения резистора согласования

Устройство SpaceWire обеспечивает интерфейс между шиной AHB и сетью SpaceWire. Данное устройство реализует стандарт SpaceWire (ECSS-E-ST-50-12C) с расширением идентификации протокола (ECSS-E-ST-50-51C). Дополнительный протокол доступа к удаленному запоминающему устройству (RMAP) реализует стандарт ECSS (ECSS-E-ST-50-52C).

Интерфейс SpaceWire настраивается через регистры, которые доступны через интерфейс APB. Данные передаются через канал DMA (Direct Memory Access), используя ведущий интерфейс AHB.

Используется три тактовых домена: для интерфейса AHB (системный тактовый сигнал), для передатчика и для приемника.

Устройство поддерживает только 32-битовые хост-системы с порядком следования «сначала старший байт» с байтовой адресацией. Передача осуществляется по положительному фронту тактового сигнала (Single Data Rate, SDR). Захват данных приёмником производится по обоим фронтам тактового сигнала (Double Data Rate, DDR).

4.17.2 Принцип работы

Общие сведения

Основные подблоки: интерфейс связи, RMAP и интерфейс AMBA. Блок-схема внутренней структуры изображена на рисунке 4.17.2.

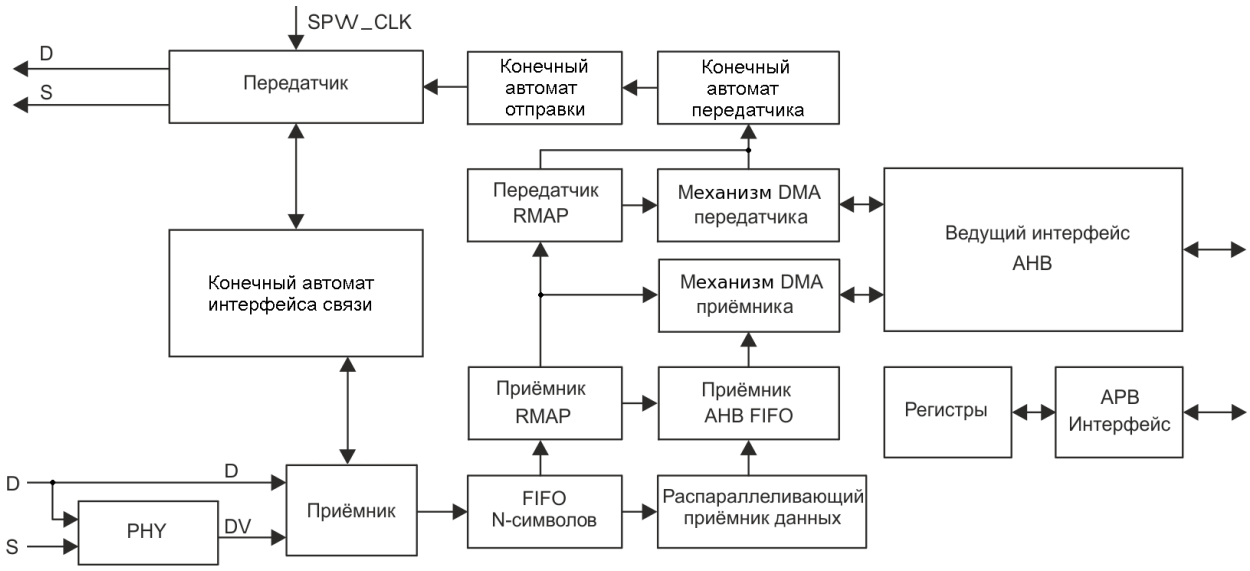


Рисунок 4.17.2 – Блок-схема

Интерфейс связи состоит из приемника, передатчика и конечного автомата (КА) интерфейса связи. Они обеспечивают связь по сети SpaceWire. Блок PHY представляет собой отдельный модуль, его краткое описание приведено в пункте «4.17.3». Интерфейс AMBA состоит из механизма DMA, интерфейса ведущего устройства AHB и интерфейса APB. Интерфейс связи обеспечивает связь с механизмом DMA через FIFO. Данные буферы используются для передачи N-символов (N-Char, normal-characters, информационные символы) между доменами шины AMBA и SpaceWire в ходе приема и передачи.

RMAP приёмник обрабатывает входящие пакеты вместо механизма DMA, если их содержимое идентифицировано как команды RMAP. Команда RMAP декодируется, и, в случае допустимости, на шине AHB выполняется соответствующая операция. При запросе ответа передатчик RMAP автоматически возвращает его источнику.

Описание интерфейса связи, механизмов DMA, RMAP и интерфейса AMBA приводится в пунктах 4.17.5, 4.17.6, 4.17.7, 4.17.8 данного РП.

Поддержка протокола

Устройство принимает пакеты только с действительным адресом назначения в первом получаемом байте. Пакеты с другими адресами без каких-либо уведомлений будут проигнорированы (за исключением «неразборчивого» режима, см. заключительный подзаголовок пункта «4.17.5»).

Следующий байт иногда интерпретируется как протокол ID, который описывается ниже. Протокол RMAP (ID=0x1) единственный протокол, манипуляции над которым осуществляются отдельно на аппаратном уровне, тогда как другие пакеты сохраняются через канал DMA. Если RMAP разрешен, все команды RMAP обрабатываются, выполняются и получают ответ автоматически через аппаратные средства RMAP. Иначе команды RMAP сохраняются через канал DMA, таким же образом, как и другие пакеты. Ответы RMAP всегда сохраняются через канал DMA. Более подробная информация о поддержке протокола RMAP находится в пункте «4.17.7». Когда RMAP отключен, нет необходимости включать идентификатор протокола в пакетах и данные могут начаться непосредственно после адреса.

Все пакеты, поступающие с расширенным протоколом ID (0x00), сохраняются через канал DMA. Это означает, что аппаратные средства RMAP не будут работать, если входящие пакеты RMAP используют расширенную идентификацию протокола. Следует отметить, что пакеты с расширенной идентификацией протокола (ID = 0x000000)

не игнорируются устройством. Необходимость игнорирования пакетов определяет пользователь, принимающий данный пакет.

При передаче пакетов поля адреса и идентификации протокола должны быть включены в буферы, где данные извлекаются. Они не добавляются автоматически.

В таблице 4.17.1 представлены типы пакетов, поддерживаемые системой. Также разрешается прием и передача с идентификатором расширенного протокола, но без поддержки расчетов CRC RMAP и целевого RMAP.

Таблица 4.17.1 – Типы пакетов SpaceWire, поддерживаемых GRSPW2

Addr	ProtID	D0	D1	D2	D3	…	Dn–2	Dn–1	EOP
Addr	D0	D1	D2	D3	D4	…	Dm–2	Dm–1	EOP

4.17.3 Интерфейс связи

Интерфейс связи обеспечивает связь по сети SpaceWire и состоит из передатчика, приемника, интерфейсов КА и FIFO.

Конечный автомат интерфейса связи

Конечный автомат (КА) контролирует интерфейс связи (для получения более подробной информации – см. стандарт SpaceWire). Обработка протокола нижнего уровня (сигнальный и символьный уровень стандарта SpaceWire) выполняется передатчиком и приёмником, в то время как КА занимается обработкой уровня обмена.

Интерфейс конечного автомата управляется через регистр управления. Линия связи может быть отключена с помощью бита запрета соединения (LD), который, в зависимости от текущего статуса, либо блокирует интерфейс связи либо принудительно переводит его в состояние сброса после сбоя (ErrorReset). Если соединение разрешено, КА интерфейса связи запускается при условии, что установлен бит старта соединения (LS) или получен NULL-символ и установлен бит автозапуска (AS).

Текущий статус интерфейса связи определяет тип символов, доступных для передачи, который вместе с запросами от хост-интерфейсов определяет, что будет отправлено.

Передача системного времени (тайм-кодов) осуществляется, если КА находится в рабочем режиме (Run) и был получен соответствующий запрос от интерфейса синхронизации (подробнее об этом – в пункте «4.17.4»).

Если интерфейс связи имеет статус соединение (Connecting) или рабочий режим (Run), то разрешена отправка символов управления потоком (FCT). FCT отправляются интерфейсом связи автоматически, когда это возможно. Данные действия производятся на основании текущего свободного места в приёмном буфере N-символов (normal characters, информационных символов) и определенного в стандарте максимального значения счетчика разрешений ожидающих передач данных – 56. Отправка FCT осуществляется до тех пор, пока в FIFO имеется место как минимум под 8 записей (объём FIFO N-символов приемника см. в пункте «4.17.9») и значение счетчика разрешений, ожидающих передачи данных, меньше или равно 48.

N-символы отправляются в рабочем режиме (Run), если они доступны из FIFO передатчика, а также, если доступны разрешения на передачу. NULL-символы отправляются при отсутствии запросов на передачу других символов или если КА имеет статус, который не разрешает выполнения других передач.

Счетчик разрешений на передачу (входящие разрешения на передачу) автоматически увеличивается на 8 при получении FCT и уменьшается на 1 при передаче каждого N-символа. Полученные N-символы сохраняются в FIFO N-символов приёмника

для последующей обработки интерфейсом DMA. Полученные коды синхронизации (тайм-коды) обрабатываются интерфейсом синхронизации.

Передатчик

Состояние FSM, счетчики разрешений на передачу, запросы интерфейса синхронизации и интерфейса DMA используются для определения следующего передаваемого символа. Тип символа и сам символ (для N-символов и тайм-кодов) для передачи находятся в передатчике нижнего уровня, который расположен в отдельном тактовом домене.

Это необходимо в связи с тем, что обычно связь SpaceWire запускается с частотой, не соответствующей частоте тактирования системы. В устройстве имеется отдельный тактовый вход, который используется для генерации тактирования передатчика. Поскольку передатчик нередко работает с большой частотой тактирования (более 100 МГц), для минимизации потребления энергии и обеспечения синхронизации вся логика, которая это позволяет, размещена в тактовом домене системы.

Логика передатчика в тактовом домене системы определяет какой символ послать следующим, устанавливает надлежащие управляющие сигналы, а также обеспечивает низкоуровневую передачу – см. рисунок 4.17.3. Передатчик отправляет запрошенные символы и генерирует соответствующие биты четности и управления. До тех пор, пока передача разрешена, при отсутствии запросов домена хост-системы отправляются NULL. Большинство сигнальных и символьных уровней стандарта SpaceWire обрабатывается в передатчике. Для сигналов данных и строба необходим внешний LVDS драйвер. Выходы сконфигурированы с одинарной скоростью передачи данных (SDR).

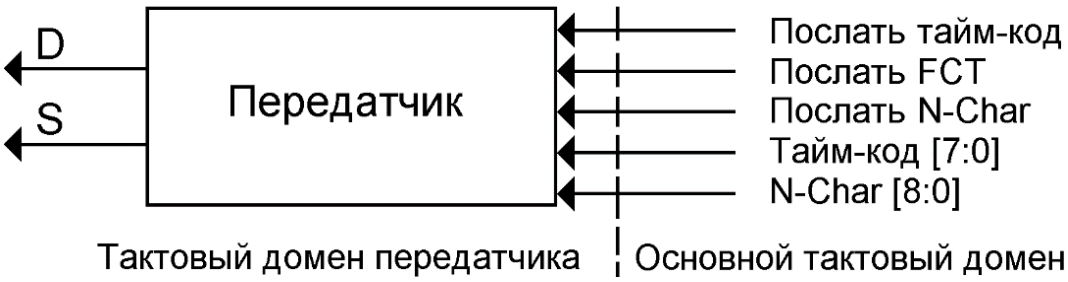


Рисунок 4.17.3 – Схема передатчика с интерфейсом связи

КА передатчика считывает N-символы для передачи из FIFO передатчика. Интерфейс DMA передает сведения о длине пакета, при этом КА передатчика при необходимости добавляет значения EOP/EEP и CRC RMAP. По окончании обработки пакета интерфейс DMA получает соответствующее уведомление и выдается новое значение длины пакета.

Приемник

Приёмник обнаруживает подключения других узлов и получает символы в виде битового потока, восстановленного из сигналов данных и строба модулем PHY, который представляет их в качестве сигнала данных и сигнала действительности данных. Приёмник и PHY расположены в отдельном тактовом домене, который работает от тактового сигнала, генерируемого модулем PHY.

Приёмник включается при выходе интерфейса связи из состояния сброса после сбоя (ErrorReset). После приема NULL-символа приёмник может начать принимать любые символы. Он обнаруживает ошибки по четности, ошибки перехода и ошибки счетчика разрешений на передачу, которые приводят к тому, что интерфейс связи входит в

состояние сброса. Рассоединения обрабатываются в интерфейсе связи в тактовом домене передатчика, так как тактирование приёмника при такой ситуации не доступно.

Принятые символы отмечаются как принадлежащие хост-домену, данные представлены в параллельном виде. Интерфейс хост-домена показан на рисунке 4.17.4.

L-символы (L-Char, link-символы, канальные символы) обрабатываются автоматически интерфейсом связи хост-домена, все N-символы сохраняются в FIFO приёмника для последующей обработки. При приеме двух или более последовательных EOP/EEP все, кроме первого из них, отбрасываются.

Рисунок 4.17.4 – Схема приемника с интерфейсом связи

Приёмник физического уровня PHY

Приёмник обеспечивает восстановление входных данных выборкой DDR. Обработка выполняется в блоке физического уровня, и данные передаются в приёмник в виде сигнала данных и сигнала корректности данных.

Установка скорости канала связи

Поле CLKDIVSTART регистра тактового делителя определяет скоростной режим во время инициализации (все состояния вплоть до «соединение» включительно). Этот регистр также используется для вычисления тайм-аута интерфейса FSM (6,4 мкс и 12,8 мкс, как определено в стандарте SpaceWire). Значение поля CLKDIVSTART всегда должно быть установлено так, чтобы во время инициализации выдерживалась частота в 10 МГц. При выполнении данного условия величина тайм-аута рассоединения также будет рассчитана корректно.

Для того чтобы рассчитать требуемое значение CLKDIVSTART, следует воспользоваться формулой CLKDIVSTART = («SPW_CLK в МГц» / 10) – 1.

Скорость передачи в рабочем режиме задается с помощью соответствующего коэффициента делителя CLKDIVRUN (поле регистра тактового делителя) и рассчитывается по следующей формуле:

«Скорость в рабочем режиме в Мбит/с» = «SPW_CLK в МГц» / (CLKDIVRUN + 1)

Примечание – При подключении к трансиверу Cobham SpaceWire существует ограничение на используемые значения делителя. Все четные значения (кроме 0) будут обеспечивать ту же скорость передачи бит, что и соответствующие им нечетные значения, которые больше на 1.

4.17.4 Распространение системного времени (тайм-кодов)

Тайм-коды являются кодами управления, которые состоят из двух флагов контроля (биты 7:6) и значений системного времени (биты 5:0). Они используются для

распространения времени в сети SpaceWire. Текущее значение времени (значение последних полученных или переданных тайм-кодов) и управляющие флаги могут быть прочитаны из регистра системного времени (тайм-кодов).

Прием тайм-кодов

Когда управляющий код принят, а также флаги контроля (биты 7:6) имеют значение «00» или сброшен бит фильтра флагов контроля тайм-кодов (TF) регистра управления, тогда принятый управляющий код считается тайм-кодом.

Если разрешен прием тайм-кодов (бит TF регистра управления установлен в 1), то полученное значение времени сохраняется в поле TIMECN регистра системного времени (тайм-кодов). Если полученное значение времени равно TIMECNT+1 (по модулю 64), тогда тайм-код считается действительным.

Когда принят действительный тайм-код, в дополнение к обновлению значения времени, полученные контрольные флаги сохраняются в поле TCTRL. Так же, когда принят действительный тайм-код, устанавливается бит TO регистра статуса и генерируется прерывание на шине AMBA, если биты разрешения прерываний (IE) и прерывания при получении тайм-кода (TQ) регистра управления установлены.

Передача тайм-кодов

Для того чтобы передать тайм-код, требуется установить бит входного отсчета (TI) регистра управления. Когда бит установлен, текущее значение времени (поле TIMECNT регистра системного времени (тайм-кодов)) инкрементируется (по модулю 64) и тайм-код, состоящий из нового значения времени и текущих флагов контроля (поле TC регистра системного времени (тайм-кодов)), отсылается. Бит TI будет оставаться в установленном состоянии до тех пор, пока тайм-код будет передаваться. Данная операция предварительно требует установки бита разрешения передачи тайм-кодов (TT) регистра управления. Если передача тайм-кодов запрещена, то запись в бит TI не возымеет эффекта.

Обратите внимание, что интерфейс связи должен быть в состоянии «Run» для того, чтобы иметь возможность отправить тайм-код.

4.17.5 Приёмный канал DMA

Механизм DMA приёмника управляет приемом данных из сети SpaceWire по каналу DMA.

Сравнение адреса

Прием пакетов каналом осуществляется на основе адреса и в зависимости от того, разрешен канал или нет.

Когда FIFO N-символов приёмника содержит один или более символов, N-символы считываются с помощью механизма DMA приёмника. Первый символ интерпретируется как логический адрес, пакет будет сохранен с использованием DMA, если его адрес будет соответствовать адресу канала DMA. Указатель на область памяти, в которой сохранен полный пакет, включая адрес и идентификатор протокола, но исключая EOP/EEP, и хранится в дескрипторе канала (объяснено позже в этом пункте).

На рисунке 4.17.5 показана блок-схема приема пакета.

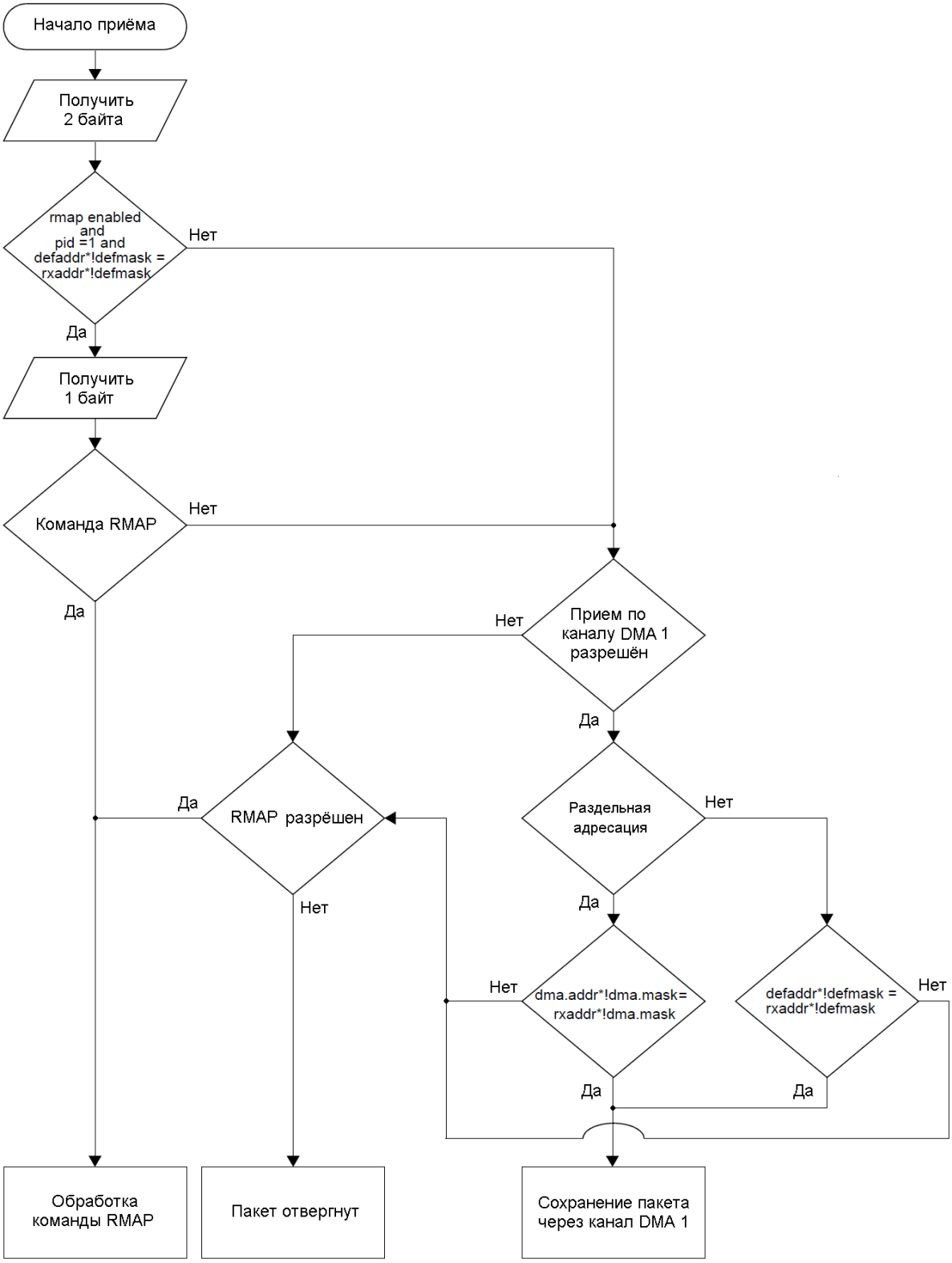


Рисунок 4.17.5 – Блок-схема приема пакетов

Каждый регистр адреса SpaceWire имеет соответствующий регистр масок. Только биты, содержащие значение 0, в соответствующем регистре масок сравниваются. Таким образом, канал DMA может принять ряд адресов. Для проверки соответствия адреса в канале DMA, если для него не установлен бит раздельной адресации, и для идентификации команд RMAP в приемнике RMAP используется регистр адреса по умолчанию. Если в регистре управления каналом DMA разрешена раздельная адресация, то для сравнения используется пара собственных регистров адреса/маски адреса.

Команда RMAP принимается в обработку, только если значение принятого адреса (включая маску) соответствует значению регистра адреса по умолчанию. В противном случае пакет будет сохранен через канал DMA, если он имеет соответствующий адрес. Если адрес не совпадает ни со значением адреса по умолчанию, ни со значением регистра адреса канала DMA, то пакет будет обрабатываться приемником RMAP (если он разрешен), так как он должен вернуть код ошибки неверного адреса. Пакет сбрасывается (вплоть до следующего EOP/EEP), только когда адрес не найден и приёмник RMAP не разрешен.

Пакеты, кроме команд RMAP, адрес которых не соответствует ни адресному регистру DMA, ни адресному регистру по умолчанию, будут игнорированы.

По крайней мере два N-символа, отличающихся от EOP/EEP, должны быть получены для сохранения пакета, предназначенного для канала DMA, пока не разрешен «неразборчивый» режим, в котором достаточно одного N-символа. Если это RMAP пакет и RMAP разрешен, необходимо три N-символа, так как старший байт определяет, где обрабатывается пакет. Пакеты меньших размеров игнорируются.

Основные функциональные возможности канала

Работа приёмника базируется на последовательно расположенных в памяти дескрипторах, которые содержат указатели на буферы, в которые должны быть сохранены пакеты. Дескриптор считывается из области дескрипторов канала DMA, и принятый пакет сохраняется в область памяти, указываемую в дескрипторе. В заключении статус сохраняется в текущий дескриптор, и производится автоматический переход к очередному дескриптору. В следующих подпунктах прием через канал DMA будет описан более подробно.

Подготовка устройства для приёма

Перед приемом некоторые регистры должны быть инициализированы. Первоначально, перед отправкой данных, необходимо перевести интерфейс связи в рабочий режим (Run). Канал DMA имеет регистр максимального размера принимаемого пакета, который задает максимальный размер пакета для получения каналом. Большие пакеты усекаются, и данные, превысившие обозначенный предел, сбрасываются. Если это произошло, то данный факт будет отражен в поле статуса дескриптора. Минимальное значение поля максимального размера приемника – 4, при этом значение можно увеличивать с шагом в четыре байта до максимального значения 33 554 428. Если максимальный размер установлен в «0», нормальная работа приемника не гарантируется.

Также должен быть установлен либо регистр адреса по умолчанию, либо регистр адреса канала DMA. Бит разрешения адреса (EN) регистра управления/статуса канала DMA определяет, какая из пар адрес/маска адреса будет использована для сравнения с адресом принимаемого пакета.

Наконец, таблица дескрипторов и регистр управления также должны быть инициализированы. Этот процесс описан в двух следующих подпунктах.

Установка адреса таблицы дескрипторов

Устройство считывает дескрипторы из области в памяти, на которую указывает регистр адреса таблицы дескрипторов приёмника. Регистр состоит из поля базового адреса и селектора дескрипторов. Базовый адрес указывает на начало области и должен быть выровнен относительно размера таблицы дескрипторов. Размер этой таблицы в байтах определяется по формуле: NRXD × 8. Поле NRXD регистра статуса определяет количество записей в таблице, каждый дескриптор занимает 8 байт.

Поле селектора дескрипторов указывает на отдельные дескрипторы и увеличивается на единицу после использования текущего дескриптора. При достижении

значением поля верхней границы, он автоматически переводится на начало. Так же подобный переход может быть осуществлен до достижения верхнего предела, путем установки бита перехода на начало в дескрипторе. Идея заключается в том, что поле селектора должно быть инициализировано значением «0» (начало области дескрипторов), но в него может быть записано и другое значение, выровненное до 8 байт для запуска где-либо в середине области. При достижении селектором верхней границы он все равно будет переходить в начало области дескрипторов.

При необходимости использования новой таблицы дескрипторов в первую очередь следует сбросить бит разрешения приёмника (RE) регистра управления/статуса канала DMA. Если бит RX (осуществляется приём) данного регистра сброшен, то обновление регистра адреса таблицы дескрипторов приёмника безопасно. После завершения обновления и установки бита доступности нового дескриптора приёмника (RD) можно снова установить бит разрешения приёмника (RE).

Разрешение на использование дескрипторов

Как указывалось выше, для обеспечения приема необходимо разрешить использование одного или более дескрипторов. Размер каждого дескриптора – 8 байт (см. в таблицах ниже). Дескрипторы должны быть записаны в область памяти, на которую указывает регистр адреса таблицы дескрипторов приёмника. В случае добавления новых указателей, они должны размещаться после предыдущего, записанного в область. В противном случае они не учитываются.

Активация дескриптора осуществляется путем установки адреса указателя на область памяти, отведенную для сохранения данных, после чего устанавливается бит разрешения на использование (EN). Бит WR устанавливается для сброса поля селектора дескриптора регистра адреса таблицы дескрипторов приёмника после завершения приема по данному дескриптору. Бит IE должен быть установлен, если после завершения приема требуется прерывание. Бит разрешения прерывания (RI) регистра управления/статуса канала DMA так же должен быть установлен для генерации прерывания.

Адрес пакета дескриптора не обязан быть выровнен по границе слова. Любое количество байт может быть получено по любому возможному адресу пакета без какой-либо избыточной перезаписи. Отдельно взятый дескриптор представлен в таблицах 4.17.2 и 4.17.3.

Таблица 4.17.2 – Слово «0» дескриптора приема GRSPW2 (смещение адреса 0x0)

31	30	29	28	27	26	25	24 … 0
TR	DC	HC	EP	IE	WR	EN	PACKETLENGTH

31	Усечённый (TR) – пакет усечен ввиду превышения заданного максимального размера
30	CRC данных (DC) – устанавливается, если обнаружена ошибка CRC данных; в противном случае – сброшено
29	CRC заголовка (HC) – устанавливается, если обнаружена ошибка CRC заголовка; в противном случае – сброшено
28	Завершение EEP (EP) – данный пакет завершен с символом «пакет завершен с ошибкой» (Error End of Packet)
27	Разрешение прерывания (IE) – если установлено, после приема пакета генерируется прерывание, если в регистре управления/статуса канала DMA установлен бит разрешения прерывания при приеме (RI)

26	Переход на начало (WR) – если установлено, следующий используемый дескриптор будет первым в таблице дескрипторов (располагаться по базовому адресу). В противном случае указатель на дескриптор увеличивается на 0x08 для обеспечения использования следующего дескриптора, расположенного далее в памяти. При достижении границы таблицы дескрипторов, указанной в предыдущем подпункте, указатель автоматически переходит обратно на базовый адрес
25	Разрешение на использование (EN) – устанавливается для активации дескриптора. Это означает, что дескриптор содержит действительные управляющие значения и область памяти, на которую ссылается поле адреса пакета и может использоваться для сохранения пакета
24–0	Размер пакета (PACKETLENGTH) – количество байт, принятых в данный буфер. Допустимо после сброса EN GRSPW

Таблица 4.17.3 – Слово «1» дескриптора приема GRSPW2 (смещение адреса 0x4)

31	0
PACKETADDRESS	

31–0	Адрес пакета (PACKETADDRESS) – адрес, указывающий на буфер для сохранения принятого пакета

Установка регистра управления DMA

Последний этап для приема пакетов – установка регистра управления: необходимо разрешить приёмник установкой бита RE в регистре управления/статуса канала DMA. Это можно сделать в любой момент: пока бит не установлен, операция не запускается. Бит доступности дескриптора приёмника (RD) также устанавливается для информирования о наличии новых активированных дескрипторов. Это действие всегда должно выполняться после активации дескрипторов; в противном случае система может не обнаружить новые дескрипторы. Остальные дескрипторы могут быть активированы после начала приема при разрешении дескриптора на использование и записи бита RD регистра управления/статуса канала DMA. После установки этих бит приём начинается непосредственно при поступлении данных.

Влияние управляющих бит во время приема

Когда приёмник не разрешен, все пакеты, отправленные в канал DMA, игнорируются. Если приёмник разрешен и адрес попадает в диапазон принимаемых адресов, то осуществляется переход в следующее состояние с проверкой бита доступности дескриптора приёмника (RD). Данный бит указывает на наличие активированных дескрипторов и должен устанавливаться внешним приложением, использующим канал DMA, при каждом разрешении на использование дескрипторов, как обозначено выше. Если бит RD и бит NS регистра управления/статуса канала DMA сброшены, то принимаемые пакеты игнорируются. Если бит NS установлен, система ожидает установки бита RD и символы остаются в FIFO N-символов в течение этого времени. Если FIFO заполняется, дальнейшая передача N-символов запрещается остановкой передачи FCT.

После установки бита RD считывается следующий дескриптор; если он активирован, пакет принимается в буфер. Если считанный указатель не разрешен, то бит RD сбрасывается, далее в зависимости от значения NS пакет может быть сброшен или оставлен в буфере.

Приёмник можно отключить в любой момент, при этом прием текущего пакета будет завершен. Бит доступности дескриптора приёмника (RD) также может быть сброшен в любой момент. При этом текущие приемы не затрагиваются, но новые

дескрипторы не будут считаны до повторной установки данного бита. RD сбрасывается устройством при считывании не активированного дескриптора.

Биты статуса

После завершения приема пакета бит разрешения на использование в текущем дескрипторе сбрасывается, и при сброшенном разрешении биты статуса остаются действительны, а количество принятых байт указывается в поле размера. Регистр управления/статуса канала DMA содержит бит статуса (PR), который устанавливается при каждом приеме пакета. Также устройство может генерировать прерывание по наступлению данного события.

Расчет CRC RMAP ведется всегда для всех принятых пакетов и всех байтов, кроме EOP/EEP. Пакет всегда рассматривается как возможный пакет RMAP, размер его заголовка определяется третьим байтом, который является полем команды. Рассчитанное значение CRC проверяется после приема заголовка (в соответствии с рассчитанным количеством байтов), и если оно ненулевое, то устанавливается бит HC, указывающий на ошибку CRC заголовка.

Значение CRC не сбрасывается после приема заголовка – расчет продолжается таким же образом до получения полного пакета. После этого, если значение CRC не нулевое, устанавливается бит DC, указывающий на ошибку CRC данных. Это означает, что система может идентифицировать ошибку CRC данных, даже если поле данных было правильным при неправильном CRC заголовка. Однако эти данные не должны использоваться при неверном заголовке, поэтому бит DC в данном случае не важен. При действительном заголовке значение CRC всегда будет нулевым, если расчет с полем данных продолжается и поведение будет таким же, как и при перезапуске расчета CRC.

Если тип принятого пакета не соответствует RMAP, то бит, указывающий на ошибку CRC заголовка, не должен быть использован. Использование бита DC в качестве индикатора корректности данных остается возможным, если полный пакет защищен CRC, рассчитанной в соответствии с методикой CRC RMAP. Это происходит потому, что система не перезапускает расчет после приема заголовка и рассчитывает полный CRC по пакету. Таким образом, используя бит DC, можно проверить пакет любого формата с одним CRC в конце пакета, рассчитанным согласно стандарту RMAP.

Если тип принятого пакета не соответствует RMAP, и в то же время в конце пакета используется любой другой тип CRC, кроме описанного в RMAP, то биты HC и DC должны быть проигнорированы.

Обработка ошибок

Если прием пакета должен быть прерван из-за перегрузки в сети, рекомендуется установить бит запрещения соединения (LD) регистра управления. К сожалению, это приведет к усечению передаваемого пакета, но это единственное безопасное решение, поскольку прием пакета – пассивная операция, зависящая от передатчика на другом конце. Также можно установить бит сброса узла (RS), но это не совсем подходящий путь, так как непереданные символы остаются в узле передатчика. Следующий символ (расположенный где-то в середине пакета) будет интерпретирован как адрес узла, что может привести к тому, что пакет будет сброшен (не со 100% уверенностью). Обычно это действие выполняется в случае приостановки приема, вызванной тем, что передатчик больше не предоставляет данные. Сброс соединения не устраняет подобную перегруженность.

При возникновении ошибки AHB в ходе приема текущий пакет сбрасывается вплоть до следующего символа EEP/EOP, после чего активный канал отключается и приёмник переходит в режим ожидания. Для обозначения данного состояния в регистре управления/статуса канала DMA устанавливается бит RA.

«Неразборчивый» режим

В устройстве поддерживается «неразборчивый» режим, в котором все принятые данные сохраняются в канале DMA, независимо от адреса узла и возможных преждевременных EOP/EEP. Это означает, что все принятые N-символы, исключая EOP/EEP, сохраняются через канал DMA. Тем не менее, проверка на соответствие длины принимаемой последовательности данных регистра максимального размера принимаемого пакета остается, при этом пакеты, превышающие указанный граничный размер, усекаются.

Команды RMAP при разрешенном «неразборчивом» режиме обрабатываются RMAP, если установлен бит RE регистра управления. Если этот бит разрешения RMAP сброшен, команды RMAP также сохраняются в канале DMA.

4.17.6 Передающий канал DMA

Передатчик данных посредством механизма DMA выполняет пересылку данных через канал DMA в сеть SpaceWire.

Базовая функциональность

DMA передатчика считывает данные с шины AHB и сохраняет их в FIFO передатчика для передачи по сети SpaceWire. Данная функциональность базируется на дескрипторах такого же типа, что и для приёмника; таблица дескрипторов имеет такое же выравнивание по адресу и ограничения по размеру. При активациях новых дескрипторов система считывает их и передает данные предписанного размера.

Подготовка устройства для передачи

Перед передачей должны быть выполнены четыре этапа. Сначала необходимо разрешить и запустить интерфейс связи записью соответствующего значения в регистр управления. После этого адрес таблицы дескрипторов следует записать в регистр адреса таблицы дескрипторов передатчика, при этом в таблице должны быть разрешены на использование один или более дескрипторов. В заключение в регистр управления/статуса канала DMA следует записать «1» в бит TE – после этого запускается передача. Более подробное описание – в последующих подпунктах.

Разрешение на использование дескрипторов

Устройство считывает дескрипторы из области в памяти, на которую указывает регистр адреса таблицы дескрипторов передатчика. Регистр состоит из поля базового адреса и селектора дескрипторов. Базовый адрес указывает на начало области и должен быть выровнен относительно размера таблицы дескрипторов. Размер этой таблицы в байтах определяется по формуле: NTXD × 16. Поле NTXD регистра статуса определяет количество записей в таблице, каждый дескриптор занимает 16 байт.

Для передачи пакетов один или несколько дескрипторов должны быть инициализированы в памяти. Для этого следует установить количество байт для передачи и указатель на область данных. Две различных длины и два поля адреса в дескрипторах передатчика используются потому, что указатели на заголовок и данные разнесены. Если длина поля равна нулю, то соответствующая часть пакета будет пропущена, если оба поля равны нулю, то пакет не отправляется. Максимальная длина заголовка – 255 байт, максимальная длина данных 16 Мбайт минус 1 байт. Когда указатель и поле размера установлены, для активации дескриптора необходимо установить бит разрешения на

использование дескриптора (EN). Данный бит устанавливается в последнюю очередь. Другие биты управления также должны быть установлены перед активацией дескриптора.

Размер дескриптора передатчика – 16 байт, максимальное количество в одной таблице – 64 (значение поля NTXD регистра статуса). Различные поля дескриптора и смещение памяти представлены в таблицах ниже.

Бит HC должен быть установлен, если требуется расчет и вставка CRC RMAP в состав заголовка, соответственно бит DC должен быть установлен для проведения аналогичной процедуры для поля данных. CRC заголовка рассчитывается на основе данных, извлеченных по указателю на заголовок (HEADERADDRESS); генерация CRC данных выполняется на основе данных, извлеченных по указателю на данные (DATAADDRESS). CRC добавляется после соответствующего поля. Поле байтов NON-CRC определяет количество байт в начале поля заголовка, которые не будут включены в расчет CRC.

CRC отправляется даже в случае, если соответствующее поле размера равняется нулю. Пакет не отправляется, включая EOP, только если оба поля размера (и заголовка и данных) сброшены.

Запуск передач

Чтобы система запустила передачу, после инициализации дескрипторов в регистре управления/статуса канала DMA должен быть установлен бит разрешения передачи (TE). Новые дескрипторы можно активировать в таблице уже в процессе работы (при запущенной передаче). При каждом добавлении набора дескрипторов должен быть установлен бит разрешения передачи, т.к. при каждом обнаружении устройством не активированного дескриптора, бит TE сбрасывается. Отдельно взятый дескриптор представлен в таблицах 4.17.4 – 4.17.7.

Таблица 4.17.4 – Слово «0» дескриптора передатчика GRSPW2 (смещение адреса 0x0)

31 18	17	16	15	14	13	12	11 8	7 0
–	DC	HC	LE	IE	WR	EN	NONCRCLEN	HEADERLEN

31–18	Зарезервировано
17	Добавить CRC данных (DC) – Добавить CRC, рассчитанный согласно спецификации RMAP, после данных, отправленных из области памяти к которой ведет указатель на данные (DATAADDRESS). CRC распространяется на все данные по этому указателю. Нулевой CRC отправляется, если размер поля данных – нулевой
16	Добавить CRC заголовка (HC) – Добавить CRC, рассчитанный согласно спецификации RMAP после данных, отправленных из области памяти к которой ведет указатель на заголовок (HEADERADDRESS). CRC распространяется на все данные по этому указателю, за исключением количества байт в начале, в соответствии со значением поля байт Non-CRC. CRC не отправляется, если размер поля заголовка – нулевой
15	Ошибка соединения (LE) – Ошибка соединения в ходе передачи пакета
14	Разрешение прерывания (IE) – Если установлено, генерируется прерывание после передачи пакета при условии, что бит разрешения прерывания передатчика (RI) регистра управления/статуса канала DMA также установлен

13 Переход на начало (WR) – Если установлено, селектор дескриптора сбрасывается, соответственно следующий считанный дескриптор будет первым в таблице дескрипторов (располагаться по базовому адресу). В противном случае, указатель на дескриптор увеличивается на 0x10 для обеспечения использования следующего дескриптора, расположенного далее в памяти

12 Разрешение (EN) – Разрешение дескриптора передатчика. Если установлены все поля управления (адреса, размера, перехода на начало и CRC), данный бит следует установить. При установленном бите дескриптор не должен быть изменён во избежание нарушения передачи. SpaceWire сбрасывает данный бит после завершения передачи

11–8 Байты Non-CRC (NONCRCLEN) – Устанавливает количество байт в начале заголовка, которые не включены в расчет CRC. Это требуется при использовании путевой адресации, т.к. один или более байтов в начале пакета могут быть отброшены до того как пакет достигнет узла назначения

7–0 Размер заголовка (HEADERLEN) – Размер заголовка в байтах. Если сброшено, заголовок пропускается

Таблица 4.17.5 – Слово «1» дескриптора передатчика GRSPW2 (смещение адреса 0x4)

31 0
HEADERADDRESS

31–0 Адрес заголовка (HEADERADDRESS) – Адрес, по которому расположен заголовок пакета. Выравнивание до границы слова не обязательно

Таблица 4.17.6 – Слово «2» дескриптора передатчика GRSPW2 (смещение адреса 0x8)

31 24	23 0
–	DATALEN

31–24 Зарезервировано

23–0 Размер данных (DATALEN) – Размер данных пакета. Если сброшено, данные не отправляются. Если размеры данных и заголовка сброшены, пакет не отправляется

Таблица 4.17.7 – Слово «3» дескриптора передатчика GRSPW2 (смещение адреса 0xC)

31 0
DATAADDRESS

31–0 Адрес данных (DATAADDRESS) – Адрес, по которому считываются данные. Выравнивание до границы слова необязательно

Процесс передачи

После установки бита TE регистра управления/статуса канала DMA сразу же начинается считывание дескриптора. Количество обозначенных байтов считывается и передается. После завершения передачи статус записывается в первое слово дескриптора, при этом в регистре управления/статуса канала DMA устанавливается бит пакет отправлен (PS). Если было запрошено прерывание, то оно генерируется. После этого считывается новый дескриптор; если он активирован, запускается новая передача. В противном случае, бит разрешения передачи (TE) сбрасывается (процесс останавливается до повторной установки данного бита).

Регистр адреса таблицы дескрипторов

Внутренний указатель, который используется для хранения текущего положения в таблице дескрипторов, можно считывать и записывать через интерфейс APB. Этот указатель устанавливается в ноль при выполнении сброса и инкрементируется каждый раз при использовании очередного дескриптора. Переход на начало выполняется автоматически при достижении границы таблицы дескрипторов (см. подпункт «Разрешение на использование дескрипторов») или раньше, если в текущем дескрипторе установлен соответствующий бит.

Регистр адреса таблицы дескрипторов передатчика может быть обновлен новой таблицей в любое время, если не ведется передача. Передача не является активной, если бит разрешения передачи (TE) равен нулю и полная таблица была отправлена, или если таблица прервана (описано ниже). Если таблица прервана, необходимо подождать сброса бита TE перед тем, как обновлять указатель на таблицу.

Обработка ошибок

Прерванная передача

Регистр управления/статуса канала DMA содержит бит «прервать передачу» (AT), установка которого вызывает прекращение текущей передачи, при этом пакет усекается и вставляется EEP. Это может потребоваться, если только требуется прерывание пакета в случае перегрузки сети SpaceWire. В случае перегрузки на шине AHB установка данного бита не улучшит ситуацию (это не является проблемой, поскольку ведомые устройства AHB имеют максимум 16 состояний ожидания). В случае прерывания пакета в дескрипторе устанавливается бит LE. Бит регистра разрешения передачи (TE) регистра управления/статуса канала DMA также сбрасывается; новые передачи не выполняются до следующего разрешения передатчика.

Ошибка AHB

При обнаружении ошибки AHB в ходе передачи активный канал DMA отключается и передатчик переходит в режим ожидания. В регистр управления/статуса канала DMA устанавливается соответствующий бит, характеризующий ошибку, и если разрешено, генерируется прерывание. Дальнейшая обработка ошибок зависит от статуса передатчика механизма DMA на момент возникновения ошибки AHB. Если дескриптор был считан, а передача пакета еще не началась, то никаких дополнительных действий больше не требуется.

Если ошибка AHB возникла в ходе передачи пакета, пакет усекается и вставляется EEP. Если ошибка возникла в ходе записи статуса в дескриптор, пакет успешно передан, но дескриптор не перезаписывается и остается активированным (это также означает, что биты ошибки в дескрипторе не устанавливаются для случая ошибок на шине AHB).

Пользователь канала должен исправить ошибку AHB и повторно активировать канал. Другие передачи AHB из блока (передатчика или приемника), который был задействован в момент обнаружения ошибки AHB, не выполняются до сброса состояния ошибки и повторного разрешения данного блока.

Ошибка соединения

При обнаружении ошибки связи в ходе передачи оставшаяся часть пакета отбраковывается вплоть до следующего EOP/EEP включительно. После этого статус

незамедлительно записывается в дескриптор (бит LE установлен) и указатель на дескриптор инкрементируется. Соединение разрывается при возникновении ошибки соединения, но система автоматически пытается снова восстановить связь при условии, что бит запуска связи (LS) регистра управления установлен и бит отключения связи (LD) регистра управления сброшен. Если бит LE в регистре управления/статуса канала DMA не установлен, механизм DMA передатчика будет ожидать осуществления перехода в рабочий режим (Run), после чего немедленно начинается новая передача при условии, что пакеты находятся в режиме ожидания. Если бит LE в регистре управления/статуса канала DMA установлен, то передатчик будет отключен при возникновении ошибки соединения, произошедшей в ходе передачи текущего пакета. В этом случае пакеты не будут передаваться до повторного разрешения передатчика.

4.17.7 RMAP

Протокол удаленного доступа к памяти (RMAP) используется для осуществления доступа к ресурсам в узле через интерфейс SpaceWire. Стандартные операции: считывание и запись в память, регистры и FIFO. В данном пункте приводятся основные принципы работы протокола RMAP и его реализации.

Основные принципы работы протокола

RMAP – это протокол, обеспечивающий удаленный доступ к ресурсам, отображаемым в памяти, через сеть SpaceWire в узле SpaceWire. Ему присвоен идентификатор протокола 0x01. Он обеспечивает три операции: запись, считывание и считывание-модификация-запись. Эти операции являются отложенными, это означает, что источник не будет дожидаться подтверждения или ответа. Это также означает, что в любой момент времени любое количество операций может ожидать выполнения, и что в данном протоколе не реализован режим ожидания. Режим ожидания должен быть реализован в пользовательском приложении, которое посылает команды. У принимающей стороны можно запросить отправку ответа и проверку целостности данных перед выполнением операции. 16 Мбайт минус 1 байт – максимально возможный размер полезной нагрузки данных, поддерживаемый в протоколе. Описание протокола – см. стандарт RMAP (ECSS-E-ST-50-52C).

Реализация

Устройство включает приёмник команд RMAP, который обрабатывает все входящие пакеты с идентификатором протокола PID = 0x01 и полем типа (бит 7 и 6 третьего байта в пакете) равным «01», если адрес попадает в диапазон, задаваемый адресом и маской регистра адреса по умолчанию. При обнаружении такого пакета он не сохраняется через канал DMA, а передается в приёмник RMAP.

Все три команды, определенные в стандарте, реализованы с некоторыми ограничениями. Поддерживаются только 32-разрядные системы с порядком следования «сначала старший байт» (Big-Endian). Это означает, что первый принятый байт в слове – старший значащий байт. Сортировщик команд не принимает пакеты RMAP, имеющие расширенный идентификатор протокола, вместо этого они всегда выгружаются в канал DMA.

Приёмник RMAP обрабатывает команды. Если они корректны и допустимы, то операция выполняется на шине AHB и формируется ответ. В случае запроса подтверждения передатчик RMAP автоматически отправляет ответ. Передачи RMAP имеют приоритет перед передачами канала DMA.

Имеется доступный для пользователя регистр ключа получателя RMAP, содержание которого сравнивается с содержанием поля ключа получателя во входящих

пакетах. В случае несоответствия, если был запрошен ответ, будет отправлен код ошибки 3. Ответы отправляются только в случае, если поле подтверждения установлено.

В случае отказа в ходе выполнения доступа по шине, значение кода ошибки будет равняться 1 (ошибка общего характера). Существует заданный порядок установки кодов ошибок в случае многократных ошибок – см. таблицу 4.17.8.

Таблица 4.17.8 – Порядок обнаружения ошибок в случае многократных ошибок в GRSPW2. Наивысший приоритет обнаружения у ошибки под номером 1

Порядок обнаружения	Код ошибки	Ошибка
1	12	Недействительный логический адрес назначения
2	2	Неиспользуемый тип пакета RMAP или код команды
3	3	Недействительный ключ получателя
4	9	Переполнение буфера проверки
5	11	Ошибка размера данных операции считывание-модификация-запись
6	10	Ошибка авторизации запрошенной операции
7*	1	Ошибка общего характера (ошибки на шине AHB в ходе записей без проверки данных)
8	5/7	Преждевременный EOP/EEP
9	4	Недействительные данные (ошибка CRC)
10	1	Ошибка общего характера (ошибки на шине AHB в ходе записей с проверкой данных или операции считывание-модификация-запись)
11	7	EEP
12	6	Слишком большой объём данных

_____

* Обнаружение ошибки на шине AHB до обнаружения преждевременного EOP/EEP или недействительных данных (ошибка CRC) не гарантируется. При длительных операциях доступа обнаружение ошибки на шине AHB может произойти с некоторой задержкой, поскольку сначала детектируются ошибки преждевременного EOP/EEP или недействительных данных (ошибка CRC).

Доступ на чтение выполняется в процессе работы, т.е. не сохраняется во временном буфере перед передачей. Это означает, что код ошибки 1 не выдается в ответе на операцию считывания, т.к. заголовок уже отправляется, пока данные считываются. В случае возникновения ошибки на шине AHB пакет усекается и отправляется EEP в качестве окончания.

Ошибки вплоть до недействительных данных (ошибка CRC) включительно (номер 9) проверяются перед командами с проверкой данных. Остальные ошибки не препятствуют выполнению операций с проверкой данных.

Все оговоренные команды, которые приняты, но имеют неподдерживаемый набор опций для данной конкретной реализации, не выполняются; в возможном ответе будет отправлен код ошибки 10.

Команды записи

Команды записи по характеристикам делятся на две подкатегории: с предварительной проверкой данных и без нее. Размер записей с проверкой не должен превышать 4 байта, при этом адрес должен быть выровнен до этого размера. Таким образом, записи размером 1 байт могут производиться в любой адрес, записи размером 2 байта должны быть выровнены до границы полуслова, записи размером 3 байта недопустимы, записи размером 4 байта должны быть выровнены до границы слова.

Поскольку для каждой команды записи с проверкой RMAP всегда выполняется только одна операция AHB, инкрементирующий бит адреса может быть установлен в любое состояние.

Ограничений по записям без проверки нет, если инкрементирующий бит установлен. Если данный бит сброшен, количество байтов должно быть кратно 4, при этом адрес должен быть выровнен до границы слова. Точное количество записанных слов при обнаружении преждевременного EOP/EEP для операций записи без предварительной проверки данных не определено.

Команды считывания

Команды считывания выполняются в процессе работы во время отправки ответа. Таким образом, при возникновении ошибки на шине AHB пакет будет усечен и завершен символом EEP. Ограничений по инкрементируемому чтению нет; неинкрементируемое чтение имеет такие же ограничения по выравниванию, как записи без проверки. Следует обратить внимание на то, что код ошибки «Отказ в авторизации» отправляется в ответе при обнаружении какого-либо нарушения, даже если поле размера сброшено. Также следует обратить внимание на то, что данные не отправляются в ответе при обнаружении какой-либо ошибки, т.е. если поле статуса отлично от нуля.

Команды считывание-модификация-запись

Поддерживаются все размеры режима считывание-модификация-запись, за исключением 6 байтовых, которые вызывают выполнение трех байтовых операций считывания и записи на шине. Доступ по шине операции считывания-модификации-записи имеет те же ограничения, что и запись с предварительной проверкой данных. Как и в случае с записью с проверкой, бит инкремента может быть установлен в любое состояние, т.к. для каждой команды считывания-модификации-записи выполняется только одна операция на шине AHB. Детектирование слишком большого объёма данных выполняется после доступа по шине, таким образом, данная ошибка не будет препятствовать выполнению операции. Данные в ответе не будут отправлены при обнаружении ошибки, т.е. если поле статуса отлично от нуля.

Управление

Обработчик команд RMAP в основном работает в фоновом режиме без внешнего вмешательства. Однако существуют несколько опций, позволяющих им управлять.

В регистре управления имеется бит разрешения (RE), который используется для полного отключения обработчика команд RMAP. Если данный бит сброшен, то аппаратная обработка пакетов RMAP не осуществляется, вместо этого все они сохраняются через канал DMA.

Команды RMAP могут выполняться не в порядке их получения, если запрос на считывание данных приходит перед одной или несколькими командами записи. Поскольку обработчик команд сохраняет ответы в буфер, разделенный на части, несколько команд может находиться в процессе обработки, в то время как ещё не оправлен ни один ответ. Данные для ответов на запросы считывания извлекаются из памяти при отправке ответа. Таким образом, записи, полученные после запросов на считывание, могут быть уже выполнены в случае перегрузки передатчика. Чтобы гарантировать отсутствие подобной ситуации, необходимо установить бит запрета буферной схемы RMAP (RD) регистра управления, чтобы обработчик команд использовал только один буфер.

Еще одной опцией управления обработчиком команд RMAP является возможность установки ключа получателя, под который отведен отдельный регистр.

В таблице 4.17.9 представлены варианты обработки различных типов пакета и полей команд RMAP аппаратными средствами GRSPW2.

Таблица 4.17.9 – Обработка различных типов пакета и полей команд RMAP аппаратными средствами GRSPW2

Биты						Команда	Действие
7	6	5	4	3	2		
Зарезервировано	Команда/Ответ	Запись/Считывание	Проверка данных перед записью	Подтверждение	Инкрементация адреса		
1	2	3	4	5	6	7	8
0	0	–	–	–	–	Ответ	Сохранение через канал DMA
0	1	0	0	0	0	Не используется	Действие отсутствует. Без ответа
0	1	0	0	0	1	Не используется	Действие отсутствует. Без ответа
0	1	0	0	1	0	Считывание одиночного адреса	Обычное выполнение. Адрес выравнивается до границы слова, размер данных должен быть кратным 4 байтам. Отправляется ответ. При нарушении ограничений по выравниванию устанавливается код ошибки 10
0	1	0	0	1	1	Считывание, адрес инкрементируется	Обычное выполнение. Ограничений нет. Отправляется ответ
0	1	0	1	0	0	Не используется	Действие отсутствует. Без ответа
0	1	0	1	0	1	Не используется	Действие отсутствует. Без ответа
0	1	0	1	1	0	Не используется	Действие отсутствует. Отправляется ответ с кодом ошибки 2
0	1	0	1	1	1	Считывание-модификация-запись, адрес инкрементируется	Обычное выполнение. Если размер не соответствует допустимым значениям считывания-модификации-записи, то действие отсутствует, устанавливается код ошибки 11. Если соответствует, то проверяются ограничения по выравниванию. 1 байт может быть считан-модифицирован-записан по любому адрес. 2 байта должны быть выровнены до границы полуслова. 3 байта не допустимы. 4 байта – выравнивание до границы слова. При нарушении данных ограничений действие отсутствует, устанавливается код ошибки 10. При возникновении ошибки на шине AHB устанавливается код ошибки 1. Отправляется ответ

Продолжение таблицы 4.17.9

1	2	3	4	5	6	7	8
0	1	1	0	0	0	Запись, одиночный адрес, без проверки до записи, без подтверждения	Обычное выполнение. Адрес выравнивается до слова, размер данных должен быть кратным 4. При нарушении выравнивания действие отсутствует. Без ответа
0	1	1	0	0	1	Запись, адрес инкрементируется, без проверки данных перед записью, без подтверждения	Обычное выполнение. Ограничений нет. Без ответа
0	1	1	0	1	0	Запись, одиночный адрес, без проверки данных перед записью, с подтверждением	Обычное выполнение. Адрес выравнивается до границы слова, размер данных должен быть кратным 4. При нарушении выравнивания действие отсутствует, устанавливается код ошибки 10. При возникновении ошибки на шине AHB устанавливается код ошибки 1. Отправляется ответ
0	1	1	0	1	1	Запись, адрес инкрементируется, без проверки данных перед записью, с подтверждением	Обычное выполнение. Ограничений нет. При возникновении ошибки на шине AHB устанавливается код ошибки 1. Отправляется ответ
0	1	1	1	0	0	Запись, одиночный адрес, с проверкой данных перед записью, без подтверждения	Обычное выполнение. Размер до 4 байт включительно. В противном случае, действие отсутствует. Ограничения по выравниванию такие же, как для команды считывание-модификация-запись. Без ответа
0	1	1	1	0	1	Запись, адрес инкрементируется, с проверкой данных перед записью, без подтверждения	Обычное выполнение. Размер до 4 байт включительно. В противном случае, действие отсутствует. Ограничения по выравниванию такие же, как для команды считывание-модификация-запись. При нарушении ограничений действие отсутствует. Без ответа
0	1	1	1	1	0	Запись, одиночный адрес, с проверкой данных перед записью, с подтверждением	Обычное выполнение. Размер до 4 байт включительно. В противном случае, действие отсутствует, устанавливается код ошибки 9. Ограничения по выравниванию такие же, как для команды считывание-модификация-запись. При нарушении ограничений действие отсутствует, устанавливается код ошибки 10. При возникновении ошибки на шине AHB устанавливается код ошибки 1. Отправляется ответ

Окончание таблицы 4.17.9

1	2	3	4	5	6	7	8
0	1	1	1	1	1	Запись, адрес инкрементируется, с проверкой данных перед записью, с подтверждением	Обычное выполнение. Размер до 4 байт включительно. В противном случае, действие отсутствует, устанавливается код ошибки 9. Ограничения по выравниванию такие же, как для команды считывание-модификация-запись. При нарушении ограничений действие отсутствует, устанавливается код ошибки 10. При возникновении ошибки на шине AHB устанавливается код ошибки 1. Отправляется ответ
1	0	–	–	–	–	Не используется	Сохранение через канал DMA
1	1	–	–	–	–	Не используется	Сохранение через канал DMA

4.17.8 Интерфейс AMBA

Интерфейс AMBA состоит из интерфейса APB, интерфейса ведущего устройства AHB и FIFO DMA. Интерфейс APB обеспечивает доступ к пользовательским регистрам – подробнее в 4.17.10. Серверы DMA имеют 32-разрядные FIFO, соединенные с интерфейсом ведущего устройства AHB и используемые при считывании и записи по шине.

Сервер DMA передатчика считывает данные с шины по частям, размер которых в два раза меньше размера FIFO. Запуск разбиения на части всегда выполняется при наполовину заполненном FIFO или если FIFO содержит последние данные пакета. Размер блока, содержащего последние данные, может быть меньше, если пакет имеет нечетное количество частей.

DMA приемника работает аналогично, за исключением того, что устройство проверяет содержимое FIFO (FIFO должно быть наполовину заполнено) и после этого выполняет запись пакета с разбиением на части, размер которых в два раза меньше размера FIFO. Последний пакет может быть меньше по размеру. Интерфейс также обрабатывает байтовый доступ. Байтовые доступы используются для буферов без выравнивания до слова и/или если размер пакета не кратен четырем байтам. При записи начала и конца принятых пакетов может быть от одной до трех одиночных байтовых записей.

Интерфейс ведомого устройства APB

Как указано выше, интерфейс APB обеспечивает доступ к пользовательским регистрам размером 32 бита. Размеры доступов к данному интерфейсу должны быть выровнены до слова. При нарушении данного ограничения результат не определён.

Интерфейс ведущего устройства AHB

Система содержит один интерфейс ведущего устройства, который используется механизмами DMA передатчика и приемника. Арбитражный алгоритм между каналами выполняется так, что если текущий пользователь отправляет повторный запрос, интерфейс всегда предоставляется. Это не приведет к проблемам «голодания», так как механизмы DMA всегда снимают свои запросы между доступами.

Доступы AHB могут иметь размер байта, полуслова и слова (HSIZE = 0x000, 0x001, 0x010). Байтовый доступ и доступ полуслова всегда NONSEQ. Однако нужно помнить,

что доступ чтения всегда имеет размер слова (HSIZE = 0x010), что может привести к разрушающему считыванию данных.

Размер пакета в два раза меньше размера FIFO AHB, за исключением последней передачи пакета, который может быть меньше. Меньшие доступы при считывании указателя и записи статуса также выполняются.

Ведущее устройство AHB также поддерживает неинкрементный доступ, адрес которого неизменен для нескольких последовательных доступов. HTRANS в этом случае всегда NONSEQ, в то время как для инкрементного доступа он будет SEQ после первого доступа. Данная опция включена для поддержки неинкрементного чтения и записи для RMAP.

Если система не использует шину после завершения передачи пакетов, то будет присутствовать один «пустой» цикл (HTRANS = IDLE).

Запросы на передачу с типом BUSY не подаются. Поддерживаются ответы ERROR, RETRY и SPLIT.

4.17.9 Аппаратные особенности

Особенности системы тактирования

Модуль приёмника, представленный на рисунке 4.17.2, тактируется от блока GRSPW2_PHY. Внутри данного блока захват данных производится по обоим фронтам тактового сигнала (DDR). В качестве выходных сигналов GRSPW2_PHY выступают двухразрядные векторы данных и корректности данных. Для корректного функционирования удвоенная частота выборки должна быть, по крайней мере, в 1,5 раза больше максимальной скорости передачи бит.

Тактовая частота передатчика генерируется на основе сигнала SPW_CLK с использованием делителя. Отдельный тактовый вход используется для того, чтобы можно было вести передачу на частотах, значительно превышающих системную. Частота передатчика должна составлять 10 МГц при установлении соединения, в рабочем режиме может быть использовано любое значение, превышающее 2 МГц.

Регистр тактового делителя содержит два поля. Одно из них определяет коэффициент деления в процессе инициализации соединения, второе задает тактовую частоту в рабочем режиме. Значение по сбросу для обоих полей – 4, что соответствует делению входной тактовой частоты в 5 раз. Подробная инструкция по выбору значений коэффициентов делителя находится в подпункте «Установка скорости канала связи» пункта «4.17.3 Интерфейс связи».

Поскольку для коэффициентов делителей частоты доступны только целочисленные значения и частота при инициализации должна быть 10 МГц, на тактовый вход передатчика должен поступать сигнал с частотой, кратной 10 МГц. Под хранение значения коэффициентов выделено 8 разрядов. Также важно помнить, что существует ограничение на соотношение тактовой частоты системы и частоты передатчика SpaceWire.

Таймеры

В устройстве реализовано два таймера: первый для генерации 6,4/12,8 мкс периодов, второй для отслеживания времени до рассоединения.

Тайм-аут интервалы формируются на основе тактирования передатчика, частота которого должна быть, по крайней мере, 10 МГц для обеспечения соблюдения временных интервалов до рассоединения. Используется тот же самый коэффициент делителя, что и в момент инициализации, поэтому для нормального функционирования необходимо определить его корректно.

Синхронизация

Скорости передачи бит приёмника и передатчика могут быть в восемь раз выше, чем основная тактовая частота процессора. Такое соотношение приводится с учетом неустойчивости фазы и искажения сигналов, поэтому соединение может запуститься даже и при большей разнице частот. Однако следует принимать во внимание тот факт, что между частотой выборки и скоростью передачи бит при приёме нет прямой зависимости.

Требования по синхронизации тактирования не единственный фактор, который может препятствовать достижению предельных значений частот, могут присутствовать технологические ограничения.

Конфигурация буферов FIFO

FIFO AHB приемника состоит из 16 записей разрядностью в 32 бита. Организация используемой памяти 16×32 бит.

FIFO AHB передатчика состоит из 16 записей разрядностью в 32 бита. Организация используемой памяти 16×32 бит.

FIFO N-символов приемника состоит из 16 записей разрядностью в 9 бит. Организация используемой памяти 16×9 бит.

Используется 2 буфера RMAP, каждый из которых состоит из 32 записей с разрядностью 8 бит. Организация используемой памяти 64×8 бит.

4.17.10 Регистры

Программирование осуществляется через регистры, отображаемые в адресуемом пространстве APB. Список используемых регистров представлен в таблице 4.17.10, детальное описание каждого из них представлено в таблицах 4.17.14 – 4.17.24. Адреса, не представленные в таблице 4.17.10, являются зарезервированными. Чтение из зарезервированной области или отдельного поля регистра всегда возвращает нулевое значение, запись в них игнорируется.

Таблица 4.17.10 – Регистры GRSPW2 (базовые адреса 0x80100500, 0x80100600, 0x80100700 и 0x80100800)

Смещение адреса APB	Регистр
0x0	Регистр управления
0x4	Регистр статуса
0x8	Регистр адреса по умолчанию
0xC	Регистр тактового делителя
0x10	Регистр ключа получателя RMAP
0x14	Регистр системного времени (тайм-кодов)
0x18	Зарезервировано
0x20	Регистр управления/статуса канала DMA (канал 1)
0x24	Регистр максимального размера принимаемого пакета (канал 1)
0x28	Регистр адреса таблицы дескрипторов передатчика (канал 1)
0x2C	Регистр адреса таблицы дескрипторов приёмника (канал 1)
0x30	Регистр адреса канала DMA (канал 1)
0x34 – 0x3C	Зарезервировано
0x40 – 0x5C	Регистры канала 2 DMA (Зарезервировано)
0x60 – 0x7C	Регистры канала 3 DMA (Зарезервировано)
0x80 – 0x9C	Регистры канала 4 DMA (Зарезервировано)

Система записи регистров управления представлена в таблице 4.17.11, в таблицах 4.17.12 и 4.17.13 приведены списки возможных состояний ячеек.

Таблица 4.17.11 – Пример организации записи регистра

31 24	23 16	15 8	7 0
П3	П2	П1	П0
Значение по сбросу П3	Значение по сбросу П2	Значение по сбросу П1	Значение по сбросу П0
Тип поля/байта П3	Тип поля/байта П2	Тип поля/байта П1	Тип поля/байта П0

31–24 Поле 3 (П3) – Описание поля/бита
23–16 Поле 2 (П2) – Описание поля/бита
15–8 Поле 1 (П1) – Описание поля/бита
7–0 Поле 0 (П0) – Описание поля/бита

Таблица 4.17.12 – Список возможных состояний ячеек значения по сбросу

Значение	Описание
0	Значение по сбросу – 0. Используется для отдельных бит
1	Значение по сбросу – 1. Используется для отдельных бит
0xNN	Шестнадцатеричное представление числа. Используется для полей
n/r	Не сбрасывается
*	Особые условия сброса, которые разъяснены в описании соответствующего поля/бита

Таблица 4.17.13 – Список возможных состояний ячеек тип поля/бита

Значение	Описание
r	Только чтение
rw	Чтение/запись
wc	Сброс по записи. Поле доступно для считывания, при записи 1 оно сбрасывается. Запись 0 игнорируется

Регистр управления

Таблица 4.17.14 – Регистр управления GRSPW2

31	30	29	28 27	26	25	24	23	22	21	20	19 18	17	16
RA	RX	RC	NHC	PO	–	ID	–	LE	PS	NP	PNPA	RD	RE
1	0	1	0	0	0	0	0	0	0	0	0x0	0	0*
r	r	r	r	r	r	r	r	rw	r	r	r	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PE	–	TL	TF	TR	TT	LI	TQ	–	RS	PM	TI	IE	AS	LS	LD
0	0	0	0	0	0	0	0	0	0	0	0	0	0*	0	0
r	r	rw	rw	rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw

31 RMAP доступен (RA) – Устанавливается, если доступен обработчик команд RMAP
30 Приемнику доступны невыровненные обращения к памяти (RX) – Устанавливается, если доступны невыровненные записи для приемника и он может писать любое количество байт по любому стартовому адресу без записи избыточных байт. Функционал не поддерживается
29 CRC RMAP доступен (RC) – Устанавливается, если разрешен CRC RMAP
28, 27 Количество каналов DMA (NCH) – Поле хранит число доступных каналов DMA минус один. Только для считывания

26	Количество портов (PO) – Количество доступных портов SpaceWire минус один
25	Зарезервировано
24	Распространение прерываний доступно (ID) – Устанавливается, если доступно распространение прерываний. Функционал не поддерживается
23	Зарезервировано
22	Разрешение режима обратной петли (LE) – Функционал не используется.
21	Выбор порта (PS) – Выбор активного порта, если бит принудительного запрета порта сброшен. «0» выбирает порт, соединенный с данными и стробом с индексом 0. «1» выбирает индекс 1. Функционал доступен только для двухпортовых модулей
20	Принудительный запрет порта (NP) – Не разрешает выбор порта. Если бит установлен, то бит выбора порта не может быть использован для выбора активного порта. Вместо этого он будет автоматически выбран при проверке активности по соответствующим каналам приема. Функционал доступен только для двухпортовых модулей
19, 18	SpaceWire Plug & Play доступен (PNPA) – 0x0 – не поддерживается, 0x1 – поддерживается идентификация устройства, 0x2 – поддерживается идентификация и конфигурация устройства. Функционал не поддерживается
17	Запрет буферной схемы RMAP (RD) – Если бит установлен, используется только один буфер RMAP. Это гарантирует последовательное выполнение всех команд RMAP
16	RMAP разрешен (RE) – Разрешает сортировщик команд RMAP. Значение после сброса: «0» (устанавливается входным сигналом RMAPEN)
15	SpaceWire Plug & Play разрешен (PE) – Разрешение использования Plug & Play. Функционал не поддерживается
14	Зарезервировано
13	Контроль блокировки разрешения передатчика (TL) – Разрешение/запрет на использование функционала регистра управления/статуса DMA по блокировке разрешения передатчика. 0 – запрещено, 1 – разрешено
12	Фильтр флагов контроля тайм-кодов (TF) – Если бит установлен, то достоверными считаются только тайм-коды с контрольными флагами «00». Если бит сброшен, то разрешены любые значения контрольных флагов
11	Прием тайм-кодов разрешен (TR) – Разрешает прием кодов синхронизации
10	Передача тайм-кодов разрешена (TT) – Разрешает передачу кодов синхронизации
9	Прерывание при ошибке связи (LI) – Разрешение/запрет на генерацию прерывания на шине AMBA при возникновении ошибки связи. Важно помнить, что бит IE также должен быть установлен, чтобы значение LI имело какое-либо влияние
8	Прерывание при получении тайм-кода (TQ) – Разрешение/запрет на генерацию прерывания при приеме достоверного кода синхронизации (тайм-кода). Важно помнить, что бит IE также должен быть установлен, чтобы значение TQ имело какое-либо влияние
7	Зарезервировано
6	Сброс (RS) – Полный сброс узла SpaceWire. С самоочисткой
5	«Неразборчивый» режим (PM) – Разрешает «неразборчивый» режим (см. «4.17.5 Приёмный канал DMA» подпункт ««Неразборчивый» режим»)

4 Входной отсчет (TI) – Хост-система генерирует входной отсчет при записи «1» в данное поле. Инкрементируется счетчик синхронизации, и передается новое значение после передачи текущего символа. Входной отсчет также генерируется при установке сигнала tick_in

3 Разрешение прерывания (IE) – Если установлено, то разрешена генерация прерываний по шине AMBA для событий, которые индивидуально маскируются битами LI и TQ данного регистра

2 Автостарт (AS) – Автоматический старт связи при приеме NULL. Значение после сброса: «0» (устанавливается входным сигналом RMAPEN)

1 Старт соединения (LS) – Старт связи, т.е. разрешение передачи, перевод из режима готовности в режим запуска

0 Запрет соединения (LD) – Кодек SpaceWire не разрешен

Регистр статуса

Таблица 4.17.15 – Регистр статуса GRSPW2

31 28	27 26	25 24	23 21	20 10	9	8	7	6 5	4	3	2	1	0
–	NRXD	NTXD	LS	–	AP	EE	IA	–	PE	DE	ER	CE	TO
0x0	0x0	0x0	0x0	0x000	0	0	0	0x0	0	0	0	0	0
r	r	r	r	r	r	wc	wc	r	wc	wc	wc	wc	wc

31–28 Зарезервировано

27, 26 Количество дескрипторов приёмника (NRXD) – Показывает размер таблицы дескрипторов механизма DMA приёмника. «00» = 128, «01» = 256, «10» = 512, «11» = 1024

25, 24 Количество дескрипторов передатчика (NTXD) – Показывает размер таблицы дескрипторов механизма DMA передатчика. «00» = 64, «01» = 128, «10» = 256, «11» = 512

23–21 Состояние соединения (LS) – Текущее состояние последовательности запуска. 0 = сброс после сбоя (ErrorReset), 1 = ожидание после сбоя (ErrorWait), 2 = готово (Ready), 3 = запущено (Started), 4 = соединение (Connecting), 5 = рабочий режим (Run)

20–10 Зарезервировано

9 Активный порт (AP) – Показывает активный порт. «0» = Порт 0; «1» = Порт 1, где номер порта ссылается на номер индекса сигналов данных и строба. Функционал доступен только для двухпортовых модулей

8 Преждевременный EOP/EEP (EE) – Устанавливается, если пакет с EOP принят после первого байта для non-RMAP пакета и после второго байта для пакета RMAP. Сбрасывается при записи 1. Значение после сброса: «0»

7 Недопустимый адрес (IA) – Устанавливается, если пакет принят с недействительным полем адреса назначения, т.е. в случае несоответствия содержимому регистра адреса по умолчанию

6–5 Зарезервировано

4 Ошибка паритета (PE) – Возникла ошибка по четности. Сбрасывается при записи 1

3 Ошибка рассоединения (DE) – Возникла ошибка при разрыве соединения. Сбрасывается при записи 1

2 Ошибка ESC-символа (ER) – Возникла ошибка ESC-символа. Сбрасывается при записи 1

1 Ошибка кредитования (CE) – Возникла ошибка при разрешении на передачу. Сбрасывается при записи 1

0 Выходной отсчет (TO) – Принято новое значение счетчика времени, сохранено в поле счетчика времени. Сбрасывается при записи 1

Регистр адреса по умолчанию

Таблица 4.17.16 – Регистр адреса по умолчанию GRSPW2

31 16	15 8	7 0
–	DEFMASK	DEFADDR
0x0000	0x00	0xFE
r	rw	rw

31–16 Зарезервировано

15–8 Маска по умолчанию (DEFMASK) – Используется для идентификации узла в сети SpaceWire. Оба поля адреса – принятый адрес и DEFADDR логически умножаются с инверсией поля DEFMASK перед проверкой адреса

7–0 Адрес узла по умолчанию (DEFADDR) – 8-битовый адрес узла, используемый для идентификации узла в сети SpaceWire

Регистр тактового делителя

Таблица 4.17.17 – Регистр тактового делителя GRSPW2

31 16	15 8	7 0
–	CLKDIVSTART	CLKDIVRUN
0x0000	0x04	0x04
r	rw	rw

31–16 Зарезервировано

15–8 Значение тактового делителя при старте (CLKDIVSTART) – значение, используемое для тактового делителя при запуске линии связи (любое из состояний соединения, кроме рабочего режима). Подробная информация по расчету требуемого значения в пункте «4.17.3». Фактическое значение делителя: значение регистра тактового делителя + 1. Значение после сброса: 4 (устанавливается входным сигналом CLKDIV10)

7–0 Значение тактового делителя в рабочем режиме (CLKDIVRUN) – значение, используемое для тактового делителя, когда интерфейс связи находится в рабочем режиме. Фактическое значение делителя: значение регистра тактового делителя + 1. Значение после сброса: 4 (устанавливается входным сигналом CLKDIV10)

Регистр ключа получателя RMAP

Таблица 4.17.18 – Регистр ключа получателя RMAP GRSPW2

31 8	7 0
–	DESTKEY
0x000000	0x00
r	rw

31–8 Зарезервировано

7–0 Ключ получателя (DESTKEY) – Ключ целевого RMAP

Регистр системного времени (тайм-кодов)

Таблица 4.17.19 – Регистр системного времени (тайм-кодов) GRSPW2

31 8	7 6	5 0
–	TCTRL	TIMECNT
0x000000	0x0	0x00
r	rw	rw

31–8 Зарезервировано

7, 6 Флаги контроля тайм-кодов (TCTRL) – Текущее значение флагов контроля системного времени (тайм-кодов). Они отсылаются в составе тайм-кода при установке бита TI регистра управления. Также данное поле обновляется при каждом получении тайм-кода

5–0 Счетчик времени (TIMECNT) – Текущее значение счетчика системного времени. Значение инкрементируется и передается в составе тайм-кода при установке бита TI регистра управления. Также данное поле обновляется при каждом получении тайм-кода. Важно понимать, что при непосредственной записи в данный регистр выставленное значение не будет передано, поскольку перед передачей оно будет инкрементировано

Регистр управления/статуса канала DMA

Таблица 4.17.20 – Регистр управления/статуса канала DMA GRSPW2

31 27	26 24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTNUM	–	EP	TR	IE	IT	RP	TP	TL	LE	SP	SA	EN	NS	RD	RX	AT	RA	TA	PR	PS	AI	RI	TI	RE	TE
0x00	0x0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	wc	wc	rw	rw	wc	wc	wc	rw	rw	rw	rw	rw	rw	r	rw	wc	wc	wc	wc	rw	rw	rw	rw	rw

31–27 Количество прерываний (INTNUM) – Количество прерываний, которые используются данным каналом DMA для отправки кодов-прерываний, маскируемых битами IE и IT данного регистра. Функционал не поддерживается

26–24 Зарезервировано

23 EEP окончание (EP) – Устанавливается, когда завершающим символом полученного пакета является EEP символ (пакет получен с ошибкой). Сбрасывается при записи 1

22 Усеченный (TR) – Устанавливается, когда полученный пакет канала DMA сохраняется не целиком из-за превышения установленной максимальной длины. Сбрасывается при записи 1

21 Разрешение передачи кодов-прерывания при EEP (IE) – Если бит установлен, то при получении пакета с завершающим символом EEP будет сгенерирован код-прерывания (также требуется глобальное разрешение на передачу кодов-прерывания). Функционал не поддерживается

20 Разрешение передачи кодов-прерывания при усечении пакета (IT) – Если бит установлен, то при получении усечённого пакета будет сгенерирован код-прерывания (также требуется глобальное разрешение на передачу кодов-прерывания). Функционал не поддерживается

19 Прерывание при получении пакета (RP) – Бит устанавливается, когда на шине AMBA генерируется прерывание, вызванное получением пакета по соответствующему каналу DMA

18 Прерывание при отправке пакета (TP) – Бит устанавливается, когда на шине AMBA генерируется прерывание, вызванное отправкой пакета по соответствующему каналу DMA

17 Блокировка разрешения передатчика (TL) – Бит устанавливается, если бит TL Регистр управления установлен, и передатчик соответствующего канала DMA запрещён из-за ошибки соединения (обеспечивается битом LE данного регистра). Пока данный бит установлен, невозможно перезапустить передатчик (т.е. нельзя установить бит TE данного регистра)

16 Запрещение работы при ошибке соединения (LE) – Запрет на дальнейшую передачу при возникновении ошибки соединения. До повторного установления бита разрешения передатчика, пакеты передаваться не будут

15 Отделять PID (SP) – Удаление PID-байта (второго байта) каждого пакета. Если данный бит установлен, то адресный байт (первый байт) будет также удален, вне зависимости от значения бита SA данного регистра.

14 Отделять адрес (SA) – Удаление адресного байта (первого байта) каждого пакета

13 Разрешение адреса (EN) – Разрешение отдельного адреса узла для этого канала

12 Без вытеснения (NS) – Если бит сброшен, то получаемые пакеты будут отбракованы при отсутствии активированного дескриптора. Если бит установлен, то GRSPW2 будет ожидать активации дескриптора

11 Доступен дескриптор приёмника (RD) – Устанавливается для указания GRSPW2 того, что в таблице дескрипторов имеются активированные дескрипторы. Сбрасывается GRSPW2 в случае обнаружения неактивированного дескриптора

10 Осуществляется приём (RX) – Устанавливается, если в текущий момент осуществляется приём по данному каналу DMA; в противном случае, бит сброшен

9 Прервать передачу (AT) – Установка данного бита прервет текущую передачу пакета и снимет разрешение на передачу. Если на текущий момент передача не производится, то произойдёт только снятие разрешения на передачу. Самоочистка

8 Ошибка AHB при приёме (RA) – Шина AHB отреагировала ошибкой на попытку доступа при приёме через данный канал DMA. Сбрасывается при записи 1

7 Ошибка AHB при передаче (TA) – Шина AHB отреагировала ошибкой на попытку доступа при передаче через данный канал DMA. Сбрасывается при записи 1

6 Пакет принят (PR) – Этот бит устанавливается каждый раз, когда пакет получен. Сбрасывается при записи «1»

5 Пакет отправлен (PS) – Этот бит устанавливается каждый раз, когда пакет отправлен. Сбрасывается при записи «1»

4 Прерывание из-за ошибки AHB (AI) – Если установлено, то при каждом возникновении ошибки на шине AHB из-за попытки доступа по данному каналу DMA генерируется прерывание

3 Прерывание при приеме (RI) – Если установлено, то при наличии установленного бита разрешения прерывания (IE) соответствующего дескриптора при каждом приеме пакета генерируется прерывание. Это происходит вне зависимости от того заканчивается пакет EEP или EOP

2 Прерывание при передаче (TI) – Если установлено, то при наличии установленного бита разрешения прерывания (IE) соответствующего дескриптора при каждой передаче пакета генерируется прерывание. Это происходит вне зависимости от того, успешно прошла передача или нет

1 Разрешение приёма (RE) – Установка данного бита разрешает прием пакетов по данному каналу

0 Разрешение передачи (TE) – Разрешает передачу для соответствующего канала DMA. Запись «1» вызывает процедуру считывания узлом SpaceWire нового дескриптора и попытку отправки пакета адресату. Важно помнить, что бит TE можно установить, только если бит TL данного регистра сброшен. Бит TE автоматически сбрасывается при обнаружении узлом SpaceWire неактивированного дескриптора или если во время передачи пакета произошла ошибка соединения (устанавливается бит LE данного регистра)

Регистр максимального размера принимаемого пакета

Таблица 4.17.21 – Регистр максимального размера принимаемого пакета GRSPW2

31 25	24 2	1 0
–	RXMAXLEN	–
0x00	n/r	0x0
r	rw	r

31–25 Зарезервировано
24–2 Максимальный размер принимаемого пакета (RXMAXLEN) – Максимальный размер пакета приёмника в байтах. Без сброса
1, 0 Зарезервировано

Регистр адреса таблицы дескрипторов передатчика

Таблица 4.17.22 – Регистр адреса таблицы дескрипторов передатчика GRSPW2

31 10*	9* 4	3 0
DESCBASEADDR	DESCSEL	–
n/r	0x00	0x0
rw	rw	r

31–10* Базовый адрес таблицы дескрипторов (DESCBASEADDR) – Устанавливает базовый адрес таблицы дескрипторов. Правая граница поля рассчитывается исходя из формулы (10 + значение поля NTXD регистра статуса). Без сброса
9*–4 Селектор дескриптора (DESCSEL) – Смещение в таблице дескрипторов. Показывает текущий используемый узлом SpaceWire дескриптор. Для чтения каждого нового дескриптора селектор увеличивается на 16 (в расчет принимается и зарезервированное поле 3–0) и конечном итоге обнуляется. Левая граница поля рассчитывается исходя из формулы (9 + значение поля NTXD регистра статуса)
3–0 Зарезервировано

Регистр адреса таблицы дескрипторов приёмника

Таблица 4.17.23 – Регистр адреса таблицы дескрипторов приёмника GRSPW2

31 10*	9* 3	2 0
DESCBASEADDR	DESCSEL	–
n/r	0x00	0x0
rw	rw	r

31–10* Базовый адрес таблицы дескрипторов (DESCBASEADDR) – Устанавливает базовый адрес таблицы дескрипторов. Правая граница поля рассчитывается исходя из формулы (10 + значение поля NRXD регистра статуса). Без сброса

9*–3 Селектор дескриптора (DESCSEL) – Смещение в таблице дескрипторов. Показывает текущий используемый узлом SpaceWire дескриптор. Для чтения каждого нового дескриптора селектор увеличивается на 8 (в расчет принимается и зарезервированное поле 2–0) и в конечном итоге обнуляется. Левая граница поля рассчитывается, исходя из формулы (9 + значение поля NRXD регистра статуса)

2–0 Зарезервировано

Регистр адреса канала DMA

Таблица 4.17.24 – Регистр адреса канала DMA GRSPW2

31 16	15 8	7 0
–	MASK	ADDR
0x0000	n/r	n/r
r	rw	rw

31–16 Зарезервировано

15–8 Маска (MASK) – Используется для идентификации узла в сети SpaceWire. Оба поля адреса – принятый адрес и ADDR логически умножаются с инверсией поля MASK перед проверкой адреса

7–0 Адрес (ADDR) – Если установлен бит EN регистра управления/статуса канала DMA, то данный адрес используется для идентификации узла в сети SpaceWire для данного канала DMA

4.17.11 Интерфейс прикладного программирования (API)

Интерфейс прикладного программирования (API – Application Programming Interface) поставляется вместе с GRSPW2. API расположен в директории $(GRLIB)/software/spw. Это файлы rmapapi.c, spwapi.c, rmapapi.h, spwapi.h. В spwapi.h содержатся объявления функций, используемых для настройки GRSPW2 и передачи данных. Соответствующие определения содержатся в spwapi.c. Файл rmapapi структурирован таким же образом и содержит функции для построения пакетов RMAP.

Эти функции описаны в этом пункте и могут быть использованы в виде простой отправной точки для разработки драйверов GRSPW2.

Базовый API GRSPW2

Базовый API GRSPW2 основан на структуре spwvars (см. таблицы 4.17.25, 4.17.26), которая хранит всю информацию для одного модуля GRSPW2. Эта информация включает его адрес на шине AMBA, а также параметры SpaceWire, такие как адрес узла и значения тактового делителя. Указатель на эту структуру используется в качестве входного параметра для всех функций. Если используется несколько ядер, для каждого модуля GRSPW2 создается и используется отдельная структура, с помощью которой осуществляется доступ к конкретному модулю.

Таблица 4.17.25 – Структура spwvars

Поле	Описание	Диапазон значений
regs	Указатель на структуру, соответствующую набору регистров GRSPW2	–
rmap	Указывает: сконфигурировано ли ядро с RMAP. Значение считывается в результате выполнения spw_init	0 – 1
rxunaligned	Указывает: сконфигурировано ли ядро с поддержкой не выровненных записей для приёмника. Значение считывается в результате выполнения spw_init	0 – 1
rmapcrc	Указывает: сконфигурировано ли ядро с поддержкой RMAP CRC. Значение считывается в результате выполнения spw_init	0 – 1
timetxen	Разрешение на передачу тайм-кодов данным модулем	0 – 1
timerxen	Разрешение на приём тайм-кодов данным модулем	0 – 1
ver	Версия GRSPW. Определяется автоматически в результате выполнения spw_init	–
khz	Значение тактовой частоты процессора. Не используется в GRSPW2	–
dmachan	Количество каналов DMA в данном модуле GRSPW2. Значение устанавливается в результате выполнения spw_init («1» – единственное возможное значение, поскольку реализован только один канал)	1
clkdiv	Значение тактового делителя, используемое модулем в рабочем режиме	0 – 255
clkdivs	Значение тактового делителя для получения требуемой для установления соединения частоты (10 Мбит/с)	0 – 255
timer	Не используется	–
dc	Не используется	–
nodeaddr	Адрес узла по умолчанию и его маска (формат записи соответствует полям регистра адреса по умолчанию), используемый для идентификации в сети SpiceWire	0 – 65535
mask	Не используется	–
destkey	Значение ключа получателя RMAP, используемое ядром	0 – 255
port	Выбор активного порта. Функционал недоступен, поскольку реализован однопортовый модуль GRSPW2	0 – 1
dma	Структура DMA	–

Таблица 4.17.26 – Структура DMA

Поле	Описание	Диапазон значений
nospill	Значение бита без вытеснения (NS) регистра управления/статуса канала DMA	0 – 1
rxmaxlen	Максимальное значение размера буфера приемника канала DMA, минимальный шаг изменения размера – 4 байта (подробное описание см. подпункт «Подготовка устройства для приёма» пункта «4.17.5 Приёмный канал DMA.)	0 – 33554432
rxpnt	Порядковый номер текущего дескриптора приёмника	0 – 127
rxchkpnt	Порядковый номер проверяемого дескриптора приёмника	0 – 127
txpnt	Порядковый номер текущего дескриптора передатчика	0 – 63
txchkpnt	Порядковый номер проверяемого дескриптора передатчика	0 – 63
addr	Адрес узла, используемый для идентификации канала DMA в сети SpaceWire, когда установлен бит EN регистра управления/статуса канала DMA. Если бит EN сброшен, используется значение адреса по умолчанию (поле «nodeaddr» структуры «spwvars»)	0 – 255
mask	Маска адреса узла, используемая для идентификации канала DMA в сети SpaceWire, когда установлен бит EN в регистре управления DMA. Если бит EN сброшен, используется значение маски адреса по умолчанию (поле «mask» структуры «spwvars»)	0 – 255
txd	Указатель на структуру txdescriptor. Подробное описание дескриптора передатчика находится в таблицах 4.17.4 – 4.17.7	–
rxd	Указатель на структуру rxdescriptor. Подробное описание дескриптора приёмника находится в таблицах 4.17.2 – 4.17.3	–

В базовом API доступны следующие функции:

int spw_setparam(int nodeaddr, int clkdiv, int destkey, int timetxen, int timerxen;
int spwadr, int khz, struct spwvars *spw, int port, int clkdivs).

Используется для установки различных параметров экземпляра структуры spwvars. Всегда должна быть выполнена в первую очередь после создания данного экземпляра. Эта функция только инициализирует структуру, она не записывает никаких данных в регистры модуля SpaceWire, доступные по шине APB (см. таблицы 4.17.27, 4.17.28).

Таблица 4.17.27 – Возвращаемые значения функции

Значение	Описание
0	Функция завершена успешно
1	Один или несколько параметров имели некорректное значение

Таблица 4.17.28 – Параметры функции spw_setparam

Параметр	Описание	Допустимый диапазон
nodeaddr	Устанавливает значение адреса по умолчанию узла SpaceWire в структуре spw	0 – 255
clkdiv	Устанавливает значение тактового делителя в рабочем режиме в структуре spw	0 – 255
destkey	Устанавливает значение ключа файла-приемника в структуре spw	0 – 255
timetxen	Устанавливает разрешение на передачу тайм-кодов данным модулем структуры spw	0 – 1
timerxen	Устанавливает разрешение на приём тайм-кодов данным модулем структуры spw	0 – 1
spwadr	Адрес регистров управления на шине APB модуля GRSPW2, который будет передан в структуру spw (инициализирует указатель на структуру spwregs)	$0 – 2^{32}–1$
khz	Значение поля должно соответствовать тактовой частоте процессора. Не используется в GRSPW2	–
spw	Указатель на экземпляр структуры spwvars	–
port	Устанавливает активный порт, если бит принудительного запрета порта сброшен. Не используется, поскольку реализован однопортовый модуль GRSPW2	0 – 1
clkdivs	Устанавливает значение тактового делителя для получения требуемой частоты модуля (10 Мбит/с) для процесса установления соединения в сети SpaceWire	0 – 255

spw_setparam_dma(int dmachan, int addr, int mask, int nospill, int rxmaxlen, struct spwvars *spw).

Используется для установки параметров заданного канала DMA в структуре spwvars. Реализован только один канал DMA, поэтому в качестве первого параметра всегда должен передаваться ноль (соответствует выбору канала 1 DMA), см. таблицы 4.17.29, 4.17.30.

Таблица 4.17.29 – Возвращаемые значения функции

Значение	Описание
0	Функция завершена успешно
1	Один или несколько параметров имели некорректное значение

Таблица 4.17.30 – Параметры функции spw_setparam_dma

Параметр	Описание	Допустимый диапазон
dmachan	Номер канала DMA (реализован один канал)	0
addr	Устанавливает значение поля addr структуры dma для заданного канала	0 – 255
mask	Устанавливает значение поля mask структуры dma для заданного канала	0 – 255
nospill	Устанавливает значение поля nospill структуры dma для заданного канала	0 – 1
rxmaxlen	Устанавливает значение поля rxmaxlen структуры dma для заданного канала	0 – 33554432
spw	Указатель на экземпляр структуры spwvars	–

int spw_init(struct spwvars *spw).

Инициализирует модуль GRSPW2, расположенный по адресу, указанному в spw. Список устанавливаемых регистров: регистр адреса по умолчанию, регистр ключа получателя RMAP, регистр тактового делителя, регистр максимального размера принимаемого пакета, регистр адреса таблицы дескрипторов передатчика, регистр адреса таблицы дескрипторов приёмника, регистр управления и регистр управления/статуса канала DMA. Все биты устанавливаются в соответствии со значениями структуры spwvars. Если бита регистра нет в структуре, то он будет установлен в 0. Под таблицы дескрипторов выделяются области памяти, выровненные по адресу под размер таблицы (с использованием функции malloc). Очищается регистр статуса и в заключение выполняется запуск соединения. Частота в рабочем режиме (Run) будет установлена в соответствии со значением поля clkdiv (см. таблицы 4.17.31, 4.17.32).

Таблица 4.17.31 – Возвращаемые значения функции spw_init

Значение	Описание
0	Функция завершена успешно
1	Один или несколько параметров были установлены некорректно или канал не удалось инициализировать

Таблица 4.17.32 – Параметры функции spw_init

Параметр	Описание	Допустимый диапазон
spw	Указатель на структуру spwvars, ассоциированную с модулем GRSPW2, который будет инициализирован	–

int wait_running(struct spwvars *spw).

Функция проверяет текущее состояние последовательности запуска канала SpaceWire. Если лимит попыток превышен, а модуль всё ещё не перешёл в рабочий режим, то функция завершается кодом ошибки (см. таблицу 4.17.33).

Таблица 4.17.33 – Возвращаемые значения функции wait_running

Значение	Описание
0	Функция завершена успешно
1	Модуль SpaceWire не перешел в рабочий режим (в состояние Run)

int set_txdesc(int dmachan, int pnt, struct spwvars *spw).

Устанавливает новый адрес в регистре адреса таблицы дескрипторов передатчика. Данную функцию можно использовать, только если передача неактивна. Также сбрасываются порядковые номера текущего и проверяемого дескрипторов, используемые в функциях spw_tx и spw_checktx. Возвращаемые значения и параметры для spw_txdesc представлены в таблицах 4.17.34, 4.17.35.

Таблица 4.17.34 – Возвращаемые значения функции set_txdesc

Значение	Описание
0	Функция выполнена успешно
1	Новый адрес записан некорректно

Таблица 4.17.35 – Параметры функции spw_txdesc

Параметр	Описание	Допустимый диапазон
dmachan	Устанавливает номер канала DMA (реализован один канал)	0
pnt	Новый адрес области таблицы дескрипторов	$0 – 2^{32}–1$
spw	Указатель на структуру spwvars, ассоциируемую с целевым модулем GRSPW2	–

int set_rxdesc(int dmachan, int pnt, struct spwvars *spw).

Устанавливает новый адрес в регистре адреса таблицы указателя приемника. Данную функцию можно использовать, если только передача неактивна. Также сбрасываются порядковые номера текущего и проверяемого дескрипторов, используемые в функциях spw_rx и spw_checkrx. Возвращаемые значения и параметры для spw_rxdesc представлены в таблицах 4.17.36 и 4.17.37.

Таблица 4.17.36 – Возвращаемые значения для spw_rxdesc

Значение	Описание
0	Функция выполнена успешно
1	Новый адрес записан некорректно

Таблица 4.17.37 – Параметры функции spw_rxdesc

Параметр	Описание	Допустимый диапазон
dmachan	Номер канала DMA (реализован один канал)	0
pnt	Новый адрес области таблицы дескрипторов	$0 – 2^{32}–1$
spw	Указатель на структуру spwvars, ассоциируемую с целевым модулем GRSPW2	–

void spw_disable(struct spwvars *spw).

Запрет на использование модуля GRSPW2, функция устанавливает бит запрета соединения (LD) регистра управления (см. таблицу 4.17.38).

Таблица 4.17.38 – Параметры функции spw_disable

Параметр	Описание	Допустимый диапазон
spw	Указатель на структуру spwvars, ассоциируемую с целевым модулем GRSPW2	–

void spw_enable(struct spwvars *spw).

Разрешение на использование модуля GRSPW2, функция сбрасывает бит запрета соединения (LD) регистра управления (см. таблицу 4.17.39).

Таблица 4.17.39 – Параметры функции spw_enable

Параметр	Описание	Допустимый диапазон
spw	Указатель на структуру spwvars, ассоциируемую с целевым модулем GRSPW2	–

void spw_start(struct spwvars *spw).

Запуск соединения GRSPW2, функция устанавливает бит старт соединения (LS) регистра управления (см. таблицу 4.17.40).

Таблица 4.17.40 – Параметры для spw_start

Параметр	Описание	Допустимый диапазон
spw	Указатель на структуру spwvars, ассоциируемую с целевым модулем GRSPW2	–

void spw_stop(struct spwvars *spw).

Останавливает GRSPW2, функция сбрасывает бит старт соединения (LS) регистра управления (см. таблицу 4.17.41).

Таблица 4.17.41 – Параметры функции spw_stop

Параметр	Описание	Допустимый диапазон
spw	Указатель на структуру spwvars, ассоциируемую с целевым модулем GRSPW2	–

int spw_setclockdiv(struct spwvars *spw).

Устанавливает регистр тактового делителя, значения полей берутся из структуры spwvars. Возвращаемые значения и параметры для spw_setclockdiv представлены в таблицах 4.17.42, 4.17.43.

Таблица 4.17.42 – Возвращаемые значения функции spw_setclockdiv

Значение	Описание
0	Функция выполнена успешно
1	Новое значение тактового делителя недопустимо

Таблица 4.17.43 – Параметры функции spw_setclockdiv

Параметр	Описание	Допустимый диапазон
spw	Указатель на структуру spwvars, ассоциируемую с целевым модулем GRSPW2	–

int spw_set_nodeadr(struct spwvars *spw).

Устанавливает регистр адреса по умолчанию при этом значения полей адреса и маски берутся из структуры spwvars. Возвращаемые значения и параметры для spw_set_nodeadr представлены в таблицах 4.17.44, 4.17.45.

Таблица 4.17.44– Возвращаемые значения функции spw_set_nodeadr

Значение	Описание
0	Функция выполнена успешно
1	Новое значение адреса/маски узла недопустимо

Таблица 4.17.45 – Параметры функции spw_set_nodeadr

Параметр	Описание	Допустимый диапазон
spw	Указатель на структуру spwvars, ассоциируемую с целевым модулем GRSPW2	–

int spw_set_chanadr(int dmachan, struct spwvars *spw).

Устанавливает новое значение отдельного адреса канала DMA. Реализован только один канал DMA, поэтому в качестве первого параметра всегда должен передаваться 0 (см. таблицы 4.17.46, 4.17.47).

Таблица 4.17.46– Возвращаемые значения функции spw_set_chanadr

Значение	Описание
0	Функция выполнена успешно
1	Новое значение отдельного адреса канала DMA недопустимо

Таблица 4.17.47 – Параметры функции spw_set_chanadr

Параметр	Описание	Допустимый диапазон
dmachan	Номер канала DMA (реализован один канал)	0
spw	Указатель на структуру spwvars, ассоциируемую с целевым модулем GRSPW	–

int spw_set_rxmaxlength(int dmachan, struct spwvars *spw).

Устанавливает регистр максимального размера принимаемого пакета, при этом значение rxmaxlen берётся из структуры spwvars. Возвращаемые значения и параметры для spw_set_rxmaxlength представлены в таблицах 4.17.48, 4.17.49.

Таблица 4.17.48 – Возвращаемые значения функции spw_set_rxmaxlength

Значение	Описание
0	Функция выполнена успешно
1	Новое значение максимального размера принимаемого пакета недопустимо

Таблица 4.17.49 – Параметры функции spw_set_rxmaxlength

Параметр	Описание	Допустимый диапазон
dmachan	Номер канала DMA (реализован один канал)	0
spw	Указатель на структуру spwvars, ассоциируемую с целевым модулем GRSPW2	–

int spw_tx(int dmachan, int hcrc, int dcrc, int skipcrcsize, int hsize,
char *hbuf, int dsize, char *dbuf, struct spwvars *spw).

Передает пакет. Могут использоваться отдельные буферы заголовка и данных. GSPW2 вставляет контрольные суммы CRC, рассчитанные согласно спецификации RMAP, после полей данных и/или заголовка, если dcrc и/или hcrc установлены в единицу. Данная функция только устанавливает дескриптор и инициирует передачу. Для того чтобы проверить, был ли передан пакет, следует использовать функцию spw_checktx. Указатель на запись таблицы дескрипторов хранится в структуре spwvars, он используется для отслеживания следующей позиции в таблице. Инкрементируется каждый раз, когда функция возвращает «0». Возвращаемые значения и параметры для spw_tx представлены в таблицах 4.17.50, 4.17.51.

Таблица 4.17.50 – Возвращаемые значения функции spw_tx

Значение	Описание
0	Функция выполнена успешно
1	Не установлен бит разрешения (EN) в текущем дескрипторе передающего канала DMA
2	Недопустимое значение количества байт в начале пакета, которые будут исключены из расчета CRC
3	Некорректно заданы параметры hcrc или dcrc
4	Не определен один или оба указателя на буферы заголовка и данных
5	Недопустимый размер заголовка в байтах
6	Недопустимый размер поля данных в байтах

Таблица 4.17.51 – Параметры функции spw_tx

Параметр	Описание	Допустимый диапазон
dmachan	Номер канала DMA (реализован один канал)	0
hcrc	Добавить CRC RMAP после поля заголовка	0 – 1
dcrc	Добавить CRC RMAP после поля данных	0 – 1
skipcrcsize	Определяет количество байт в начале пакета, которые будут исключены из расчета CRC	0 – 15
hsize	Размер заголовка в байтах	0 – 255
hbuf	Указатель на данные заголовка	–
dsize	Размер поля данных в байтах	$0 – 2^{24}–1$
dbuf	Указатель на область данных	–
spw	Указатель на структуру spwvars, ассоциированную с целевым модулем GRSPW2, который будет осуществлять передачу пакета	–

int spw_rx(int dmachan, char *buf, struct spwvars *spw).

Устанавливает разрешение на исполнение дескриптора приёмника. Пакет будет сохранен в буфере «buf». Для проверки корректности приёма пакета должна быть использована функция spw_checkrx. Указатель «rxpnt» во вложенной структуре dma структуры spwvars используется для отслеживания следующей позиции в таблице дескрипторов. Он автоматически инкрементируется каждый раз, когда функция возвращает значение «0». Возвращаемые значения и параметры для spw_rx представлены в таблицах 4.17.52 и 4.17.53.

Таблица 4.17.52 – Возвращаемые значения функции spw_rx

Значение	Описание
0	Функция выполнена успешно
1	Свободные указатели для приема отсутствуют

Таблица 4.17.53 – Параметры функции spw_rx

Параметр	Описание	Допустимый диапазон
dmachan	Номер канала DMA (реализован один канал)	0
buf	Указатель на область данных	–
spw	Указатель на структуру spwvars, ассоциированную с целевым модулем GRSPW2, который будет осуществлять приём пакета	–

int spw_checkrx(int dmachan, int *size, struct rxstatus *rxs, struct spwvars *spw).

Проверяет, был ли осуществлен прием пакета, возвращаемые значения функции отображены в таблице 4.17.54. После приема пакета размер в байтах сохраняется в соответствии с параметром размера; статус – в структуре rxs. Указатель «rxchkpnt» во вложенной структуре dma структуры spwvars используется для отслеживания положения дескриптора в таблице дескрипторов, который должен быть опрошен. Он автоматически инкрементируется каждый раз, когда функция возвращает ненулевое значение. Параметры для spw_checkrx представлены в таблицах 4.17.55 и 4.17.56.

Таблица 4.17.54 – Возвращаемые значения функции spw_checkrx

Значение	Описание
0	Пакет не был принят
1	Пакет был принят

Таблица 4.17.55 – Параметры функции spw_checkrx

Параметр	Описание	Допустимый диапазон
dmachan	Номер канала DMA (реализован один канал)	0
size	Если пакет был принят, то переменная содержит количество принятых байт	–
rxs	Если пакет был принят, то в данную структуру сохраняется информация о статусе пакета	–
spw	Указатель на структуру spwvars, ассоциированную с целевым модулем GRSPW2, который должен быть опрошен	–

Таблица 4.17.56 – Структура rxstatus

Поле	Описание	Допустимый диапазон
truncated	Пакет был усечён	0 – 1
dcrcerr	Установлен бит ошибки CRC данных. Указывает на состояние ошибки в случае, если принятый пакет – пакет RMAP	0 – 1
hcrcerr	Установлен бит ошибки CRC заголовка. Указывает на состояние ошибки в случае, если принятый пакет – пакет RMAP	0 – 1
eep	Пакет завершен с EEP	0 – 1

int spw_checktx(int dmachan, struct spwvars *spw).

Проверяет была ли осуществлена передача пакета. Указатель «txchkpnt» во вложенной структуре dma структуры spwvars используется для отслеживания положения дескриптора в таблице дескрипторов, который должен быть опрошен. Он автоматически инкрементируется каждый раз, когда функция возвращает ненулевое значение. Возвращаемые значения и параметры для spw_checktx представлены в таблицах 4.17.57 и 4.17.58.

Таблица 4.17.57 – Возвращаемые значения функции spw_checktx

Значение	Описание
0	Пакет не был передан
1	Пакет был передан корректно
2	Пакет был передан некорректно

Таблица 4.17.58 – Параметры функции spw_checktx

Параметр	Описание	Допустимый диапазон
dmachan	Номер канала DMA (реализован один канал)	0
spw	Указатель на структуру spwvars, ассоциированную с целевым модулем GRSPW2	–

void send_time(struct spwvars *spw).

Послать новый тайм-код. Увеличивает счетчик системного времени в GRSPW2 и передает данное значение в сеть SpaceWire. Параметры для send_time представлены в таблице 4.17.59.

Таблица 4.17.59 – Параметры для времени отправки

Параметр	Описание	Допустимый диапазон
spw	Указатель на структуру spwvars, ассоциированную с целевым модулем GRSPW2	–

int send_time_exp(int ctrl, int time, struct spwvars *spw).

Послать заданный тайм-код. Флаги контроля и предварительно декрементированное (по модулю 64) значение счетчика времени записываются в регистр системного времени (тайм-кодов), затем выполняются действия, аналогичные send_time. Возвращаемые значения и параметры функции send_time_exp представлены в таблицах 4.17.60, 4.17.61.

Таблица 4.17.60 – Параметры функции send_time_exp

Параметр	Описание	Допустимый диапазон
ctrl	Значение флагов контроля тайм-кодов	
time	Посылаемое значение счетчика системного времени	0 – 63
spw	Указатель на структуру spwvars, ассоциированную с целевым модулем GRSPW2	–

Таблица 4.17.61 – Возвращаемые значения функции send_time_exp

Значение	Описание
0	Тайм-код был передан
–1	Передаваемое значение функции не попадает в разрешенный диапазон

int check_time(struct spwvars *spw).

Проверяет был ли получен тайм-код. Функция возвращает значение бита входного отсчета (TO) регистра статуса, после чего сбрасывает данный бит. Возвращаемые значения и параметры для check_time представлены в таблицах 4.17.62, 4.17.63.

Таблица 4.17.62 – Возвращаемые значения функции check_time

Значение	Описание
0	Код синхронизации не принят
1	Принято новое значение счетчика времени

Таблица 4.17.63 – Параметры функции check_time

Параметр	Описание	Допустимый диапазон
spw	Указатель на структуру spwvars, ассоциированную с модулем GRSPW2, который должен быть опрошен	–

int get_time(struct spwvars *spw).

Возвращает текущее значение счетчика времени (см. таблицу 4.17.64). Параметры для get_time представлены в таблице 4.17.65.

Таблица 4.17.64 – Возвращаемые значения функции get_time

Значение	Описание
0 – 63	Текущее значение счетчика времени

Таблица 4.17.65 – Параметры функции get_time

Параметр	Описание	Допустимый диапазон
spw	Указатель на структуру spwvars, ассоциированную с модулем GRSPW2, который должен быть опрошен	–

int get_time_ctrl(struct spwvars *spw).

Возвращает текущее значение флагов контроля тайм-кода, записанных в регистр системного времени (тайм-кодов) GRSPW2. Возвращаемые значения и параметры для check_time представлены в таблицах 4.17.66, 4.17.67.

Таблица 4.17.66 – Возвращаемые значения функции get_time_ctrl

Значение	Описание
0 – 3	Возвращает текущее значение флагов контроля тайм-кодов

Таблица 4.17.67 – Параметры функции get_time_ctrl

Параметр	Описание	Допустимый диапазон
spw	Указатель на структуру spwvars, ассоциированную с модулем GRSPW2, который должен быть опрошен	–

void spw_reset(struct spwvars *spw).

Сбрасывает GRSPW2. Параметры для spw_reset представлены в таблице 4.17.68.

Таблица 4.17.68 – Параметры функции spw_reset

Параметр	Описание	Допустимый диапазон
spw	Указатель на структуру spwvars, ассоциированную с модулем GRSPW2, который должен быть сброшен	–

void spw_rmapen(struct spwvars *spw).

Разрешение на использование аппаратного сортировщика команд RMAP. Параметры для spw_rmapen представлены в таблице 4.17.69.

Таблица 4.17.69 – Параметры функции spw_rmapen

Параметр	Описание	Допустимый диапазон
spw	Указатель на структуру spwvars, ассоциированную с целевым модулем GRSPW2	–

void spw_rmapdis(struct spwvars *spw).

Запрет на использование аппаратного сортировщика команд RMAP. Параметры для spw_rmapdis представлены в таблице 4.17.70.

Таблица 4.17.70 – Параметры функции spw_rmapdis

Параметр	Описание	Допустимый диапазон
spw	Указатель на структуру spwvars, ассоциированную с целевым модулем GRSPW2	–

int spw_setdestkey(struct spwvars *spw).

Устанавливает ключ получателя RMAP GRSPW2. Возвращаемые значения и параметры для spw_setdestkey представлены в таблицах 4.17.71 и 4.17.72.

Таблица 4.17.71 – Возвращаемые значения функции spw_setdestkey

Значение	Описание
0	Функция выполнена успешно
1	Некорректное значение параметра ключа получателя в структуре spwvars

Таблица 4.17.72 – Параметры функции spw_setdestkey

Параметр	Описание	Допустимый диапазон
spw	Указатель на структуру spwvars, ассоциированную с целевым модулем GRSPW2	–

void spw_setsepaddr(int dmachan, struct spwvars *spw).

Разрешает использование отдельного адреса узла для данного канала DMA (см. таблицу 4.17.73).

Таблица 4.17.73 – Параметры функции spw_setsepaddr

Параметр	Описание	Допустимый диапазон
dmachan	Номер канала DMA (реализован один канал)	0
spw	Указатель на структуру spwvars, ассоциированную с целевым модулем GRSPW2	–

void spw_disablesepaddr(int dmachan, struct spwvars *spw).

Запрещает использование отдельного адреса узла для данного канала DMA (см. таблицу 4.17.74).

Таблица 4.17.74 – Параметры функции spw_disablesepaddr

Параметр	Описание	Допустимый диапазон
dmachan	Номер канала DMA (реализован один канал)	0
spw	Указатель на структуру spwvars, ассоциированную с целевым модулем GRSPW2	–

void spw_enablerx(int dmachan, struct spwvars *spw).

Разрешить прием пакетов по данному каналу DMA (см. таблицу 4.17.75).

Таблица 4.17.75 – Параметры функции spw_enablerx

Параметр	Описание	Допустимый диапазон
dmachan	Номер канала DMA (реализован один канал)	0
spw	Указатель на структуру spwvars, ассоциированную с целевым модулем GRSPW2	–

void spw_disablerx(int dmachan, struct spwvars *spw).

Запретить прием пакетов по заданному каналу DMA (см. таблицу 4.17.76).

Таблица 4.17.76 – Параметры функции spw_disablerx

Параметр	Описание	Допустимый диапазон
dmachan	Номер канала DMA (реализован один канал)	0
spw	Указатель на структуру spwvars, ассоциированную с целевым модулем GRSPW2	–

void spw_disable_promiscuous(struct spwvars *spw).

Отключение «неразборчивого» режима (см. таблицу 4.17.77).

Таблица 4.17.77 – Параметры функции spw_disable_promiscuous

Параметр	Описание	Допустимый диапазон
spw	Указатель на структуру spwvars, ассоциированную с целевым модулем GRSPW2	–

void spw_enable_promiscuous(struct spwvars *spw).

Включение «неразборчивого» режима (см. таблицу 4.17.78).

Таблица 4.17.78 – Параметры функции spw_enable_promiscuous

Параметр	Описание	Допустимый диапазон
spw	Указатель на структуру spwvars, ассоциированную с целевым модулем GRSPW2	–

API RMAP GRSPW2

API RMAP содержит только одну функцию, используемую для создания заголовков RMAP.

int build_rmap_hdr(struct rmap_pkt *pkt, char *hdr, int *size).

Создает заголовки RMAP в буфере, указанном в hdr. Возвращаемые значения и параметры для build_rmap_hdr представлены в таблицах 4.17.79 – 4.17.81. Данные заголовка берутся из структуры rmap_pkt.

Таблица 4.17.79 – Возвращаемые значения для build_rmap_hdr

Значение	Описание
0	Функция выполнена успешно
1	Значения type или verify структуры rmap_pkt не попадают в разрешенный диапазон
2	Значение ack структуры rmap_pkt не попадает в разрешенный диапазон
3	Значение incr структуры rmap_pkt не попадает в разрешенный диапазон
4	Значение dstspalen структуры rmap_pkt не попадает в разрешенный диапазон
5	Значение srcspalen структуры rmap_pkt не попадает в разрешенный диапазон
6	Значение destkey структуры rmap_pkt не попадает в разрешенный диапазон
7	Значение destaddr структуры rmap_pkt не попадает в разрешенный диапазон
9	Значение srcaddr структуры rmap_pkt не попадает в разрешенный диапазон
10	Значение tid структуры rmap_pkt не попадает в разрешенный диапазон
11	Значение len структуры rmap_pkt не попадает в разрешенный диапазон
12	Значение status структуры rmap_pkt не попадает в разрешенный диапазон

Таблица 4.17.80 – Параметры для build_rmap_hdr

Параметр	Описание	Допустимый диапазон
pkt	Указатель на структуру rmap_pkt, которая содержит данные для создания заголовка	–
hdr	Указатель на буфер, в котором заголовок будет создан	–
size	После создания заголовка переменная будет содержать его размер	–

Таблица 4.17.81 – Поля структуры rmap_pkt

Поле	Описание	Допустимый диапазон
1	2	3
type	Выбор типа создаваемого пакета	writecmd, readcmd, rmwcmd, writerep, readrep, rmwrep
verify	Осуществлять ли проверку данных перед записью	yes, no
ack	Отправлять ли подтверждение	yes, no
incr	Должен ли адрес быть инкрементирован	yes, no
destaddr	Устанавливает адрес получателя	0 – 255
destkey	Устанавливает ключ получателя	0 – 255
srcaddr	Устанавливает адрес источника	0 – 255
tid	Устанавливает поле идентификатора передачи	0 – 65 535
addr	Устанавливает адрес выполняемой операции. Поле расширенного адреса всегда сброшено	$0 – 2^{32}–1$
len	Количество байт для записи, считывания или операции считывания-модификации-записи	$0 – 2^{24}–1$
status	Устанавливает поле статуса	0 – 11

Окончание таблицы 4.17.81

1	2	3
dstspalen	Количество байт адреса пути от отправителя, которые будут вставлены перед адресом получателя	0 – 228
dstspa	Указатель на область памяти, в которой хранятся байты адреса пути от отправителя	–
srcspalen	Количество байт адреса пути от отправителя, которые будут вставлены в команду. В ответе эти байты вставляются перед адресом возврата	0 – 12
srcspa	Указатель на область памяти, в которой хранятся байты адреса пути от отправителя	–

4.18 Интерфейс Ethernet с поддержкой EDCL

4.18.1 Общие сведения

Контроллер доступа к среде в сетях Ethernet от Cobham Gaisler (GRETH) обеспечивает интерфейс между шиной AHB AMBA и сетью Ethernet, см. рисунок 4.18.1. Он поддерживает скорость 10/100 Мбит/с в полнодуплексном и полудуплексном режимах. Интерфейс AMBA состоит из интерфейса APB, обеспечивающего конфигурацию и управление интерфейса ведущего устройства AHB, обрабатывающего поток данных. Поток данных обрабатывается через каналы DMA. И передатчики и приемник имеют по одному механизму DMA. Оба этих механизма используют общий интерфейс ведущего устройства AHB. Для подключения к внешнему PHY используется независимый от среды интерфейс (MII). GRETH также предоставляет служебный канал (MI) интерфейса MII, который используется для конфигурации PHY.

Также обеспечена поддержка протокола линии связи отладки Ethernet (EDCL). Для удаленной отладки используется протокол, основанный на UDP/IP.

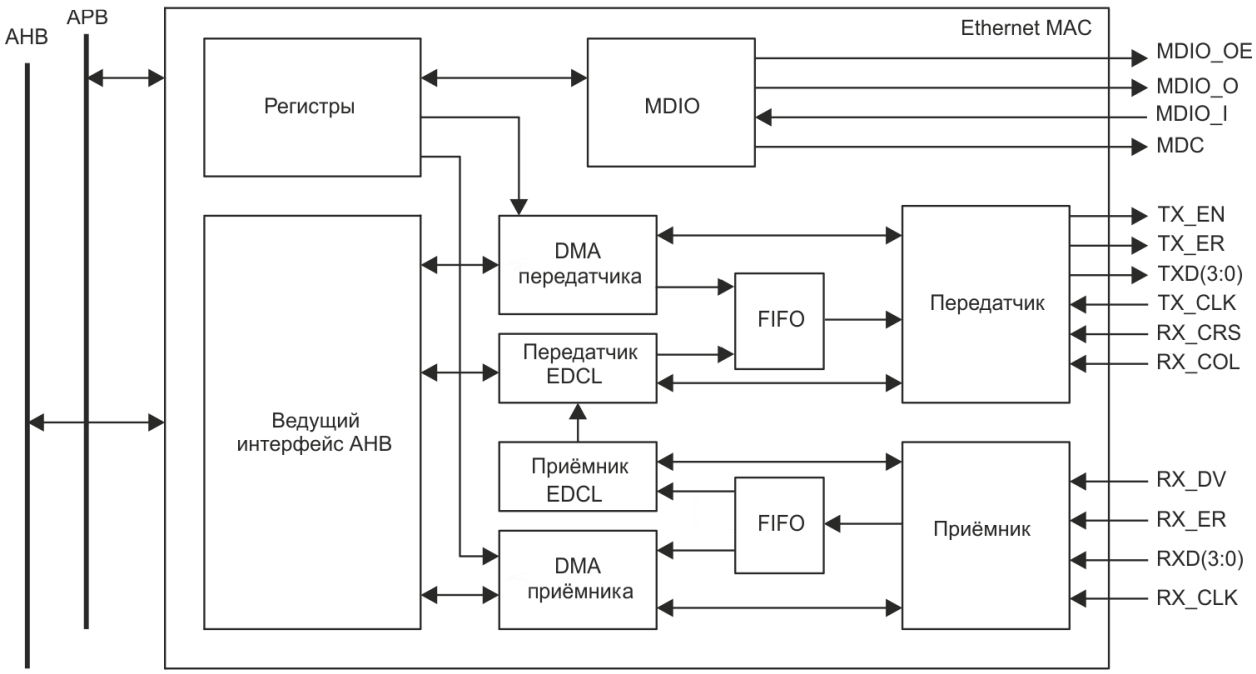


Рисунок 4.18.1 – Блок-схема внутренней структуры GRETH

4.18.2 Принцип работы

Общие сведения о системе

GRETH состоит из трех функциональных блоков: каналов DMA, интерфейса MDIO и линии связи отладки Ethernet (EDCL).

Основой функционирования контроллера являются каналы DMA, которые используются для передачи данных между шиной AHB и сетью Ethernet. Приемник и передатчик имеют по отдельному каналу DMA. Работа каналов DMA управляется через регистры, доступ к которым осуществляется через интерфейс APB.

Интерфейс MDIO (служебный канал) используется для получения доступа к регистрам конфигурации и статуса в одном или более PHY, присоединенных к контроллеру доступа к среде (Media Access Controller – MAC). Работа данного интерфейса также управляется через интерфейс APB.

EDCL обеспечивает доступ к операциям чтения и записи по шине AHB через Ethernet. EDCL использует протоколы UDP, IP, ARP вместе с протоколом уровня клиентского приложения. EDCL не содержит регистры с пользовательским доступом и всегда работает параллельно с каналами DMA.

Независимый от среды интерфейс (MII) используется для связи с PHY. Передатчик Ethernet пересылает все данные из домена AHB по сети Ethernet через интерфейс MII. Приемник Ethernet оправляет на хранение все данные из сети Ethernet по шине AHB. Оба эти интерфейса при передаче потоков данных используют FIFO.

Каналы DMA и EDCL используют общий приемник и передатчик Ethernet.

Поддержка протокола

GRETH реализован в соответствии со стандартом IEEE 802.3-2002. Дополнительный подуровень управления не поддерживается. Это означает, что пакеты программ с типом 0x8808 (только определяемые на данный момент пакеты программ управления) не учитываются.

Тактирование

GRETH имеет три домена тактирования: тактирование AHB, тактирование приемника Ethernet и тактирование передатчика Ethernet. Сигналы тактирования передатчика и приемника Ethernet генерируются внешним PHY Ethernet и подаются на вход контроллера через интерфейс MII. Эти три домена тактирования не связаны друг с другом; все сигналы, пересекающие области тактирования, полностью синхронизированы в устройстве.

Поддерживаются режимы работы 10/100 Мбит/с в дуплексном и полудуплексном режимах. Минимальное тактирование AHB для функционирования в 10 мегабитном режиме составляет 2,5 МГц, в 100 мегабитном – 25 МГц. Использование частот AHB ниже заявленных приведет к чрезмерной потере пакетов.

4.18.3 Интерфейс Tx DMA

Интерфейс DMA передатчика используется для передачи данных по сети Ethernet. Передача выполняется с помощью указателей, расположенных в памяти.

Установка дескриптора

Отдельно взятый дескриптор представлен в таблицах 4.18.1 и 4.18.2. Количество отправляемых байт должно быть установлено в поле размера; поле адреса должно

указывать на данные. Адрес должен быть выровнен по границе слова (4 байта). Если бит разрешения прерывания (IE) установлен, прерывание генерируется после отправки пакета (помимо этого требуется, чтобы в регистре управления также был установлен бит прерывания передатчика (TI)). Прерывание генерируется независимо от того, успешно ли передан пакет. Также до выполнения передачи должен быть установлен бит перехода на новую запись (WR) – см. ниже в данном пункте.

Таблица 4.18.1 – Слово «0» дескриптора передачи GRETH (смещение адреса 0x0)

31 16	15	14	13	12	11	10 0
–	AL	UE	IE	WR	EN	LENGTH

31–16	Зарезервировано
15	Ошибка предельного значения попыток (AL) – Пакет не передан, потому что достигнуто максимальное количество попыток
14	Ошибка недостаточного наполнения FIFO (UE) – Пакет неправильно передан из-за ошибки, связанной с недостаточным наполнением FIFO
13	Разрешение прерывания (IE) – Разрешает прерывания. Прерывание генерируется после отправки пакета с данным дескриптором при условии, что в регистре управления установлен бит разрешения прерывания передатчика. Прерывание генерируется независимо от того, успешно ли передан пакет или с ошибкой
12	Переход на новую запись (WR) – Установка бита обнуляет указатель дескриптора после того, как этот дескриптор был использован. Если этот бит сброшен, то указатель будет увеличен на 8. Указатель автоматически обнуляется при достижении граничного значения в 1 Кбайт таблицы дескрипторов
11	Разрешение (EN) – Установка этого бита приведет к активации дескриптора. Должен устанавливаться в последнюю очередь после всех полей дескриптора
10–0	LENGTH – Количество передаваемых байт

Таблица 4.18.2 – Слово «1» дескриптора передачи GRETH (смещение адреса 0x4)

31 2	1 0
ADDRESS	–

31–2	Адрес (ADDRESS) – Указатель на область буфера, из которой загружаются данные пакета
1, 0	Зарезервировано

Для активации дескриптора должен быть установлен бит разрешения (EN), после чего данный идентификатор не следует изменять до сброса бита разрешения, производимого контроллером GRETH.

Запуск передачи

Активации дескриптора недостаточно для запуска передачи. Сначала необходимо в GRETH поместить указатель на область памяти, содержащую дескрипторы. Это выполняется в регистре указателя дескриптора передатчика (см. «4.18.9 Регистры»). Адрес должен быть выровнен до 1 Кбайта. Биты 31–10 содержат базовый адрес области дескриптора, в то время, как биты (9–3) формируют указатель на отдельный дескриптор. Первый дескриптор должен размещаться в базовом адресе; после использования в GRETH поле указателя увеличивается на 8, чтобы указывать на следующий дескриптор. Указатель автоматически обнуляется при достижении граничного значения в

1 Кбайт (использование дескриптора со смещением адреса 0x3F8). Бит WR может быть установлен для перевода указателя на базовый адрес до достижения границы в 1 Кбайт.

Поле указателя для максимальной гибкости также может быть перезаписано, но при записи в регистр указателя дескриптора необходимо соблюдать осторожность. Он не должен изменяться пользователем в процессе передачи.

Последний шаг для активации передачи – установить бит разрешения передачи (TE) в регистре управления, при этом GRETH сообщается, что в таблице дескрипторов имеются другие активные дескрипторы. Этот бит должен быть всегда установлен при активации новых дескрипторов, даже если передача уже выполняется. Активация дескриптора всегда должна быть выполнена перед установкой бита разрешения передачи.

Обработка дескриптора после передачи

После окончания передачи пакета статус записывается в первое слово в соответствующем дескрипторе. Бит ошибки, связанной с неполной загрузкой (UE), устанавливается, если FIFO стало пустым до того, как пакет был полностью передан. Бит ошибки предельного значения попыток (AL) устанавливается, если количество конфликтных ситуаций превысило максимально допустимое значение. Пакет будет передан успешно, только если оба бита сброшены. Другие биты в первом слове дескриптора сбрасываются после завершения передачи, в то время как второе слово остается без изменений.

Бит разрешения должен использоваться как индикатор возможности использовать дескриптор снова, что происходит, когда он был сброшен контроллером GRETH. В регистре статуса GRETH имеется три бита, которые содержат статус передачи. Бит ошибки передатчика (TE) устанавливается, если передача завершена с ошибкой (когда, по крайней мере, установлен один из этих двух битов статуса в указателе передачи). Бит прерывания передатчика (TI) устанавливается, если передача завершена успешно.

Бит ошибки AHB передатчика (TA) устанавливается при обнаружении ошибки AHB при считывании дескриптора или пакетных данных. Любые активные передачи останавливаются, и передатчик отключается. Передача может быть вновь активирована установкой регистра разрешения передачи.

Установка данных для передачи

Данные для передачи должны быть размещены по адресу, на который указывает поле адреса дескриптора. GRETH не добавляет Ethernet-адрес и поле типа, поэтому они должны быть сохранены в буфере данных. 4-байтовый CRC Ethernet автоматически добавляется в конце каждого пакета. Каждый дескриптор соответствует отдельному пакету Ethernet. Если поле размера в дескрипторе больше 1514 байт, пакет не будет отправлен.

4.18.4 Интерфейс Rx DMA

Интерфейс DMA приемника используется для получения данных из сети Ethernet. Прием выполняется с помощью дескрипторов, расположенных в памяти.

Установка дескрипторов

Отдельно взятый дескриптор представлен в таблицах 4.18.3 и 4.18.4. Поле адреса должно указывать на буфер с выравниванием по границе слова (4 байта), в котором сохраняются полученные данные. GRETH сохраняет в буфере не более 1514 байт (соответственно максимальная длина поля данных принимаемого кадра

Ethernet II / IEEE 802.3 составляет 1500 байт). Если бит разрешения прерывания (IE) установлен, прерывание генерируется после того, как полученный пакет размещен в буфере (помимо этого требуется, чтобы в регистре управления также был установлен бит прерывания приемника (RI)). Прерывание генерируется независимо от того, успешно ли получен пакет. Также до активации дескриптора должен быть установлен бит перехода на новую запись (WR) – см. ниже в данном пункте.

Таблица 4.18.3 – Слово «0» дескриптора приема GRETH (смещение адреса 0x0)

31 27	26	25 19	18	17	16	15	14	13	12	11	10 0
–	MC	–	LE	OE	CE	FT	AE	IE	WR	EN	LENGTH

31–27	Зарезервировано
26	Групповая передача (MC) – Если данный бит установлен, то адрес назначения пакета является широковещательным (не передается). Данный функционал не поддерживается
25–19	Зарезервировано
18	Ошибка размера (LE) – Поле размера/типа пакета не соответствует фактическому количеству полученных байтов
17	Ошибка переполнения (OE) – Кадр получен некорректно в связи с переполнением FIFO
16	Ошибка CRC (CE) – В данном кадре обнаружена ошибка CRC
15	Слишком большой кадр (FT) – Получен кадр, превышающий максимально допустимый размер. Превышающая часть отсекается
14	Ошибка выравнивания (AE) – Получено нечетное количество полубайт
13	Разрешение прерывания (IE) – Разрешает прерывания. Прерывание генерируется после приема пакета с данным дескриптором при условии, что в регистр управления установлен бит разрешения прерывания приемника. Прерывание генерируется независимо от того, успешно ли принят пакет или с ошибкой
12	Переход на новую запись (WR) – Установка бита обнуляет указатель дескриптора после того, как этот дескриптор был использован. Если этот бит сброшен, то указатель будет увеличен на 8. Указатель автоматически обнуляется при достижении граничного значения в 1 Кбайт таблицы дескрипторов
11	Разрешение (EN) – Установка этого бита приведет к активации дескриптора. Должен устанавливаться в последнюю очередь после всех полей дескриптора
10–0	LENGTH – Количество принятых байт данного дескриптора

Таблица 4.18.4 – Слово «1» дескриптора приема GRETH (смещение адреса 0x4)

31 2	1 0
ADDRESS	–

31–2	Адрес (ADDRESS) – Указатель на область буфера, в которую загружаются данные пакета
1, 0	Зарезервировано

Запуск приема

Активации дескриптора недостаточно для запуска приема. Сначала необходимо в GRETH установить указатель на область памяти, содержащую дескрипторы. Это выполняется в регистре указателя дескриптора приемника (см. «4.18.9 Регистры»). Адрес должен быть выровнен до 1 Кбайта. Биты 31–10 содержат базовый адрес области

дескриптора, в то время как биты 9–3 формируют указатель на отдельный дескриптор. Первый дескриптор должен размещаться в базовом адресе; после использования в GRETH поле указателя увеличивается на 8, чтобы указывать на следующий дескриптор. Указатель автоматически обнуляется по достижении граничного значения в 1 Кбайт (использование дескриптора со смещением адреса 0x3F8). Бит WR в дескрипторах устанавливается для обнуления указателей по достижению границы в 1 Кбайт.

В поле указателя для максимальной гибкости также может быть перезапись, но при записи в регистр указателя дескриптора необходимо соблюдать осторожность. Он не должен изменяться пользователем в процессе приема.

Последний шаг для активации приема – установить бит разрешения приема (RE) в регистре управления. При этом GRETH считывает первый дескриптор и ожидает входящий пакет.

Обработка дескриптора после приема

GRETH указывает на завершение приема сбросом бита активации дескриптора (EN). Другие биты управления (WR, IE) также сбрасываются. Количество полученных байт указано в поле размера. Сохраняемые части Ethernet кадра – адрес назначения, исходный адрес, поля типа и данных. Биты 17–14 в первом слове указателя являются битами статуса, указывающими на различные ошибки приема. Все четыре бита оказываются сброшенными после приема без ошибок. Описание битов статуса приводится в таблице 4.18.3.

Получаемые пакеты, размер которых меньше минимального размера Ethernet в 64 байта, не рассматриваются как принятые и не учитываются. Текущий дескриптор приемника остается без изменений до первого приема пакета с размером соответствующим требованиям. Бит TS регистра статуса устанавливается при каждой регистрации подобного события.

Если пакет получен с адресом, не принятым MAC, устанавливается бит регистра статуса IA.

Если пакеты превышают максимальный размер, устанавливается бит FT в дескрипторе приема. Поле размера не всегда содержит корректное значение полученных байт. Подсчет прекращается после записи в память слова, содержащего последний байт, не превышающий максимального размера.

Адресное слово дескриптора контроллер GRETH оставляет в неизменном виде.

Прием с ошибками AHB

Если во время считывания дескриптора или сохранения данных происходит ошибка AHB, в регистре статуса устанавливается бит ошибки AHB приемника (RA), при этом приемник отключается. Текущий прием прерывается. Повторная активация приема может быть выполнена с помощью установки бита разрешения приема (RE) в регистре управления.

Прием MAC-адреса

Устройство принимает пакеты либо с индивидуальным адресом, установленным в регистрах MAC-адреса, либо с широковещательным адресом. Многоадресная рассылка, при которой используется хеш-функция для фильтрации полученных пакетов, не поддерживается.

4.18.5 Интерфейс MDIO

Интерфейс MDIO обеспечивает доступ к регистру конфигурации и статуса PHY через двухпроводной интерфейс, который входит в интерфейс MII. GRETH обеспечивает полную поддержку интерфейса MDIO.

Служебный канал (MI) содержит следующие управляющие сигналы:

Management Data Clock (EMDC) – тактовая частота для последовательного канала данных MDIO, которая вычисляется по формуле:

$$\frac{clk}{2 \times (mdcscaler + 1)},$$

где clk – тактовая частота процессора;
mdcscaler равно 50.

Management Data (EMDIO) – двунаправленный последовательный канал связи с PHY.

Interrupt Line (EMDINTN) – линия прерывания, активный уровень – «0».

Интерфейс MDIO позволяет адресовать до 32 независимых PHY, содержащих до 32-х 16-битовых регистров (7 обязательных по стандарту IEEE 802.3u и дополнительные, расширяющие функционал, если реализованы). Чтобы осуществить считывание регистра PHY, следует поместить в регистр управления MDIO адрес PHY, адрес требуемого регистра и установить бит «Чтение» (RD). При этом устанавливается бит занятости. При завершении операции этот бит автоматически сбрасывается. Если операция прошла успешно, бит «Ошибка соединения» (LF) будет находиться в состоянии сброса, при этом поле данных содержит считанные данные. Если операция прошла неудачно, то устанавливается бит «Ошибка соединения» (LF), при этом поле данных не определено.

Чтобы инициировать операцию записи, следует поместить в регистр статуса/управления MDIO 16 бит данных, адрес PHY, адрес требуемого регистра и установить бит «Запись» (WR). После завершения записи происходит автоматический сброс бита занятости. Если операция прошла успешно, бит «Ошибка соединения» (LF) будет находиться в состоянии сброса.

Прерывания PHY

Контроллер также поддерживает прерывания изменения статуса PHY. Чувствительный к уровню сигнал прерывания может быть подключен к входу прерывания интерфейса Ethernet (EMDINTN). Бит изменения статуса PHY в регистре статуса устанавливается при каждом обнаружении события этого сигнала. Если бит разрешения прерывания статуса PHY установлен во время события, контроллер генерирует прерывание на шине AHB.

4.18.6 Линия связи отладки Ethernet (EDCL)

EDCL обеспечивает доступ к встроенной шине AHB через Ethernet. EDCL использует протоколы UDP, IP и ARP вместе с протоколом уровня клиентского приложения. Протокол уровня клиентского приложения использует алгоритм автоматического запроса повторной передачи (ARQ) для обеспечения надежной передачи инструкций AHB. Через эту линию связи могут быть осуществлены операции чтения и записи по любому адресу на шине AHB.

Принцип работы

EDCL получает пакеты параллельно с приемником MAC канала DMA. EDCL использует отдельный MAC-адрес, который используется для различения пакетов EDCL и пакетов MAC, доставленных по каналу DMA. EDCL также имеет собственный IP-адрес. Так как пакеты ARP используют широковещательный адрес Ethernet, IP-адрес должен использоваться для отличия пакетов ARP EDCL и пакетов, проходящих через канал DMA. Пакеты EDCL не обрабатываются каналом DMA приема.

Если после проверки пакеты признаны корректными, то выполняется операция на шине AHB. Операция выполняется с помощью того же интерфейса ведущего устройства AHB, который используется механизмами DMA. Ответы автоматически отправляются передатчиком EDCL после завершения операции. Используется общий передатчик Ethernet и механизма DMA передатчика, при этом EDCL имеет более высокий приоритет.

Протоколы EDCL

EDCL принимает кадры Ethernet, содержащие IP или ARP пакеты данных. ARP-пакеты обрабатываются согласно спецификации протокола без исключений.

Пакеты IP переносят фактические команды AHB. EDCL требует кадр Ethernet, содержащий IP, UDP и части уровня специального приложения EDCL. В таблице 4.18.5 представлен пакет IP, требуемый EDCL. Содержание различных заголовков протокола – см. соответствующую литературу TCP/IP.

Таблица 4.18.5 – Пакет IP, требуемый EDCL

Заголовок Ethernet	Заголовок IP	Заголовок UDP	Смещение 2 байта	Слово управления 4 байта	Адрес 4 байта	Данные 0–242 4-байтовые слова	CRC Ethernet

Для успешного соединения с EDCL требуется следующее: корректный MAC-адрес назначения, установленный с помощью параметров настройки, поле типа Ethernet, содержащее 0x0806 (ARP) или 0x0800 (IP). IP-адрес сравнивается со значением, определенным параметрами настройки для обеспечения соответствия. Контрольная сумма заголовка IP и идентификационные поля не проверяются. Существует несколько ограничений для полей заголовка IP. Версия должна занимать четыре байта, размер заголовка должен быть 5 байт (без опций). Поле протокола должно всегда быть 0x11 с указанием пакета UDP. Размер и контрольная сумма – единственные поля IP с возможностью изменений для ответа.

EDCL предусматривает только один канал связи единовременно, поэтому проверять номер порта UDP не требуется. В ответе и в поле источника, и в поле назначения указывается исходный номер порта источника. Контрольная сумма UDP не используется, поле контрольной суммы в ответах обнулено.

Поле данных UDP содержит поля протокола приложения EDCL. В таблице 4.18.6 представлены поля протокола приложения (исключая поле данных) в пакетах, полученных EDCL.

Таблица 4.18.6 – Поля уровня приложения EDCL в кадрах приема

16 бит	14 бит	1 бит	10 бит	7 бит
Смещение	Порядковый номер	RD/WR – чтение/запись	Размер	Не используется

16-битовое смещение используется для выравнивания до слова остальных данных уровня приложения в памяти, может содержать любое значение. Поле RD/WR (чтение/записи) определяет, какая операция должна быть выполнена: чтение (0) или запись (1). Поле размера содержит количество байт для считывания или записи. Если

RD/WR установлено, поле данных, указанное в таблице 4.18.6, содержит данные для записи. Если RD/WR сброшено, поле данных не содержит полученных пакетов.

В таблице 4.18.7 представлены поля уровня приложения ответов EDCL. Поле размера для запросов записи в ответах всегда сброшено. Для запросов чтения оно содержит количество байт данных, содержащихся в поле данных.

Таблица 4.18.7 – Поля уровня приложения ответов EDCL в кадрах передачи

16 бит Смещение	14 бит Порядковый номер	1 бит ACK/NAK	10 бит Размер	7 бит Не используется

EDCL реализует алгоритм Go-Back-N, обеспечивающий надежность передач. 14-битовый порядковый номер в полученных пакетах проверяется на совпадение со значением внутреннего счетчика. Если они не совпадают, то операция не выполняется и в кадре ответа бит ACK/NAK устанавливается. Кадр ответа содержит внутреннее значение счетчика в поле порядкового номера. Если порядковые номера совпадают, то операция выполняется, внутреннее значение счетчика сохраняется в поле порядкового номера, в ответном кадре бит ACK/NAK сбрасывается и внутренний счетчик инкрементируется. Поле размера всегда сброшено для кадров ACK/NAK = 1. Неиспользуемое поле не проверяется и копируется в ответ. Таким образом, поле может быть установлено для содержания, например, дополнительных битов идентификатора, в случае необходимости.

Адресные установки IP и Ethernet EDCL

EDCL имеет фиксированный MAC-адрес – 02:00:EE:00:00:07 (см. таблицы 4.18.18 и 4.18.19). IP-адрес EDCL можно изменять программным способом (см. таблицу 4.18.17), значение по сбросу – 192.168.0.51.

Размер буфера EDCL

EDCL имеет выделенную внутреннюю буферную память, в которой сохраняются полученные пакеты во время обработки, см. таблицу 4.18.8. Попытка отправки большего числа пакетов, чем количество буферов пакетов до получения ответа, приведет к сбросу пакетов. В случае отправки пакетов, превышающих максимально допустимый размер, дальнейшее поведение контроллера не определено.

Таблица 4.18.8 – Размер буфера EDCL

Общий размер буфера (Кбайты)	Количество буферов пакетов	Размер буфера пакетов (байты)	Максимальная полезная нагрузка данных (байты)
2	4	512	456

4.18.7 Независимый от среды интерфейс (MII)

Для работы с подуровнем MAC и физическим уровнем контроллер GRETH использует независимый от среды интерфейс (MII), который определен в стандарте IEEE 802.3u. Интерфейс Ethernet реализован согласно настоящей спецификации. Он использует 16 управляющих сигналов.

Канал передачи данных (MII) включает в себя:

Transmit Data (ETX_D(3–0)) – группа параллельных сигналов данных, которые поступают на выход из MAC;

Transmit Enable (ETX_EN) – выход разрешение передачи. MAC устанавливает данный сигнал, когда установлены достоверные данные на TXD;

Transmit Error (ETX_ER) – выходной сигнал, который сообщает о том, что в передаваемом потоке данных произошла ошибка;

Transmit Clock (ETX_CLK) – внешний тактовый сигнал передатчика MAC.

Канал приема данных (MII) включает в себя:

Receive Data (ERX_D(3–0)) – группа параллельных сигналов данных, которые поступают на вход MAC;

Receive Data Valid (ERX_DV) – вход достоверности принимаемых данных;

Receive Error (ERX_ER) – вход ошибки приема;

Receive Clock (ERX_CLK) – внешний тактовый сигнал приемника MAC;

Collision Detected (ERX_COL) – сигнализирует о коллизии на линии;

Carrier Sense (ERX_CRS) – вход обнаружения несущей.

4.18.8 Программные драйверы

Драйверы MAC GRETH предусмотрены для следующих операционных систем: RTEMS, eCos, uClinux и Linux, начиная с версии 2.6. С текущей версией драйвера на исходном языке, интегрированной в ядро Linux, можно ознакомиться, перейдя по ссылке https://github.com/torvalds/linux/tree/master/drivers/net/ethernet/aeroflex.

4.18.9 Регистры

Программирование ядра осуществляется через регистры, отображаемые в адресуемом пространстве APB, см. таблицы 4.18.9 – 4.18.19.

Таблица 4.18.9 – Регистры GRETH (базовый адрес 0x80000500)

Смещение адреса APB	Регистр
0x00	Регистр управления
0x04	Регистр статуса/источника прерывания
0x08	Регистр старших значащих бит MAC-адреса
0x0C	Регистр младших значащих бит MAC-адреса
0x10	Регистр статуса/управления MDIO
0x14	Регистр указателя на дескриптор передачи
0x18	Регистр указателя на дескриптор приема
0x1C	Регистр IP адреса EDCL
0x20 – 0x24	Зарезервировано (групповая адресация не поддерживается)
0x28	Регистр старших значащих бит MAC-адреса EDCL
0x2C	Регистр младших значащих бит MAC-адреса EDCL

Регистр управления

Таблица 4.18.10 – Регистр управления GRETH

31	30 28	27	26	25	24 15	14	13	12	11	10	9 8	7	6	5	4	3	2	1	0
EA	BS	–	MA	MC	–	ED	RD	DD	ME	PI	–	SP	RS	PM	FD	RI	TI	RE	TE
1	0x1	0	1	0	0x000	0	0	0	0	0	0x0	1	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw	rw

31	Доступна EDCL (EA) – Бит установлен, поскольку EDCL реализована
30–28	Размер буфера EDCL (BS) – Показывает объем памяти, используемый для буферов EDCL. 0 = 1 Кбайт, 1 = 2 Кбайта, …, 6 = 64 Кбайта
27	Зарезервировано
26	Доступны прерывания MDIO (MA) – Бит установлен, поскольку модуль поддерживает прерывания MDIO

25	Доступна групповая адресация (MC) – Бит сброшен, поскольку модуль не поддерживает прием многоадресных пакетов
24–15	Зарезервировано
14	Запрет на EDCL (ED) – Устанавливается для снятия разрешения EDCL и сбрасывается для разрешения. Данный функционал не поддерживается
13	Разрешение отладки ОЗУ (RD) – Бит сброшен, поскольку режим отладки ОЗУ не доступен
12	Запрещение обнаружения дуплексного режима (DD) – Отключение конечного автомата (FSM) детектирования скорости/дуплекса EDCL. Если FSM не завершает обнаружение, интерфейс MDIO блокируется в режиме занятости. Если программному обеспечению требуется доступ к MDIO, FSM можно отключить; как только бит занятости будет сброшен, интерфейс MDIO будет доступен. Следует обратить внимание на то, что FSM не может быть повторно активирован
11	Разрешение групповой адресации (ME) – Разрешение приема групповых адресов
10	Разрешение прерывания при изменении статуса PHY (PI) – Разрешает генерацию прерывания при обнаружении изменения статуса PHY
9, 8	Зарезервировано
7	Скорость (SP) – Устанавливает текущий скоростной режим. Если бит сброшен – 10 Мбит, если установлен – 100 Мбит. Не используется интерфейсом MII, данный бит актуален для GRETH с интерфейсом RMII. Значение по умолчанию автоматически считывается из PHY после сброса
6	Сброс (RS) – Если установлено, контроллер GRETH возвращается к настройкам по умолчанию. Автоматически сбрасывается по завершении. До тех пор пока бит не сброшен, интерфейс подчиненного устройства должен использоваться только для опроса данного бита
5	«Неразборчивый» режим (PM) – Если бит установлен, GRETH работает в «неразборчивом» режиме, что означает, что GRETH принимает все пакеты независимо от адреса назначения
4	Полный дуплекс (FD) – Если бит установлен, GRETH работает в полнодуплексном режиме, в противном случае – в полудуплексе
3	Прерывание приемника (RI) – Разрешение прерываний приемника. Прерывание генерируется при приеме пакета, если бит установлен. Прерывание генерируется независимо от того, успешно получен пакет или с ошибкой
2	Прерывание передатчика (TI) – Разрешение прерываний передатчика. Прерывание генерируется при передаче пакета, если бит установлен. Прерывание генерируется независимо от того, успешно передан пакет или с ошибкой
1	Разрешение приема (RE) – Бит должен быть установлен каждый раз, когда активируют новые дескрипторы. Если этот бит установлен, GRETH считывает новые дескрипторы и при обнаружении неразрешенного дескриптора GRETH останавливает работу, пока вновь не будет установлен RE. Этот бит должен быть установлен после активации новых дескрипторов
0	Разрешение передачи (TE) – Бит должен быть установлен каждый раз, когда активируют новые дескрипторы. До тех пор, пока этот бит установлен, GRETH считывает новые дескрипторы. При обнаружении неразрешенного идентификатора контроллер останавливает работу, пока вновь не будет установлен TE. Этот бит должен быть установлен после активации новых дескрипторов

Регистр статуса/источника прерывания

Таблица 4.18.11 – Регистр статуса/источника прерывания GRETH

31 9	8	7	6	5	4	3	2	1	0
–	PS	IA	TS	TA	RA	TI	RI	TE	RE
0x000000	0	0	0	n/r	n/r	n/r	n/r	n/r	n/r
r	wc	wc	wc	wc	wc	wc	wc	wc	wc

31–9 Зарезервировано
8 Изменение статуса PHY (PS) – Устанавливается при обнаружении изменения статуса PHY
7 Недопустимый адрес (IA) – Получен пакет с адресом, не принятым MAC. Сбрасывается записью логической «1»
6 Слишком короткий (TS) – Получен пакет меньше минимального размера. Сбрасывается записью логической «1»
5 Ошибка AHB передатчика (TA) – Обнаружена ошибка AHB в механизме DMA передатчика. Очищается записью логической «1». Без сброса
4 Ошибка AHB приемника (RA) – Обнаружена ошибка AHB в механизме DMA приемника. Очищается записью логической «1». Без сброса
3 Прерывание передатчика (TI) – Пакет передан без ошибок. Очищается записью логической «1». Без сброса
2 Прерывание приема (RI) – Пакет принят без ошибок. Очищается записью логической «1». Без сброса
1 Ошибка передатчика (TE) – Пакет передан с ошибкой. Очищается записью логической «1». Без сброса
0 Ошибка приемника (RE) – Пакет принят с ошибкой. Очищается записью логической «1». Без сброса

Регистр старших значащих бит MAC-адреса

Таблица 4.18.12 – Регистр старших значащих бит MAC-адреса GRETH

31 16	15 0
–	Биты [47:32] MAC адреса
0x0000	n/r
r	rw

31–16 Зарезервировано
15–0 Два старших значащих байта MAC-адреса. Без сброса

Регистр младших значащих бит MAC-адреса

Таблица 4.18.13 – Регистр младших значащих бит MAC-адреса GRETH

31 0
Биты [31:0] MAC адреса
n/r
rw

31–0 Четыре младших значащих байта MAC-адреса. Без сброса

Регистр статуса/управления MDIO

Таблица 4.18.14 – Регистр статуса/управления MDIO GRETH

31 16	15 11	10 6	5 4	3	2	1	0
DATA	PHYADDR	REGADDR	–	BU	LF	RD	WR
0x0000	0x01	0x00	0x0	0	1	0	0
rw	rw	rw	r	r	r	rw	rw

31–16 Данные (DATA) – Содержит данные, считанные в ходе операции чтения, и данные для передачи

15–11 Адрес PHY (PHYADDR) – Поле содержит адрес PHY, доступ к которому открывается в ходе операции записи или операции считывания

10–6 Адрес регистра (REGADDR) – Поле содержит адрес регистра, доступ к которому открывается в ходе операции записи или операции считывания

5, 4 Зарезервировано

3 Занято (BU) – В ходе операции этот бит устанавливается. После завершения операции и при неактивности линии управления этот бит сбрасывается

2 Ошибка соединения (LF) – После завершения операции (BUSY = 0) этот бит устанавливается, если не обнаружено функциональное соединение

1 Чтение (RD) – Запускает операцию чтения через интерфейс управления. Данные сохраняются в поле данных

0 Запись (WR) – Запускает операцию записи через интерфейс управления. Данные берутся из поля данных

Регистр указателя на дескриптор передачи

Таблица 4.18.15 – Регистр указателя на дескриптор передачи GRETH

31 10	9 3	2 0
BASEADDR	DESCPNT	–
n/r	0x00	0x0
rw	rw	r

31–10 Базовый адрес строки таблицы указателя передатчика (BASEADDR) – Базовый адрес таблицы дескрипторов передатчика. Без сброса

9–3 Указатель на дескриптор (DESCPNT) – Указатель на отдельный дескриптор. Автоматически инкрементируется через Ethernet MAC

2–0 Зарезервировано

Регистр указателя на дескриптор приема

Таблица 4.18.16 – Регистр указателя на дескриптор приема GRETH

31 10	9 3	2 0
BASEADDR	DESCPNT	–
n/r	0x00	0x0
rw	rw	r

31–10 Базовый адрес строки таблицы указателя приемника (BASEADDR). Без сброса

9–3 Указатель на дескриптор (DESCPNT) – Указатель на отдельный дескриптор. Автоматически инкрементируется через Ethernet MAC

2–0 Зарезервировано

Регистр IP адреса EDCL

Таблица 4.18.17 – Регистр IP адреса EDCL GRETH

31	0
EDCL IP адрес	
0xC0A80033	
rw	

31–0 IP-адрес EDCL

Регистр старших значащих бит MAC-адреса EDCL

Таблица 4.18.18 – Регистр старших значащих бит MAC-адреса EDCL GRETH

31 16	15 0
–	Биты 47:32 EDCL MAC-адреса
0x0000	0x0200
r	rw

31–16 Зарезервировано
15–0 Два старших значащих байта MAC-адреса EDCL

Регистр младших значащих бит MAC-адреса EDCL

Таблица 4.18.19 – Регистр младших значащих бит MAC-адреса EDCL GRETH

31	0
Биты 31:0 EDCL MAC-адреса	
0xEE000007	
rw	

31–0 Четыре младших значащих байта MAC-адреса EDCL

4.19 Интерфейс CAN

4.19.1 Общие сведения

В состав микропроцессоров входят два интерфейса CAN. Базовые адреса интерфейсов указаны в таблице 2.5.1.

Интерфейс CAN обеспечивает мост между AHB AMBA и регистрами двух ядер CAN. Интерфейс подчиненного устройства AHB отображается в области ввода-вывода AHB. Сигнал прерывания блока CAN выставляется на шину прерывания AHB.

Блок-схема представлена на рисунке 4.19.1.

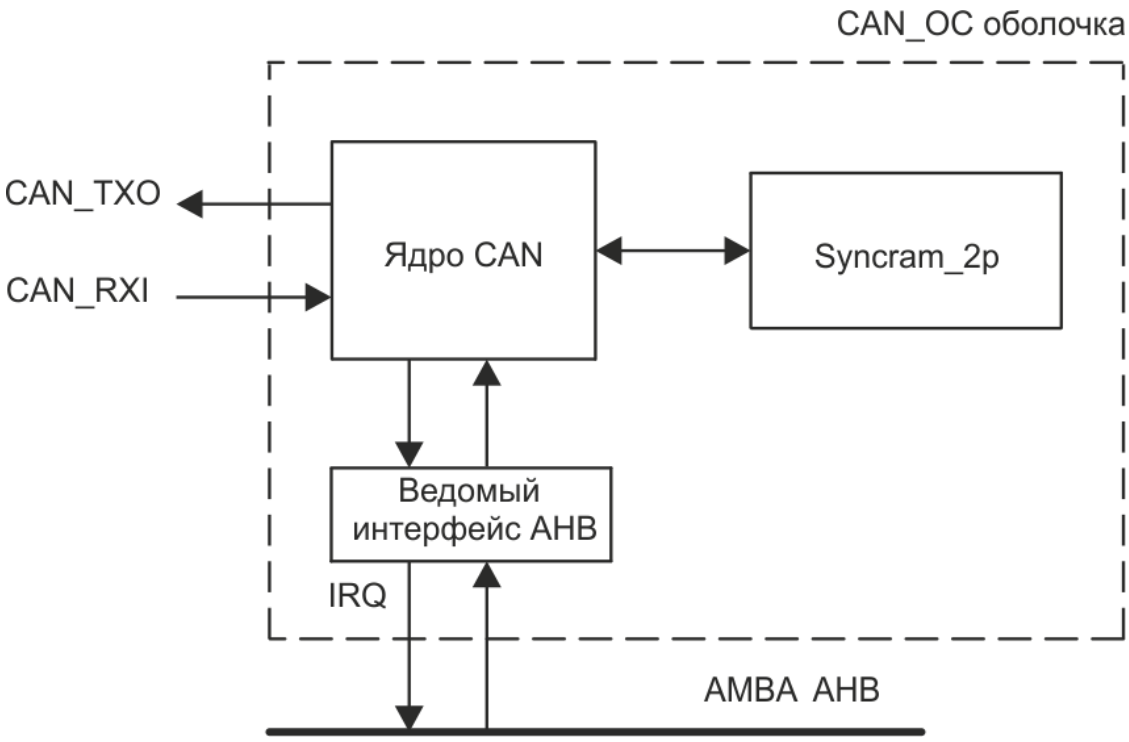


Рисунок 4.19.1 – Блок-схема ядра CAN

4.19.2 Общие сведения о контроллере интерфейса CAN

Контроллер CAN базируется на основе Philips SJA1000 и имеет совместимую карту регистров с несколькими исключениями. Он также поддерживает режимы BasicCAN (аналогичный PCA82C200) и PeliCAN. В режиме PeliCAN поддерживаются расширенные возможности протокола CAN 2.0B. Выбор режима работы осуществляется через регистр тактового предделителя.

В этом документе представлены регистры и описаны их функциональные возможности. Техническое описание Philips SJA1000 можно использовать в качестве справочного материала.

Карта регистров и функциональные возможности этих двух режимов работы отличаются. Описание режимов BasicCAN и PeliCAN приводится ниже. Типовые регистры (тактовый предделитель и синхронизация шины) описаны в отдельной главе. Карты регистров могут быть разными в зависимости от того, находится ли устройство в рабочем режиме или в режиме сброса. При сбросе устройство запускается в режиме сброса, ожидая задачу конфигурации. Для входа в рабочий режим необходимо сбросить бит запроса сброса в регистре управления. Чтобы повторно войти в режим сброса, необходимо опять установить этот бит в высокое логическое состояние.

4.19.3 Интерфейс AHB

Размер всех регистров – 1 байт, при этом адреса, фигурирующие в описании интерфейса, байтовые. Байтовое чтение и запись используются при взаимодействии через интерфейс с устройством. Байт чтения дублируется на всех байтовых линиях шины AHB. Данные передаются в обратном порядке – «сначала старший бит». Нумерация битов в описании интерфейса: бит 7 – старший значащий бит, бит 0 – младший значащий бит.

4.19.4 Режим BasicCAN

Карта регистров BasicCAN

Карта регистров BasicCAN представлена в таблице 4.19.1.

Таблица 4.19.1 – Размещение адреса BasicCAN (базовые адреса 0xFFFC0000 и 0xFFFC0100)

Адрес	Рабочий режим		Режим сброса	
	Чтение	Запись	Чтение	Запись
1	2	3	4	5
0	Управление	Управление	Управление	Управление
1	(0xFF)	Команда	(0xFF)	Команда
2	Статус	–	Статус	–
3	Прерывание	–	Прерывание	–
4	(0xFF)	–	Код приема	Код приема
5	(0xFF)	–	Маска приема	Маска приема
6	(0xFF)	–	Синхронизация шины 0	Синхронизация шины 0
7	(0xFF)	–	Синхронизация шины 1	Синхронизация шины 1
8	(0x00)	–	(0x00)	–
9	(0x00)	–	(0x00)	–
10	TX id1	TX id1	(0xFF)	–
11	TX id2, rtr, dlc	TX id2, rtr, dlc	(0xFF)	–
12	Байт данных TX 1	Байт данных TX 1	(0xFF)	–
13	Байт данных TX 2	Байт данных TX 2	(0xFF)	–
14	Байт данных TX 3	Байт данных TX 3	(0xFF)	–
15	Байт данных TX 4	Байт данных TX 4	(0xFF)	–
16	Байт данных TX 5	Байт данных TX 5	(0xFF)	–
17	Байт данных TX 6	Байт данных TX 6	(0xFF)	–
18	Байт данных TX 7	Байт данных TX 7	(0xFF)	–
19	Байт данных TX 8	Байт данных TX 8	(0xFF)	–
20	RX id1	–	RX id1	–
21	RX id2, rtr, dlc	–	RX id2, rtr, dlc	–
22	Байт данных RX 1	–	Байт данных RX 1	–
23	Байт данных RX 2	–	Байт данных RX 2	–
24	Байт данных RX 3	–	Байт данных RX 3	–
25	Байт данных RX 4	–	Байт данных RX 4	–
26	Байт данных RX 5	–	Байт данных RX 5	–
27	Байт данных RX 6	–	Байт данных RX 6	–
28	Байт данных RX 7	–	Байт данных RX 7	–
29	Байт данных RX 8	–	Байт данных RX 8	–
30	(0x00)	–	(0x00)	–
31	Тактовый предделитель	Тактовый предделитель	Тактовый предделитель	Тактовый предделитель

Регистр управления

Регистр управления содержит биты разрешения прерывания, а также бит запроса сброса, согласно таблице 4.19.2.

Таблица 4.19.2 – Интерпретирование бита в регистре управления (CR) (адрес 0)

Бит	Название	Описание
CR.7	–	Зарезервировано
CR.6	–	Зарезервировано
CR.5	–	Зарезервировано
CR.4	Разрешение прерывания при переполнении	1 – разрешено, 0 – не разрешено
CR.3	Разрешение прерывания при обнаружении ошибки	1 – разрешено, 0 – не разрешено
CR.2	Разрешение прерывания при передаче данных	1 – разрешено, 0 – не разрешено
CR.1	Разрешение прерывания при приеме данных	1 – разрешено, 0 – не разрешено
CR.0	Запрос сброса	При записи 1 в этом бите прерывается текущая передача данных и схема входит в режим сброса. Запись 0 возвращает схему в рабочий режим

Регистр команд

Запись «1» в соответствующем бите этого регистра запускает действие, поддерживаемое устройством, согласно таблице 4.19.3.

Таблица 4.19.3 – Интерпретирование бита в регистре команд (CMR) (адрес 1)

Бит	Название	Описание
CMR.7	–	Зарезервировано
CMR.6	–	Зарезервировано
CMR.5	–	Зарезервировано
CMR.4	–	Не используется
CMR.3	Сброс переполнения данных	Сбрасывает бит статуса переполнения данных
CMR.2	Очищение буфера приема	Очищает текущий буфер приема для приема новых данных
CMR.1	Прерывание передачи	Запрещает выполнение еще не начавшейся передачи
CMR.0	Запрос передачи	Начинает передачу сообщения в буфере TX

Передача запускается при записи «1» в CMR.0. Запрет выполнения передачи разрешен при записи «1» в CMR.1 при условии, что передача еще не началась. Если передача началась, она не прерывается, когда устанавливается CMR.1, но если произойдет ошибка, повторной передачи не будет. Подача команды «Очищение буфера приема» должна производиться после считывания содержимого буфера приема, чтобы освободить память.

Если в очереди FIFO другое сообщение, генерируется новое прерывание при получении данных (если разрешено), при этом снова устанавливается бит статуса буфера приема. Для сброса бита статуса переполнения данных необходимо сделать запись «1» в CMR.3.

Регистр статуса

Регистр статуса используется только для считывания данных и отображает текущий статус устройства в соответствии с таблицей 4.19.4.

Таблица 4.19.4 – Интерпретирование бита в регистре статуса (SR) (адрес 2)

Бит	Название	Описание
SR.7	Статус шины	«1», если блок работает с отключенной шиной и не участвует в работе шины
SR.6	Статус ошибки	По крайней мере показания одного счетчика ошибок достигли или превысили предельное значение CPU (96)
SR.5	Статус передачи	«1» при передаче сообщения
SR.4	Статус приема	«1» при получении сообщения
SR.3	Передача окончена	«1» указывает, что последнее сообщение успешно передано
SR.2	Статус буфера передачи	«1» означает, что CPU может записывать в буфер передачи
SR.1	Статус переполнения данных	«1», если сообщение было потеряно в связи с отсутствием места в FIFO
SR.0	Статус буфера приема	«1», если имеется сообщение в приемнике FIFO

Статус буфера приема сбрасывается при подаче команды очищения буфера приема и устанавливается в высокое логическое состояние, если в FIFO имеется несколько сообщений. Статус переполнения данных сигнализирует о том, что сообщение, которое было принято, не поместилось в FIFO, потому что недостаточно места.

Примечание – Этот бит отличается от SJA1000 и устанавливается после считывания FIFO.

Если статус буфера передачи в высоком логическом состоянии, CPU может сделать запись в буфер передачи. В процессе выполнения передачи буфер заблокирован и этот бит сброшен. Бит окончания передачи при подаче запроса передачи сбрасывается и не устанавливается до тех пор, пока сообщение не будет успешно передано.

Регистр прерывания

Регистр прерывания сигнализирует CPU о причине прерывания. Биты прерывания устанавливаются только при установке в регистр управления соответствующего бита разрешения прерывания согласно таблице 4.19.5.

Таблица 4.19.5 – Интерпретирование бита в регистре прерывания (IR) (адрес 3)

Бит	Название	Описание
IR.7	–	Зарезервировано
IR.6	–	Зарезервировано
IR.5	–	Зарезервировано
IR.4	–	Не используется
IR.3	Прерывание при переполнении данных	Устанавливается при установке SR.1
IR.2	Прерывание при ошибке	Устанавливается при изменении статуса ошибки или статуса шины
IR.1	Прерывание при передаче	Устанавливается при освобождении буфера передачи (бит статуса 0 →1)
IR.0	Прерывание при приеме	Этот бит установлен, если в FIFO имеется несколько сообщений

Этот регистр сбрасывается при чтении, за исключением IR.0. Следует обратить внимание на то, что этот регистр отличается от SJA1000, в котором все биты сбрасываются при чтении в режиме BasicCAN. Ядро сбрасывает бит прерывания приема при подаче команды «очищение буфера приема» (как в режиме PeliCAN).

Также следует обратить внимание на то, что биты IR.5 – IR.7 считываются как установленные, а бит IR.4 – как сброшенный.

Буфер передачи

В таблице 4.19.6 показана топология буфера передачи. В BasicCAN можно передавать и принимать только стандартные кадровые сообщения (сообщения EFF на шине игнорируются).

Таблица 4.19.6 – Топология буфера передачи

Адрес	Имя	Биты							
		7	6	5	4	3	2	1	0
10	Бит ID 1	ID.10	ID.9	ID.8	ID.7	ID.6	ID.5	ID.4	ID.3
11	Бит ID 2	ID.2	ID.1	ID.0	RTR	DLC.3	DLC.2	DLC.1	DLC.0
12	Данные TX 1	Бит TX 1							
13	Данные TX 2	Бит TX 2							
14	Данные TX 3	Бит TX 3							
15	Данные TX 4	Бит TX 4							
16	Данные TX 5	Бит TX 5							
17	Данные TX 6	Бит TX 6							
18	Данные TX 7	Бит TX 7							
19	Данные TX 8	Бит TX 8							

Если бит RTR установлен, байты данных не отправляются, но DLC остается в кадре данных и требует определения, согласно соответствующему кадру. Следует обратить внимание на то, что определить DLC, превышающий 8 байтов можно, но делать этого не следует во избежание несогласованности. Если DLC > 8, можно отправить только 8 байтов.

Буфер приема

Буфер приема с адресом 20 – 29 является видимой частью 64-байтового FIFO RX. Его топология идентична топологии буфера передачи.

Входной полосовой фильтр

Сообщения могут отфильтровываться по имени, используя код приема и регистры маски приема. Верхние 8 бит в 11-битовом имени сравниваются с битами в регистре кода приема, причем сравниваются только сброшенные в регистре маски приема биты. Если обнаружено соответствие, сообщение сохраняется в FIFO.

4.19.5 Режим PeliCAN

Карта регистров PeliCAN

Размещение адреса PeliCAN показано в таблице 4.19.7.

Таблица 4.19.7 – Размещение адреса PeliCAN(базовые адреса 0xFFFC0000 и 0xFFFC0100)

№	Рабочий режим				Режим сброса	
	Чтение		Запись		Чтение	Запись
1	2		3		4	5
0	Режим		Режим		Режим	Режим
1	(0x00)		Команда		(0x00)	Команда
2	Статус		–		Статус	–
3	Прерывание		–		Прерывание	–
4	Разрешение прерывания		Разрешение прерывания		Разрешение прерывания	Разрешение прерывания
5	Зарезервировано (0x00)		–		Зарезервировано (0x00)	–
6	Синхронизация шины 0		–		Синхронизация шины 0	Синхронизация шины 0
7	Синхронизация шины 1		–		Синхронизация шины 1	Синхронизация шины 1
8	(0x00)		–		(0x00)	–
9	(0x00)		–		(0x00)	–
10	Зарезервировано (0x00)		–		Зарезервировано (0x00)	–
11	Сбор данных о потерянном арбитраже		–		Сбор данных о потерянном арбитраже	–
12	Сбор данных о коде ошибки		–		Сбор данных о коде ошибки	–
13	Предельное значение ошибок		–		Предельное значение ошибок	Предельное значение ошибок
14	Счетчик ошибок RX		–		Счетчик ошибок RX	Счетчик ошибок RX
15	Счетчик ошибок TX		–		Счетчик ошибок TX	Счетчик ошибок TX
16	RX FI SFF	RX FI EFF	TX FI SFF	TX FI EFF	Код приема 0	Код приема 0
17	RX ID 1	RX ID 1	TX ID 1	TX ID 1	Код приема 1	Код приема 1
18	RX ID 2	RX ID 2	TX ID 2	TX ID 2	Код приема 2	Код приема 2
19	Данные RX 1	RX ID 3	Данные TX 1	TX ID 3	Код приема 3	Код приема 3
20	Данные RX 2	RX ID 4	Данные TX 2	TX ID 4	Маска приема 0	Маска приема 0
21	Данные RX 3	Данные RX 1	Данные TX 3	Данные TX 1	Маска приема 1	Маска приема 1
22	Данные RX 4	Данные RX 2	Данные TX 4	Данные TX 2	Маска приема 2	Маска приема 2
23	Данные RX 5	Данные RX 3	Данные TX 5	Данные TX 3	Маска приема 3	Маска приема 3
24	Данные RX 6	Данные RX 4	Данные TX 6	Данные TX 4	Зарезервировано (0x00)	–
25	Данные RX 7	Данные RX 5	Данные TX 7	Данные TX 5	Зарезервировано (0x00)	–
26	Данные RX 8	Данные RX 6	Данные TX 8	Данные TX 6	Зарезервировано (0x00)	–
27	FIFO	Данные RX 7	–	Данные TX 7	Зарезервировано (0x00)	–
28	FIFO	Данные RX 8	–	Данные TX 8	Зарезервировано (0x00)	–
29	Счетчик сообщений RX		–		Счетчик сообщений RX	–
30	(0x00)		–		(0x00)	–
31	Тактовый предделитель		Тактовый предделитель		Тактовый предделитель	Тактовый предделитель

Буферы передачи и приема имеют разную топологию, в зависимости от того, что передается/принимается: формат стандартного кадра (SFF) или формат расширенного кадра (EFF). См. пункт ниже.

Регистр режимов

Интерпретирование бита в регистре режимов (MOD) показано в таблице 4.19.8.

Таблица 4.19.8 – Интерпретирование бита в регистре режимов (MOD) (адрес 0)

Бит	Имя	Описание
MOD.7	–	Зарезервировано
MOD.6	–	Зарезервировано
MOD.5	–	Зарезервировано
MOD.4	–	Не используется (режим ожидания в SJA1000)
MOD.3	Режим фильтра приема	1 – режим одинарного фильтра, 0 – режим двойного фильтра
MOD.2	Режим самотестирования	Если установлен, контроллер – в режиме самотестирования
MOD.1	Режим прослушивания	Если установлен, контроллер – в режиме прослушивания
MOD.0	Режим сброса	При записи 1 в этом бите прерывается текущая передача данных, и разрешен режим сброса. Запись 0 возвращает в рабочий режим

Запись в MOD.1 – MOD.3 можно сделать при условии нахождения в режиме сброса.

В режиме ожидания устройство не отправляет подтверждения. Следует обратить внимание на то, что в отличие от SJA1000, устройство продолжает работу при обнаружении ошибок – передача активных ошибочных кадров не прекращается!

В режиме самотестирования устройство может успешно завершить передачу без получения подтверждения, если дана команда запроса самоприема. Следует обратить внимание на то, что устройство должно быть подключено к действующей шине, в режиме обратной петли не выполняются подключения устройств внутри микропроцессора.

Регистр команд

Запись «1» в соответствующем бите данного регистра запускает действие, поддерживаемое устройством. Интерпретирование бита в регистре команд (CMR) показано в таблице 4.19.9.

Таблица 4.19.9 – Интерпретирование бита в регистре команд (CMR) (адрес 1)

Бит	Имя	Описание
CMR.7	–	Зарезервировано
CMR.6	–	Зарезервировано
CMR.5	–	Зарезервировано
CMR.4	Запрос самоприема	Передает и одновременно получает сообщения
CMR.3	Переполнение со сбросом данных	Сбрасывает бит статуса переполнения данных
CMR.2	Очищение буфера приема	Очищает текущий буфер приема для приема новых данных
CMR.1	Запрет передачи	Запрещает выполнение еще не начавшейся передачи
CMR.0	Запрос передачи	Начинает передачу сообщения в буфере TX

Передача запускается при записи «1» в CMR.0. Запрет выполнения разрешен при записи 1 в CMR.1 при условии, что передача еще не началась. Одновременная установка

CMR.0 и CMR.1 обеспечивает так называемую передачу одиночного кадра, т.е. устройство не будет повторно передавать сообщение, если передача не удалась в первый раз.

Подача команды «Очищение буфера приема» должна производиться после считывания содержания буфера приема, чтобы освободить память. Если в очереди FIFO другое сообщение, генерируется новое прерывание при получении данных (если разрешено), при этом снова устанавливается бит статуса буфера приема.

Бит запроса самоприема и режим самотестирования обеспечивает самотестирование устройства без участия других устройств на шине. Сообщение одновременно передается и принимается, при этом генерируется прерывание при приеме и при передаче.

Регистр статуса

Регистр статуса используется только для считывания данных и отображает текущий статус устройства, как показано в таблице 4.19.10.

Таблица 4.19.10 – Интерпретирование бита в регистре статуса (SR) (адрес 2)

Бит	Название	Описание
SR.7	Статус шины	«1», если устройство работает с отключенной шиной и не участвует в работе шины
SR.6	Статус ошибки	По крайней мере, показания одного счетчика ошибок достигли или превысили предельное значение
SR.5	Статус передачи	«1» во время передачи сообщения
SR.4	Статус приема	«1» во время получения сообщения
SR.3	Передача окончена	«1» указывает, что последнее сообщение успешно передано
SR.2	Статус буфера передачи	«1» означает, что CPU может записывать в буфер передачи
SR.1	Статус переполнения данных	«1», если сообщение было потеряно в связи с отсутствием места в FIFO
SR.0	Статус буфера приема	«1», если имеется сообщение в приемнике FIFO

Бит статуса буфера приема сбрасывается, если в FIFO нет сообщений. Статус переполнения данных сигнализирует о том, что сообщение, которое было принято, не поместилось в FIFO, потому что недостаточно места.

Примечание – Этот бит отличается от SJA1000 и устанавливается после считывания FIFO.

Если статус буфера передачи в высоком логическом состоянии, CPU может сделать запись в буфере передачи. В процессе выполнения передачи буфер заблокирован, и этот бит сброшен.

Бит окончания передачи при подаче запроса передачи или запроса самоприема сбрасывается и не устанавливается до тех пор, пока сообщение не будет успешно передано.

Регистр прерывания

Регистр прерывания сигнализирует CPU о причине прерывания. Биты прерывания устанавливаются только при установке в регистр разрешения прерывания соответствующего бита разрешения прерывания, как показано в таблице 4.19.11.

Этот регистр сбрасывается при чтении с исключительным состоянием IR.0, то есть при отсутствии сообщений в FIFO.

Таблица 4.19.11 – Интерпретирование бита в регистре прерывания (IR) (адрес 3)

Бит	Название	Описание
IR.7	Прерывание при ошибке шины	Устанавливается при обнаружении ошибки на шине
IR.6	Прерывание при потере приоритета	Устанавливается, если устройство потеряло приоритет
IR.5	Прерывание при прекращении работы из-за обнаружения ошибки	Устанавливается, если устройство прекращает работу при обнаружении ошибки
IR.4	–	Не используется (прерывание при выходе из режима ожидания SJA1000)
IR.3	Прерывание при переполнении данных	Устанавливается при установке бита статуса переполнения данных
IR.2	Прерывание при ошибке	Устанавливается при изменении статуса ошибки или статуса шины
IR.1	Прерывание при передаче	Устанавливается при освобождении буфера передачи
IR.0	Прерывание при приеме	Этот бит установлен, если в FIFO имеются сообщения

Регистр разрешения прерывания

В регистре разрешения прерывания могут быть разрешены/не разрешены отдельные источники прерывания. Если они разрешены, в регистре прерывания можно установить соответствующий бит, при этом генерируется прерывание, как показано в таблице 4.19.12.

Таблица 4.19.12 – Интерпретирование бита в регистре разрешения прерывания (IER) (адрес 4)

Бит	Название	Описание
IR.7	Прерывание при ошибке шины	«1» – разрешено, «0» – не разрешено
IR.6	Прерывание при потере арбитража	«1» – разрешено, «0» – не разрешено
IR.5	Прерывание при прекращении работы из-за обнаружения ошибки	«1» – разрешено, «0» – не разрешено
IR.4	–	Не используется (прерывание при выходе из режима ожидания SJA1000)
IR.3	Прерывание при переполнении данных	«1» – разрешено, «0» – не разрешено
IR.2	Прерывание при ошибке	«1» – разрешено, «0» – не разрешено
IR.1	Прерывание при передаче	«1» – разрешено, «0» – не разрешено
IR.0	Прерывание при приеме	«1» – разрешено, «0» – не разрешено

Регистр сбора данных о потерянном приоритете

Интерпретирование бита в регистре сбора данных о потерянном приоритете (ALC) показано в таблице 4.19.13.

Если устройство теряет приоритет, позиция двоичного разряда последовательности данных процессора записывается в регистр сбора данных о потерянном приоритете. Регистр не меняет содержимое до окончания считывания.

Таблица 4.19.13 – Интерпретирование бита в регистре сбора данных о потерянном приоритете (ALC) (адрес 11)

Бит	Название	Описание
ALC.7 – 5	–	Зарезервировано
ALC.4 – 0	Номер бита	Бит, в котором произошла потеря приоритета

Регистр захвата кода ошибки

Интерпретирование бита в регистре захвата кода ошибки (ECC) показано в таблице 4.19.14.

Таблица 4.19.14 – Интерпретирование бита в регистре захвата кода ошибки (ECC) (адрес 12)

Бит	Название	Описание
ECC.7–6	Код ошибки	Номер кода ошибки
ECC.5	Направление	1 – ошибка при приеме, 0 – ошибка при передаче
ECC.4–0	Сегмент	В какой части кадра произошла ошибка

В случае ошибки шины регистр захвата кода ошибки устанавливается в зависимости от ошибки (при передаче или при приеме) и в зависимости от части кадра, в которой произошла ошибка. Как и регистр ALC, регистр ECC не меняет значение до его считывания. В таблице 4.19.15 приводится интерпретирование битов 7 – 6 ECC.

Таблица 4.19.15 – Интерпретирование кода ошибки

ECC.7–6	Описание
0	Ошибка бита
1	Ошибка формы
2	Ошибка содержимого
3	Другое

Интерпретирование битов 4 – 0 регистра ECC приведено в таблице 4.19.16.

Таблица 4.19.16 – Интерпретирование битов 4 – 0 ECC

ECC.4–0	Описание
0x03	Начало кадра
0x02	ID.28 – ID.21
0x06	ID.20 – ID.18
0x04	Бит SRTR
0x05	Бит IDE
0x07	ID.17 – ID.13
0x0F	ID.12 – ID.5
0x0E	ID.4 – ID.0
0x0C	Бит RTR
0x0D	Зарезервированный бит 1
0x09	Зарезервированный бит 0
0x0B	Код длины данных
0x0A	Поле данных
0x08	Последовательность CRC
0x18	Предделитель CRC
0x19	Слот подтверждения
0x1B	Предделитель подтверждения
0x1A	Окончание кадра
0x12	Перерыв
0x11	Активный флаг ошибки
0x16	Пассивный флаг ошибки
0x13	Доминирующие биты допустимости
0x17	Предделитель ошибки
0x1C	Флаг переполнения

Регистр ограничений ошибок и предупреждений

Этот регистр обеспечивает установку ограничений количества ошибок и предупреждений CPU. По умолчанию предельное значение – 96. Выполнять запись в регистр можно только в режиме сброса.

Регистр счетчика ошибок RX (адрес 14)

Этот регистр показывает значение счетчика ошибок RX. Выполнять запись в регистр можно только в режиме сброса. При отключении шины значение счетчика сбрасывается в «0».

Регистр счетчика ошибок TX (адрес 15)

Этот регистр показывает значение счетчика ошибок TX. Выполнять запись в регистр можно только в режиме сброса. Если происходит отключение шины, этот регистр инициализируется как протокол обратного счета, который определяет 128 событий сигнала вне шины, и статус отключения от шины может быть прочитан из этого регистра. CPU может вызвать отключение шины с помощью записи 0xFF в этот регистр. Следует обратить внимание на то, что в отличие от SJA1000, устройство сразу сигнализирует об отключении шины, а не после включения рабочего режима. Восстановление отключенной шины начинается при включении рабочего режима после записи 0xFF в этот регистр в режиме сброса.

Буфер передачи

Буфер передачи используется только для записи данных и закреплен за адресом 16 – 28. Чтение данных из этой области возможно после их преобразования в буфере приема, описание которого приводится в следующем пункте. Топология буфера передачи зависит от того, что передается: стандартный кадр (SFF) или расширенный кадр (EFF), как показано в таблице 4.19.17.

Таблица 4.19.17 – Топология буфера передачи

№	Запись (SFF)	Запись (EFF)
16	Информация о кадре TX	Информация о кадре TX
17	TX ID 1	TX ID 1
18	TX ID 2	TX ID 2
19	Данные TX 1	TX ID 3
20	Данные TX 2	TX ID 4
21	Данные TX 3	Данные TX 1
22	Данные TX 4	Данные TX 2
23	Данные TX 5	Данные TX 3
24	Данные TX 6	Данные TX 4
25	Данные TX 7	Данные TX 5
26	Данные TX 8	Данные TX 6
27	–	Данные TX 7
28	–	Данные TX 8

Информация о кадре TX (топология этого поля одинакова для кадров SFF и EFF) показана в таблице 4.19.18.

Таблица 4.19.18 – Информация о кадре TX (адрес 16)

Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
FF	RTR	–	–	DLC.3	DLC.2	DLC.1	DLC.0

Бит 7 – FF выбирает формат кадра, т.е. интерпретирование его как расширенный или стандартный кадр. 1 = EFF, 0 = SFF.

Бит 6 – RTR должен быть установлен для кадра запроса удаленной передачи.

Биты 5, 4 – безразличное состояние.

Биты (3 – 0) – DLC определяет код длины данных, присваивая значение от 0 до 8. Если значение превышает 8, передаются 8 байтов.

Идентификатор TX 1 (это поле одинаково для кадров SFF и EFF), как показано в таблице 4.19.19.

Таблица 4.19.19 – Идентификатор TX 1 (адрес 17)

Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
ID.28	ID.27	ID.26	ID.25	ID.24	ID.23	ID.22	ID.21

Биты (7 – 0) – Верхние 8 битов идентификатора.

В таблице 4.19.20 представлен идентификатор TX 2, кадр SFF.

Таблица 4.19.20 – Идентификатор TX 2 (адрес 18)

Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
ID.20	ID.19	ID.18	–	–	–	–	–

Биты (7 – 5) – Нижние 3 бита идентификатора SFF.

Биты (4 – 0) – безразличное состояние.

В таблице 4.19.21 представлен идентификатор TX 2, кадр EFF

Таблица 4.19.21 – Идентификатор TX 2 (адрес 18)

Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
ID.20	ID.19	ID.18	ID.17	ID.16	ID.15	ID.14	ID.13

Биты (7 – 0) – Биты (20 – 13) 29-битового идентификатора EFF.

В таблице 4.19.22 представлен идентификатор TX 3, кадр EFF.

Таблица 4.19.22 – Идентификатор TX 3 (адрес 19)

Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
ID.12	ID.11	ID.10	ID.9	ID.8	ID.7	ID.6	ID.5

Биты (7 – 0) – Биты (12 – 5) 29-битового идентификатора EFF.

В таблице 4.19.23 представлен идентификатор TX 4, кадр EFF.

Таблица 4.19.23 – Идентификатор TX 4 (адрес 20)

Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
ID.4	ID.3	ID.2	ID.1	ID.0	–	–	–

Биты (7 – 3) – Биты (4 – 0) 29-битового идентификатора EFF.

Буфер приема

Для кадров SFF поле данных расположено в адресе 19–26 и для кадров EFF – в адресе 21–28, как показано в таблице 4.19.24.

Таблица 4.19.24 – Поле данных для кадров SFF и EFF

№	Чтение (SFF)	Чтение (EFF)
1	2	3
16	Информация о кадре RX	Информация о кадре RX
17	RX ID 1	RX ID 1
18	RX ID 2	RX ID 2
19	Данные RX 1	RX ID 3
20	Данные RX 2	RX ID 4
21	Данные RX 3	Данные RX 1
22	Данные RX 4	Данные RX 2
23	Данные RX 5	Данные RX 3
24	Данные RX 6	Данные RX 4
25	Данные RX 7	Данные RX 5
26	Данные RX 8	Данные RX 6
27	FI RX следующего сообщения в FIFO	Данные RX 7
28	ID1 RX следующего сообщения в FIFO	Данные RX 8

В таблице 4.19.25 дана информация о кадре RX (топология этого поля одинакова для кадров SFF и EFF).

Таблица 4.19.25 – Информация о кадре RX (адрес 16)

Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
FF	RTR	0	0	DLC.3	DLC.2	DLC.1	DLC.0

Бит 7 – Формат кадра принятого сообщения. 1 = EFF, 0 = SFF.
Бит 6 – «1» в случае кадра RTR.
Биты (5, 4) – Всегда «0».
Биты (3 – 0) – DLC определяет код длины данных.

Идентификатор RX 1 (это поле одинаково для кадров SFF и EFF), как показано в таблице 4.19.26.

Таблица 4.19.26 – Идентификатор RX 1 (адрес 17)

Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
ID.28	ID.27	ID.26	ID.25	ID.24	ID.23	ID.22	ID.21

Биты (7 – 0) – Старшие 8 бит идентификатора.

В таблице 4.19.27 представлен идентификатор RX 2, кадр SFF.

Таблица 4.19.27 – Идентификатор RX 2 (адрес 18)

Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
ID.20	ID.19	ID.18	RTR	0	0	0	0

Биты (7 – 5) – Нижние 3 бита идентификатора SFF.
Бит 4 – «1» в случае кадра RTR.
Биты (3 – 0) – Всегда «0».

В таблице 4.19.28 представлен идентификатор RX 2, кадр EFF.

Таблица 4.19.28 – Идентификатор RX 2 (адрес 18)

Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
ID.20	ID.19	ID.18	ID.17	ID.16	ID.15	ID.14	ID.13

Биты (7 – 0) – Биты (20–13) 29-битового идентификатора EFF.
В таблице 4.19.29 представлен идентификатор RX 3, кадр EFF.

Таблица 4.19.29 – Идентификатор RX 3 (адрес 19)

Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
ID.12	ID.11	ID.10	ID.9	ID.8	ID.7	ID.6	ID.5

Биты (7 – 0) – Биты (12 – 5) 29-битового идентификатора EFF.
В таблице 4.19.30 представлен идентификатор RX 4, кадр EFF.

Таблица 4.19.30 – Идентификатор RX 4 (адрес 20)

Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
ID.4	ID.3	ID.2	ID.1	ID.0	RTR	0	0

Биты (7 – 3) – Биты (4–0) 29–битового идентификатора EFF.
Бит 2 – «1» в случае кадра RTR.
Биты (1, 0) – безразличное состояние.

Входной полосовой фильтр

Входной полосовой фильтр может использоваться для фильтрации сообщений, не удовлетворяющих определенным требованиям. Если сообщение отфильтровывается, оно не помещается в FIFO приема и CPU его не обрабатывает.

Существует два разных режима фильтрации: одноступенчатого и двойного фильтра. Выбор фильтра задается битом 3 в регистре режимов. В режиме одинарного фильтра используется только один 4-байтовый фильтр. В режиме двойного фильтра используются два простых фильтра, и если один из них сигнализирует о соответствии, сообщение принимается. Каждый фильтр состоит из двух частей: код приема и маска приема. Регистры кодов используются для определения конкретного набора для соответствия, а регистры масок определяют биты с безразличным состоянием. Всего для входного полосового фильтра используются восемь регистров, как показано в таблице 4.19.31. Следует обратить внимание на то, что в них чтение/запись выполняется только в режиме сброса.

Таблица 4.19.31 – Регистры входного полосового фильтра

Адрес	Описание
16	Код приема 0 (ACR0)
17	Код приема 1 (ACR1)
18	Код приема 2 (ACR2)
19	Код приема 3 (ACR3)
20	Маска приема 0 (AMR0)
21	Маска приема 1 (AMR1)
22	Маска приема 2 (AMR2)
23	Маска приема 3 (AMR3)

Режим одинарного фильтра, стандартный кадр

При приеме стандартного кадра в режиме одинарного фильтра регистры ACR(0–3) сравниваются с входящим сообщением следующим образом:

ACR0.7–0 и ACR1.7–5 сравниваются с ID.28–18;
ACR1.4 сравнивается с битом RTR;
ACR1.3–0 не используются;
ACR2 и ACR3 сравниваются с байтами данных 1 и 2.

Соответствующие биты в регистрах AMR выбирают, имеет ли значение результат сравнения. Если в регистре масок бит установлен, то результат сравнения не имеет значения.

Режим одинарного фильтра, расширенный кадр

При приеме расширенного кадра в режиме одинарного фильтра регистры ACR(0–3) сравниваются с входящим сообщением следующим образом:

ACR0.7–0 и ACR1.7–0 сравниваются с ID.28–13;
ACR2.7–0 и ACR3.7–3 сравниваются с ID.12–0;
ACR3.2 сравнивается с битом RTR;
ACR3.1–0 не используются.

Соответствующие биты в регистрах AMR выбирают, имеет ли значение результат сравнения. Если в регистре масок бит установлен, то результат сравнения не имеет значения.

Режим двойного фильтра, стандартный кадр

При приеме стандартного кадра в режиме двойного фильтра регистры ACR(0–3) сравниваются с входящим сообщением следующим образом:

Фильтр 1
ACR0.7–0 и ACR1.7–5 сравниваются с ID.28–18;
ACR1.4 сравнивается с битом RTR;
ACR1.3–0 сравниваются со старшей тетрадой байта данных 1;
ACR3.3–0 сравниваются с младшей тетрадой байта данных 1.

Фильтр 2
ACR2.7–0 и ACR3.7–5 сравниваются с ID.28–18;
ACR3.4 сравнивается с битом RTR.

Соответствующие биты в регистрах AMR выбирают, имеет ли значение результат сравнения. Если в регистре масок бит установлен, то результат сравнения не имеет значения.

Режим двойного фильтра, расширенный кадр

При приеме расширенного кадра в режиме двойного фильтра регистры ACR0–3 сравниваются с входящим сообщением следующим образом:

Фильтр 1
ACR0.7–0 и ACR1.7–0 сравниваются с ID.28–13;

Фильтр 2
ACR2.7–0 и ACR3.7–0 сравниваются с ID.28–13.

Соответствующие биты в регистрах AMR выбирают, имеет ли значение результат сравнения. Если в регистре масок бит установлен, то результат сравнения не имеет значения.

Счетчик сообщений RX

Регистр счетчика сообщения RX по адресу 29 содержит количество сообщений, которые в настоящее время хранятся в FIFO приема. Старшие три бита всегда сброшены.

4.19.6 Общие регистры

Имеется три общих регистра с одинаковыми адресами и одинаковыми функциональными возможностями в обоих режимах BasiCAN и PeliCAN. Это регистр тактового предделителя и регистры синхронизации шины «0» и «1».

Регистр тактового предделителя

Регистр тактового делителя представлен в таблице 4.19.32 . Функционал тактового делителя не реализован, выход CLKOUT устройства CAN не доступен. Регистр тактового делителя управляет только выбором режима работы BasiCAN / PeliCAN.

Таблица 4.19.32 – Интерпретирование битов в регистре тактового предделителя (CDR) (адрес 31)

Бит	Имя	Описание
CDR.7	Режим CAN	1 – PeliCAN, 0 – BasiCAN (значение по сбросу)
CDR.6	–	Не используется (бит cbp SJA1000)
CDR.5	–	Не используется (бит rxinten SJA1000)
CDR.4	–	Зарезервировано
CDR.3	Отключение тактового сигнала	Отключает выход CLKOUT (не доступен)
CDR.2–0	Тактовый делитель	Выбор частоты (не доступен)

Регистр временных параметров BTR0 шины

В таблице 4.19.33 представлено интерпретирование битов регистра шины (BTR0).

Таблица 4.19.33 – Интерпретирование битов регистра шины BTR0 (адрес 6)

Бит	Имя	Описание
BTR0.7–6	SJW	Длительность синхронизации перехода
BTR0.5–0	BRP	Предделитель скорости передачи

Системный тактовый сигнал ядра CAN рассчитывается следующим образом:

$$t_{scl} = 2 \times t_{clk} \times (BRP + 1),$$

где t_{clk} – системный тактовый сигнал.

Длительность синхронизации перехода определяет, сколько циклов синхронизации (t_{scl}) за один интервал передачи бита можно выровнять за одну повторную синхронизацию.

Регистр временных параметров BTR1 шины

В таблице 4.19.34 представлено интерпретирование битов регистра шины (BTR1).

Таблица 4.19.34 – Интерпретирование битов регистра шины BTR1 (адрес 7)

Бит	Имя	Описание
BTR1.7	SAM	«1» – шина тестируется 3 раза, «0» – точка единичного тестирования
BTR1.6–4	TSEG2	Временной отрезок 2
BTR1.3–0	TSEG1	Временной отрезок 1

Интервал побитовой передачи шины CAN определяется системным тактовым сигналом CAN и временными отрезками 1 и 2, как показано в уравнениях ниже:

$$t_{tseg1} = t_{scl} \times (TSEG1 + 1),$$
$$t_{tseg2} = t_{scl} \times (TSEG2 + 1),$$
$$t_{bit} = t_{tseg1} + t_{tseg2} + t_{scl}.$$

Дополнительная переменная t_{scl} образуется из стартового отрезка синхронизации. Тестирование проводится между интервалами битовой передачи TSEG1 и TSEG2.

4.19.7 Особенности интерфейса CAN

В этом пункте перечислены различия между этим контроллером CAN и SJA1000, на котором он базируется:

- все биты, связанные с режимом ожидания, недоступны;
- регистры управления выходом и тестовые регистры отсутствуют (чтение 0x00);
- бит 6 регистра тактового предделителя (CBP) и 5 (RXINTEN) не реализован;
- запрос на прерывание при переполнении и статус не устанавливаются, пока не считаны данные FIFO.

Специфические отличия BasicCAN:

- бит запроса на прерывание при приеме не сбрасывается при чтении, работает как в режиме PeliCAN;
- бит CR.6 всегда сброшен и не является регистром, как в SJA1000.

Специфические отличия PeliCAN:

- запись 256 в регистр счетчика ошибок TX, в режиме сброса, сразу отключает шину;
- регистр стартового адреса буфера чтения отсутствует;
- адреса выше 31 не реализованы (т.е. внутренний доступ RAM/FIFO);
- устройство передает активные кадры ошибок только в режиме ожидания.

4.20 Интерфейс USB 2.0

4.20.1 Общие сведения

Контроллер хост-системы USB 2.0 Gaisler Research (GRUSBHC) обеспечивает связь между шиной AHB AMBA и универсальной последовательной шиной (USB). Стандарт USB 2.0 во многом аналогичен стандарту USB 1.1 и совместим с ним. Однако, к двум прежним скоростям в нём добавляется новая скорость – 480 Мбайт/с. Хост-контроллер поддерживает поток обмена USB в высокоскоростном, полноскоростном и низкоскоростном режимах. В высокоскоростном режиме USB 2.0 используется усовершенствованный контроллер с реализацией усовершенствованного интерфейса контроллера хост-системы версии 1.0 (EHCI 1.0). Поток обмена в полноскоростном и низкоскоростном режимах (USB 1.1) обрабатывается универсальным контроллером, реализующим интерфейс версии 1.1 (UCHI 1.1). Каждый контроллер имеет собственный интерфейс ведущего устройства AMBA AHB. Конфигурация и управление усовершенствованного контроллера хост-системы выполняется через шину AMBA APB. Доступ к регистрам сопровождающего контроллера выполняется через интерфейс

ведомого устройства AMBA AHB. На рисунке 4.20.1 показана блок-схема хост-системы USB 2.0.

Контроллер поддерживает приемо-передатчики ULPI.

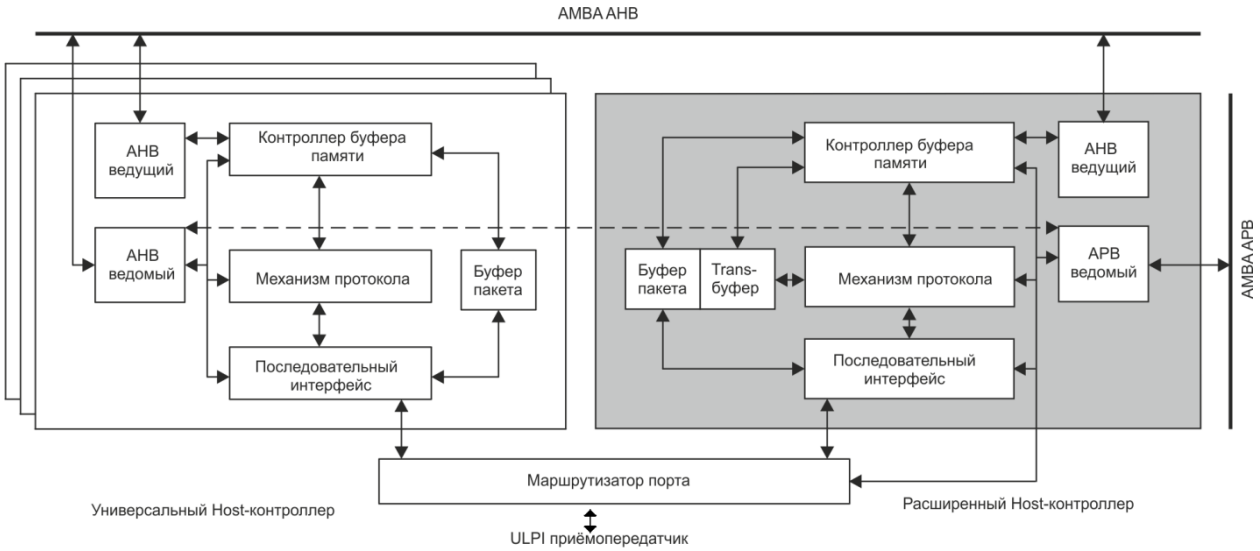


Рисунок 4.20.1 – Блок-схема хост-системы

4.20.2 Принцип работы

Общие сведения о системе

В системе содержатся два типа контроллеров: усовершенствованный контроллер и универсальный контроллер. Универсальный контроллер работает, как сопровождающий контроллер усовершенствованного контроллера.

Усовершенствованный контроллер соответствует интерфейсу усовершенствованного контроллера хост-системы за исключением опции Light Host Controller Reset (см. спецификацию Enhanced Host Controller Interface (EHCI)), которая не реализована.

Универсальный контроллер соответствует интерфейсу универсального контроллера хост-системы с некоторыми исключениями. Поле HCHalted в регистре статуса USB реализовано только для считывания вместо считывания/записи. Регистр статуса/управления портом (Port Status/Control) расширен полями «превышение тока» (Over Current) и «изменение превышения тока» (Over Current Change). Изменения в обоих регистрах выполняются в соответствии с текущей реализацией интерфейса. Изменения соответствуют описанию битов в спецификации EHCI.

Поддержка протокола

Усовершенствованный контроллер полностью поддерживает поток обмена в высокоскоростном режиме в соответствии со спецификацией USB 2.0. Кроме того, поддерживается режим Asynchronous Park, а контроллер имеет счетчик NAK.

Универсальный контроллер хост-системы поддерживает поток обмена в полноскоростном и низкоскоростном режимах.

Дескриптор и буферизация данных

Усовершенствованный контроллер хост-системы выполняет предвыборку одного кадра изохронных дескрипторов. Перед выполнением передачи производится выборка

всех данных полезной нагрузки. Усовершенствованный контроллер имеет 2048-байтовый буфер для дескрипторов и 2048-байтовый буфер для данных полезной нагрузки, которые могут содержать данные для двух передач.

Универсальный контроллер хост-системы не выполняет предвыборку указателей. Передача на шине может начинаться до извлечения из памяти всех данных полезной нагрузки, после извлечения трех слов данных. Универсальный контроллер имеет 1024-байтовый буфер для данных полезной нагрузки. Дескриптор передачи в UHCI описывает передачу, полезная нагрузка которой составляет 1280 байт. Согласно спецификации USB, максимально допустимая полезная нагрузка данных ограничена 1023 байтами, при этом контроллер не передает полезную нагрузку, превышающую 1023 байта. Если полезная, разрешенная, нагрузка дескриптора превышает 1023 байта, контроллер сделает попытку передачи первых 1023 байт, после чего передача будет помечена как завершенная.

Универсальный контроллер и усовершенствованный контроллер используют общий буфер данных полезной нагрузки. Таким образом, требуется только два 2048-байтных буфера.

Тактирование и сброс

Ядро модуля USB содержит два тактовых домена: системный тактовый домен и тактовый домен USB. Тактовый домен USB работает с частотой 60 МГц. Все сигналы, пересекающие границу тактового домена, синхронизированы во избежание нестабильности.

Переключение входа сброса может быть асинхронным или синхронным. Все регистры в системном домене тактирования, требующие сброса, сбрасываются синхронно. Предполагается, что вход сброса удерживается установленным до стабилизации сигнала тактирования. Во избежание нежелательных действий USB, в тактовом домене USB асинхронно сбрасывается несколько регистров. При старте тактирования USB все остальные регистры в домене USB, требующие значения по сбросу, также сбрасываются.

От сброса устройство генерирует сигнал сброса приемо-передатчиков USB, который асинхронно устанавливается применительно к тактовому сигналу USB, и асинхронно снимается.

Порядок следования байт

Устройство всегда выполняет доступ к младшему значащему байту полезной нагрузки данных при нулевом смещении. Регистры имеют порядок следования «сначала младший байт» с перестановкой байтов.

4.20.3 Операции DMA

В обоих типах контроллеров размер пакетов DMA составляет 16 слов.

Это значение показывает, сколько слов в пакете контроллер может получить из памяти, а не количество операций с памятью при получении доступа к шине.

При записи данных полезной нагрузки обратно в память, требующей адресации по байтам или в полусловах, количество операций может достигнуть 17, прежде чем шина будет освобождена.

Если хост-контроллером дается побайтно выровненный буфер данных, длина пакета может достичь 17 слов при выборке данных полезной нагрузки из памяти.

Универсальный контроллер хост-системы использует пакет размером в четыре слова при выборке Дескрипторов. Конфигурация универсального контроллера позволяет запускать передачи по USB до полного извлечения данных из памяти. Количество слов, которые должны быть извлечены из памяти до запуска передачи по USB, равно двум.

4.20.4 Порядок следования байт

Внутренние операции выполняются в порядке «сначала младший байт». Поскольку система подсоединена к шине, использующей порядок следования «сначала старший байт», все строки данных AMBA используют перестановку байт.

Регистры хост-контроллера используют порядок «сначала младший байт» с перестановкой байтов DWORD.

В таблице 4.20.1 показан порядок следования «сначала младший байт» с перестановкой байтов (32 бита) в двух 16-битовых регистрах. Регистр R1 расположен по адресу 0x00, регистр R2 расположен по адресу 0x02.

Таблица 4.20.1 – R1 и R2 с адресацией в порядке следования «сначала младший байт» с перестановкой байтов DWORD

31 24	23 16	15 8	7 0
R1(7:0)	R1(15:8)	R2(7:0)	R2(15:8)

Дескрипторы передач внутренней памяти также используют перестановку байт. Данная конфигурация используется для операционных систем, таких как Linux, в которых используется перестановка байтов в системах, использующих порядок следования «сначала старший байт».

4.20.5 Поддержка приемо-передатчиков

Контроллер поддерживает приемо-передатчики ULPI. Используется внешний источник питания шины USB (VBUS) и внешний индикатор ошибки с активным низким уровнем.

Описание интерфейса ULPI приводится в UTMI+ Low Pin Interface Specification версии 1.1.

4.20.6 Регистры

Усовершенствованный контроллер хост-системы

Программирование выполняется через регистры, отображаемые в адресном пространстве APB, как показано в таблицах 4.20.2, 4.20.3. Описание регистров приводится в спецификации EHCI.

Таблица 4.20.2 – Регистры возможностей усовершенствованного хост-контроллера (базовый адрес 0x80100100)

Смещение адреса APB	Регистр
0x00	Длина регистра возможностей
0x01	Зарезервировано
0x02	Номер версии интерфейса
0x04	Структурные параметры
0x08	Параметры возможностей
0x0C	Описание распределения портов по контроллерам-компаньонам

Таблица 4.20.3 – Операционные регистры усовершенствованного контроллера хост-системы (базовый адрес 0x80100100)

Смещение адреса APB	Регистр
0x14	Управление USB*
0x18	Статус USB
0x1C	Разрешение прерывания USB
0x20	Индекс кадра USB
0x24	Селектор сегмента 4G (зарезервировано)
0x28	Базовый адрес списка кадров
0x2C	Адрес следующего асинхронного списка
0x54	Регистр сконфигурированных флагов
0x58	Регистр статуса/управления порта

_____

* Мягкий сброс хост-контроллера (Light Host Controller Reset) не реализован.

Универсальный контроллер хост-системы

Программирование контроллера выполняется через регистры, отображаемые в адресном пространстве области ввода-вывода AHB. Их список и особенности реализации приведены в таблицах 4.20.4 – 4.20.6. Общее описание регистров приводится в руководстве по проектированию интерфейса универсального контроллера хост-системы (UHCI) версии 1.1.

Таблица 4.20.4 – Регистры универсального хост-контроллера (базовый адрес 0xFFFA0000)

Смещение адреса AHB	Регистр
0x00	Управление USB
0x02	Статус USB*
0x04	Разрешение прерывания USB
0x06	Номер кадра
0x08	Базовый адрес списка кадров
0x0C	Начало кадра модификации
0x10	Статус/управление портом**

_____

* Бит HCH реализован только для считывания и по умолчанию имеет значение 1.

** Добавлены поля «обнаружено превышение по току» и «наблюдается превышение по току».

Таблица 4.20.5 – Изменения в регистре статуса USB

15 … 6	5	4 … 0
Соответствует UHCI	HCH	Соответствует UHCI
	1	
	r	

15–6 Соответствует UHCI

5 Контроллер хост-системы остановлен (HCH) – Аналогично спецификации UHCI, но поле изменено с Read/Write Clear на Read Only. Бит сброшен, когда Run/Stop установлен. Значение данного бита по умолчанию изменено на 1.

4–0 Соответствует UHCI

Таблица 4.20.6 – Изменения в регистре статуса/управления порта

15 12	11	10	9 0
Соответствует UHCI	OCC	OC	Соответствует UHCI
	0	0	
	wc	r	

15–12 Соответствует UHCI

11 Обнаружено превышение по току (OCC) – Устанавливается при переключении сигнала превышения по току (OC)

10 Наблюдается превышение по току (OC) – Бит установлен, пока наблюдается превышение по току

9–0 Соответствует UHCI

4.21 Интерфейс PCI

4.21.1 Общие сведения

Интерфейс реализован на основе модуля GRPCI2. Этот модуль является мостом между шиной PCI и шиной AHB AMBA. Интерфейс соединяется с шиной PCI через интерфейс ведомого и интерфейс ведущего устройства. При подключении к шине AMBA интерфейс ведущего устройства AHB используется для работы с ведомым PCI, интерфейс ведомого устройства AHB – для работы с ведущим устройством PCI. Реализован контроллер прямого доступа к памяти (DMA). Для организации работы DMA используется PCI инициатор, подключенный к шине PCI, и ведущее устройство AHB, подключенное к шине AMBA. Доступ к регистрам конфигурации выполняется через интерфейс ведомого устройства APB AMBA, как показано на рисунке 4.21.1.

Интерфейсы PCI и AMBA принадлежат к двум разным тактовым доменам. Синхронизация выполняется через буферы FIFO.

Интерфейс PCI соответствует спецификации локальной шины PCI 2.2.

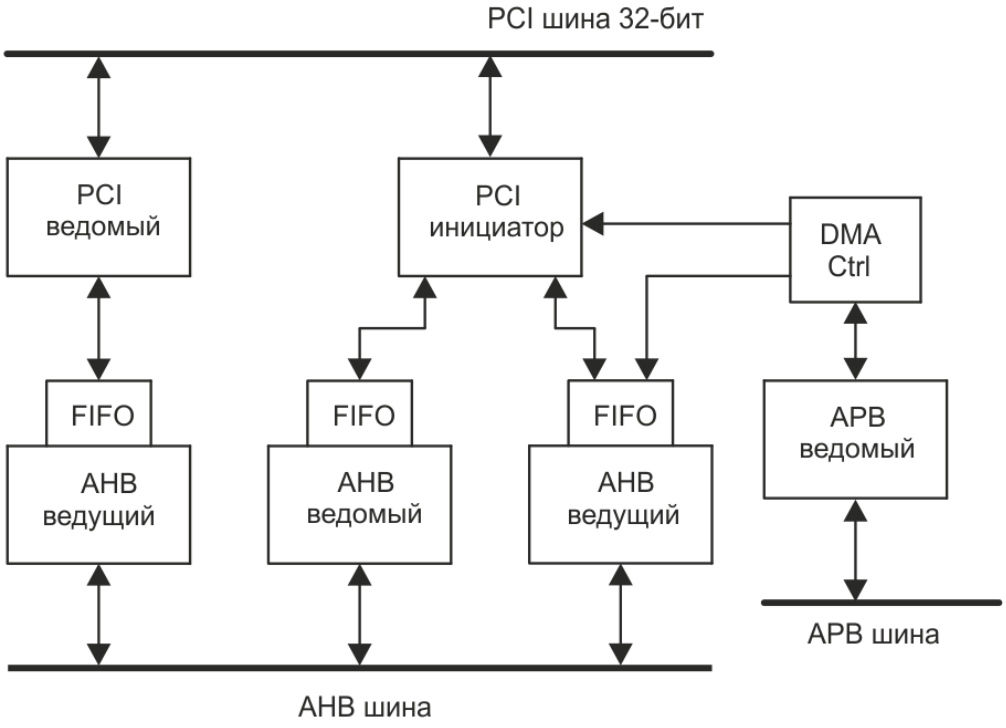


Рисунок 4.21.1 – Блок-схема интерфейса PCI

4.21.2 Конфигурация

Регистры конфигурации располагаются в области конфигурации PCI (в соответствии со спецификацией локальной шины PCI 2.2) и доступны через интерфейс ведомого устройства APB AMBA (для управления работой системы и управления DMA).

Базовый адрес размещения управляющих регистров **0x80000400.**

Реализованная конфигурация определяется считыванием регистра статуса, доступ к которому выполняется через интерфейс ведомого устройства APB:

- идентификация поставщика и устройства PCI: vendorid = 0x0, deviceid = 0x0;
- код класса PCI и идентификация версии: classcode = 0x0 и revisionid = 0x0;
- реализован интерфейс 32-битового инициирующего устройства PCI;
- реализован интерфейс 32-битового адресата (ведомого) PCI;
- реализован контроллер DMA;
- емкость и количество FIFO: емкость равна 3, количество равно 2;
- номер генерируемого прерывания 6.

Область конфигурации PCI

Реализованы следующие регистры в области конфигурации PCI (см. таблицы 4.21.1 – 4.21.8). Порядок следования байт данной области приведен в пункте «Циклы конфигурации PCI». Для получения дополнительной информации о каждом поле данных регистров – см. спецификацию локальной шины PCI.

Таблица 4.21.1 – Реализованные регистры области конфигурации PCI (базовый адрес 0xFFF90000)

Смещение адреса PCI	Регистр
0x00	Регистр идентификации устройства и идентификации поставщика
0x04	Регистр статуса и управления
0x08	Регистр кода класса и идентификации версии
0x0C	Регистр BIST, типа заголовка, таймера периода ожидания и размера строки кэша
0x10	Регистр базового адреса
0x34	Регистр указателя характеристик
0x3C	Регистр Max_Lat, Min_Gnt, вывода прерывания и строки прерывания

Регистр идентификации устройства и идентификации поставщика

Таблица 4.21.2 – Регистр идентификации устройства и идентификации поставщика (смещение адреса 0x00)

31 1615 0

Device ID	Vendor ID
0x0000	0x0000
r	r

31–16 Идентификатор устройства (Device ID)
15–0 Идентификатор поставщика (Vendor ID)

Регистр статуса и управления

Таблица 4.21.3 – Регистр статуса и управления (смещение адреса 0x04)

31	30	29	28	27	26 25	24	23	22	21	20	19	18 16
DPE	SSE	RMA	RTA	STA	DEV SEL timing	MDPE	FBBC	–	66 MHz	CL	IS	–
0	0	0	0	0	0x1	0	0	0	1	1	0	0x0
wc	wc	wc	wc	wc	r	r	r	r	r	r	r	r

15 11	10	9	8	7	6	5	4	3	2	1	0
–	ID	Not Imp	SE	–	PER	Not Imp	MWI	Not Imp	BM	MS	IOS
0x00	0	0	0	0	0	0	0	0	0	0	0
r	rw	r	rw	r	rw	r	rw	r	rw	rw	rw

31	Обнаружена ошибка по четности
30	Сигнализирована системная ошибка
29	Получено сообщение о прекращении работы ведущего устройства
28	Получено сообщение о прекращении работы ведомого устройства
27	Сигнализировано о прекращении работы ведомого устройства
26, 25	Временные настройки DEVSEL, «01» = среднее значение DEVSEL
24	Ошибка по четности данных ведущего устройства
23	Флаг быстрого режима обмена данными «вплотную».
22	Зарезервировано
21	Флаг отображения возможности работы на 66 МГц
20	Список характеристик
19	Статус прерывания
18–11	Зарезервировано
10	Запрещение прерывания
9	НЕ РЕАЛИЗОВАНО
8	Разрешение SERR#
7	Зарезервировано
6	Ответ «ошибка по четности»
5	НЕ РЕАЛИЗОВАНО
4	Запись памяти и разрешение инвалидации
3	НЕ РЕАЛИЗОВАНО
2	Разрешение ведущему генерировать PCI доступы
1	Разрешение устройству отвечать на доступы к области памяти.
0	Разрешение устройству отвечать на доступы к области ввода-вывода

Регистр кода класса и идентификации версии

Таблица 4.21.4 – Регистр кода класса и идентификации версии (смещение адреса 0x08)

31 8	7 0
Код класса	ID версии
0x000000	0x00
r	r

31–8	Код класса
7–0	Идентификатор версии

Регистр BIST, типа заголовка, таймера периода ожидания и размера строки кэша

Таблица 4.21.5 – Регистр BIST, типа заголовка, таймера периода ожидания и размера строки кэша (смещение адреса 0x0C)

31 24	23 16	15 8	7 0
BIST	Тип заголовка	Таймер задержки	Размер строки кэша
0x00	0x00	0x00	0x00
r	r	rw	r

31–24 НЕ РЕАЛИЗОВАНО, читается, как «0»
23–16 Тип заголовка, читается, как «0»
15–8 Таймер периода ожидания, все биты доступны для записи
7–0 НЕ РЕАЛИЗОВАНО, читается, как «0»

Регистр базового адреса

Таблица 4.21.6 – Регистр базового адреса (смещение адреса 0x10)

31 12	11 4	3	2 1	0
Базовый адрес	–	PF	Type	MS
0x0000000	0	*	0x0	0
rw	r	r	r	r

31–12 Регистр базового адреса PCI (BAR).
11–4 НЕ РЕАЛИЗОВАНО, читается, как «0»
3 Предварительная выборка. Если предварительная выборка отсутствует, читается, как «0»
2, 1 Тип, читается, как «0»
0 Индикатор области памяти

Регистр указателя характеристик

Таблица 4.21.7 – Регистр указателя характеристик (смещение адреса 0x34)

31 8	7 0
–	Указатель характеристик
0x000000	0x40
r	r

31–8 Зарезервировано
7–0 Указатель характеристик. Указывает на первый пункт списка характеристик расширенной области конфигурации PCI

Регистр Max_Lat, Min_Gnt, вывода прерывания и строки прерывания

Таблица 4.21.8 – Регистр Max_Lat, Min_Gnt, вывода прерывания и строки прерывания (смещение адреса 0x3C)

31 24	23 16	15 8	7 0
Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line
0x00	0x00	0x00	0x00
r	r	r	rw

31–24 НЕ РЕАЛИЗОВАНО, читается, как «0»
23–16 НЕ РЕАЛИЗОВАНО, читается, как «0»

15–8 Вывод прерывания – читается, как «0» (только для считывания)
7–0 Строка прерывания

Расширенная область конфигурации PCI

В данном подпункте рассматриваются регистры в расширенной области конфигурации PCI. Эти регистры реализуют отображение адресов PCI в адреса на шине AMBA и опцию изменения порядка следования байт шины PCI.

Регистры, реализованные в данном диапазоне адресов AMBA, соответствуют спецификации локальной шины PCI 2.2, (см. таблицы 4.21.9 – 4.21.15). Базовая часть смещения адреса PCI задается значением регистра указателя характеристик.

Таблица 4.21.9 – Регистры расширенной области конфигурации PCI (базовый адрес 0xFFF90000)

Смещение адреса PCI	Регистр
0x40 + 0x00	Регистр размера, следующего указателя, идентификации
0x40 + 0x04	Регистр отображения PCI BAR в адрес AHB
0x40 + 0x1C	Регистр отображения расширенной области конфигурации PCI в адрес AHB
0x40 + 0x20	Регистр базового адреса области ввода-вывода AHB и конфигурации шины PCI
0x40 + 0x24	Регистр размера PCI BAR и предварительной выборки
0x40 + 0x3C	Регистр границы пакета предварительной выборки ведущего устройства AHB

Регистр размера, следующего указателя, идентификации

Таблица 4.21.10 – Регистр длины, следующего указателя, идентификатора (смещение адреса 0x40)

31 24	23 16	15 8	7 0
–	Длина	Следующий указатель	Capability ID
0x00	0x40	0x00	0x09
r	r	r	r

31–24 Зарезервировано
23–16 Длина, читается, как 0x40 (только для считывания).
15–8 Указатель на следующий пункт списка характеристик. Значение 0x0 (только для считывания)
7–0 Идентификатор характеристик. Читается, как 0x09, идентифицируя поставщика (только для считывания)

Регистр отображения PCI BAR в адрес AHB

Таблица 4.21.11 – Регистр отображения PCI BAR в адрес AHB (смещение адреса 0x44)

31 12	11 0
Отображение PCI BAR в адрес AHB	–
0x00000	0x000
rw	r

31–12 32-битовый регистр отображения адреса для PCI BAR0. Трансляция доступа PCI BAR в базовый адрес AHB. Размер PCI BAR определяет количество доступных для использования бит.
11–0 Зарезервировано, считывается как «0»

Регистр отображения расширенной области конфигурации PCI в адрес AHB

Таблица 4.21.12 – Регистр отображения расширенной области конфигурации PCI в адрес AHB (смещение адреса 0x5C)

31 8	7 0
Регистр отображения расширенной области	–
0x000000	0x00
rw	r

31–8 Трансляция доступа к расширенной области конфигурации PCI (кроме диапазона адресов внутренних регистров, расположенных в данной области конфигурации) в адрес AHB

7–0 Зарезервировано

Регистр базового адреса области ввода-вывода AHB и конфигурации шины PCI

Таблица 4.21.13 – Регистр базового адреса области ввода-вывода AHB и конфигурация шины PCI (регистр порядка следования) (смещение адреса 0x60)

31 20	19 2	1	0
AHB IO базовый адрес	–	DISEN	Endian
0xFFF	0x00000	1	0
r	r	rw	rw

31–20 Базовый адрес области ввода-вывода AHB (только для считывания)

19–2 Зарезервировано

1 Разрешение сброса тайм-аута доступа ведомого устройства. Если установлено, ведомый будет сбрасывать ожидающий доступ, если в ходе выполнения циклов тактирования PCI 2^{15} не обнаружен повтор доступа (не дублируется для каждой функции PCI)

0 Изменение порядка следования байтов на шине PCI. 1: порядок следования «сначала младший байт» на шине PCI, 0: порядок следования «сначала старший байт» на шине PCI. Значение после сброса – 0

Регистр размера PCI BAR и предварительной выборки

Таблица 4.21.14 – Регистр размера PCI BAR и предварительной выборки (смещение адреса 0x64)

31 12	11 4	3	2 1	0
Маска размера PCI BAR	–	Pre	–	Type
0xFC000	0x00	0	0x0	0
rw	r	rw	r	rw

31–12 Регистр маски размера каждого PCI BAR. Если бит[n] установлен, бит[n] в регистре PCI BAR реализован, и выполняется возврат ненулевого значения. Все биты, начиная с младшего установленного бита до бита 31, должны быть установлены. Если бит 31 сброшен, данный PCI BAR не разрешен

11–4 Зарезервировано

3 Бит предварительной выборки в регистре PCI BAR

2, 1 Зарезервировано

0 Тип BAR. 0 = BAR памяти, 1 = BAR области ввода-вывода

Регистр границы пакета предварительной выборки ведущего устройства AHB

Таблица 4.21.15 – Регистр границы пакета предварительной выборки ведущего устройства AHB (смещение адреса 0x7C)

31	30	16	15	0
SRF	–		Длина пакета	
0	0x0000		0x0FFF	
rw	r		rw	

31	Сохранить FIFO чтения (SRF) Когда бит установлен, данные предварительной выборки FIFO будут сохраняться до следующего доступа PCI, когда ведомое устройство PCI прекращает доступ с отключением без данных
30–16	Зарезервировано
15–0	Максимальное количество тактов в пакете минус 1. (Максимальное значение 0xFFFF, что соответствует 0x10000 тактам, адресации 65 Кбайт)

4.21.3 Принцип работы

Поддержка доступа

Система поддерживает одиночный и пакетный доступ на шине AHB AMBA и на шине PCI. Дополнительная информация о поддерживаемых командах PCI приведена в пункте 4.21.5.

FIFO

Система имеет отдельные FIFO для каждого тракта: чтение ведомого устройства PCI, запись ведомого устройства PCI, считывание ведущего устройства PCI, запись ведущего устройства PCI, DMA из AHB в PCI и DMA из PCI в AHB (см. рисунок 4.21.2).

Разрешение байтов и слияние байтов (порядок следования)

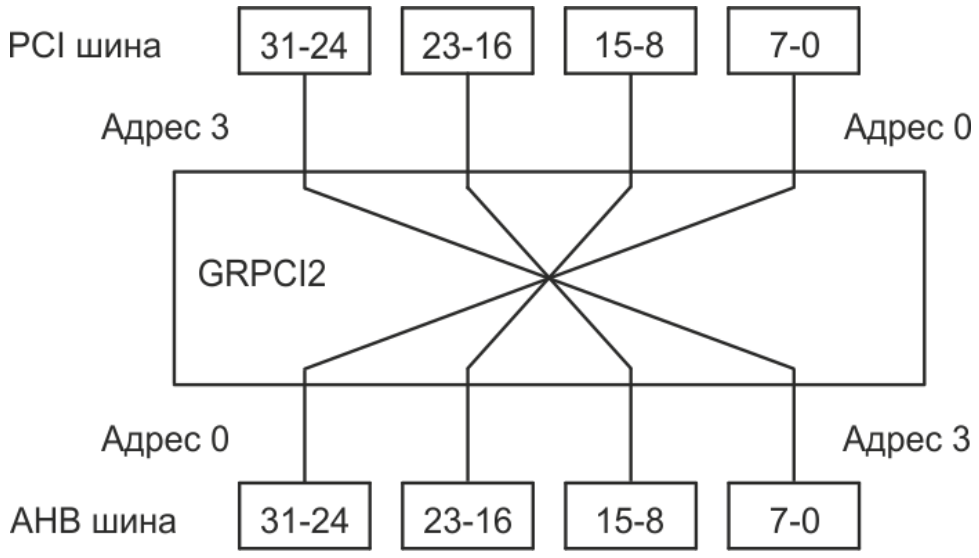


Рисунок 4.21.2 – Перестановка байтов

В таблице 4.21.16 представлены поддерживаемые адрес/размер AHB и комбинации разрешения байтов PCI.

Таблица 4.21.16 – Соответствие адреса/размера AHB комбинации разрешения байтов PCI

HSIZE AHB	ADDRESS[1:0] AHB	CBE[3:0] с порядком следования «сначала младший байт»	CBE[3:0] с порядком следования «сначала старший байт»
00 (8 бит)	00	1110	0111
00 (8 бит)	01	1101	1011
00 (8 бит)	10	1011	1101
00 (8 бит)	11	0111	1110
01 (16 бит)	00	1100	0011
01 (16 бит)	10	0011	1100
10 (32 бита)	00	0000	0000

Т.к. шина AHB в GRLIB определяется как шина, использующая порядок следования «сначала старший байт», система может определять шину PCI как шину, использующую порядок следования «сначала младший байт» (см. спецификацию локальной шины PCI) с помощью изменения порядка следования байт. Также система может определять шину PCI как шину, использующую порядок следования «сначала старший байт», без изменения порядка следования байт.

Порядок следования байт на шине PCI конфигурируется через расширенную область конфигурации PCI. Значение по умолчанию – «сначала младший байт».

Циклы конфигурации PCI

Доступ к области конфигурации PCI при установке порядка следования байт не меняется. Область конфигурации PCI всегда определяется порядком следования «сначала младший байт» (см. спецификацию локальной шины PCI). Это означает, что ведомый PCI не меняет байтовый порядок, даже если разрешено изменение порядка следования байт и ведущее устройство PCI всегда меняет порядок следования байт доступа области конфигурации PCI на порядок «сначала младший байт».

Данные, сохраненные в регистре области конфигурации PCI как 0x12345678 (бит[31:0]), передаются на шину AHB как 0x78563412 (бит[31:0]). Это означает, что не 8-битовый доступ к области конфигурации PCI необходимо конвертировать в программном обеспечении для получения правильного порядка следования байт.

Доступ к памяти и области ввода-вывода

Доступ к памяти и области ввода-вывода при изменении порядка следования байт меняется. Система определяет шину PCI с порядком следования «сначала младший байт» в первом случае, если система является хост-системой PCI и внешние периферийные устройства с порядком следования «сначала младший байт» выполняют передачи DMA в память хост-системы. Во втором случае – если система является внешним периферийным устройством и выполняет передачи DMA в хост-систему PCI с порядком следования «сначала младший байт».

Пакеты

Шина PCI: Ведомый PCI завершает пакет, если FIFO не доступны (ведущее устройство AHB AMBA не может быстро заполнить или освободить FIFO) или в ходе считывания, когда пакет достигает длины, указанной в регистре границы пакета предварительной выборки ведущего устройства AHB. Данный регистр определяет предельную границу пакета, т.е. если регистр содержит значение 0x400 бит (граница

адреса 4 Кбайта), устройство выполняет предварительную выборку данных только до этой границы, после чего завершает пакет с разъединением.

Ведущее устройство PCI останавливает пакет при переполнении работы таймера ожидания или в ходе считывания, когда пакет достигает длины, указанной в регистре границы пакета предварительной выборки ведущего устройства PCI (если ведущее устройство AHB, выполняющее доступ, не маскировано). Если ведущее устройство маскировано в данном регистре, устанавливается граница в 1 Кбайт. Ведущее устройство PCI не выполняет предварительную выборку данных, превышающих границу адреса.

Шина AHB: Поскольку FIFO доступны для записи и данные в FIFO доступны для считывания, ведомое устройство AHB не ограничивает длину пакета. Длина пакета для ведущего устройства AHB ограничивается емкостью FIFO. Ведущее устройство AHB выполняет пакет в пределах FIFO. Поддерживается только режим пакета с линейным инкрементом.

DMA: Регистр границы пакета предварительной выборки ведущего устройства AHB или регистр границы пакета предварительной выборки ведущего устройства PCI не влияют на доступ DMA.

Заполнение всех FIFO начинается при таком же смещении слова, что и доступ к шине (т.е. если емкость FIFO – 8 слов и стартовый адрес пакета – 0x4, первое слово данных сохраняется во второй записи FIFO, при этом в данном буфере FIFO сохраняется только семь слов).

Принцип работы хост-системы

Устройство содержит входной сигнал (PCI_HOST), который должен быть установлен (в активном низком логическом состоянии) для выполнения операций хост-системы PCI. Статус данного сигнала отображается в регистре статуса и характеристик, доступного через интерфейс ведомого устройства APB. Устройство генерирует циклы конфигурации PCI только при установленном сигнале (устройство является хост-системой).

В спецификации группы изготовителей промышленных компьютеров PCI (PCIMG) используется вывод C2 на коннекторе P2. Данный вывод должен иметь нагрузочные pull-up резисторы, т.к. слоты внешних периферийных устройств оставляют их неподсоединенными.

При установленном сигнале хост-системы ведомое устройство PCI выдает соответствующие ответы при выполнении циклов конфигурации, если сигналы IDSEL не установлены (AD[31:11] находятся в низком логическом состоянии). Это необходимо для конфигурации ведущим устройствам PCI своего ведомого PCI.

4.21.4 Интерфейс ведущего устройства PCI

Доступ к интерфейсу ведущего устройства PCI выполняется через интерфейс ведомого устройства AHB AMBA. Интерфейс ведомого устройства AHB занимает 256 Мбайт адресуемого пространства памяти AHB и 128 Кбайт адресуемого пространства области ввода/вывода AHB. Доступ к адресуемому пространству памяти AHB переводится в цикл памяти PCI. Доступ к первым 64 Кбайтам области ввода-вывода AHB переводится в цикл ввода/вывода PCI. Следующие 64 Кбайта переводятся в циклы конфигурации PCI.

Адресация:

- область пространства ввода-вывода 0xFFF80000 – 0xFFF8FFFF;
- область пространства конфигурации 0xFFF90000 – 0xFFF9FFFF;
- базовый адрес памяти ведущего устройства PCI 0xE0000000.

Циклы доступа к памяти

Единичный доступ считывания области памяти AHB транслируется в доступ считывания памяти PCI, считывание пакета транслируется в доступ комплексного считывания памяти PCI. Запись в данную область памяти транслируется в доступ записи PCI.

Трансляция определяется ведущим устройством AHB в регистрах отображения адреса PCI, доступ к которым выполняется через интерфейс ведомого устройства APB. Каждое ведущее устройство AHB на шине AHB AMBA имеет отдельный регистр отображения адреса. Эти регистры содержат старшие значащие биты адреса PCI.

Если ведущее устройство PCI выполняет передачу на шине PCI и не принимает новые запросы, интерфейс ведомого устройства AHB выдает ответ AMBA RETRY. Это выполняется в ходе чтения при извлечении ведущим устройством PCI запрашиваемых данных для заполнения FIFO считывания или в ходе записи, если FIFO записи не доступно.

Циклы доступа к области ввода-вывода

Доступ к младшим 64 Кбайтам адресуемого пространства области ввода-вывода AHB транслируется в циклы доступа к области ввода-вывода PCI. Трансляция адреса определяется регистром отображения адреса из AHB в PCI для области ввода-вывода PCI. В данном регистре устанавливается 16 старших значащих битов адреса PCI. Доступ к регистру отображения адреса из AHB в PCI для области ввода-вывода PCI выполняется через интерфейс ведомого устройства APB. Если бит IB (пакет области ввода-вывода PCI) в регистре управления (доступ к которому выполняется через интерфейс ведомого устройства APB) сброшен, ведущее устройство PCI не выполняет доступ к области ввода-вывода.

Циклы доступа к области конфигурации

Доступ к следующему 64 Кбайтовому блоку памяти (диапазон смещения адреса – от 64 Кбайт до 128 Кбайт) адресуемого пространства области ввода-вывода AHB транслируется в циклы доступа к области конфигурации PCI. Адрес AHB транслируется в адрес области конфигурации PCI, который отличается для циклов конфигурации PCI типа 0 и типа 1. Если в регистре управления поле номера шины (доступ к которому выполняется через интерфейс ведомого устройства APB) сброшено, используется тип 0 циклов конфигурации PCI. Если поле номера шины установлено, используется тип 1 циклов конфигурации PCI. Отображение адреса области ввода-вывода AHB в адрес области конфигурации для циклов конфигурации PCI типа 0 и типа 1 определяется в соответствии с таблицами Таблица 4.21.17 и 4.21.18.

Циклы доступа к области конфигурации PCI генерируются только хост-системой. Устройство содержит входной сигнал хост-системы (PCI_HOST), который должен быть установлен (в активном низком логическом состоянии) для операций хост-системы PCI. Статус данного сигнала отображается в регистре статуса и характеристик, доступ к которому выполняется через интерфейс ведомого устройства APB. Если бит CB (пакет конфигурации PCI) в регистре управления (доступ к которому выполняется через интерфейс ведомого устройства APB) сброшен, ведущее устройство PCI не выполняет пакетные доступы к области конфигурации.

Таблица 4.21.17 – Отображение адреса области ввода-вывода AHB в цикл конфигурации PCI, тип 0

31 16	15 11	10 8	7 2	1 0
AHB ADDRESS MSB	IDSEL	FUNC	REGISTER	BYTE

31–16	Старшие значащие биты адреса AHB: Не используется для отображения адреса цикла конфигурации PCI
15–11	IDSEL: Данное поле декодируется для управления AD[IDSEL+10] PCI. Предполагается, что каждый из сигналов AD[31:11] подключен (объединительной платой PCI) к одной соответствующей линии IDSEL.
10–8	FUNC: Выбор функции устройства
7–2	REGISTER: Используется для индексации DWORD PCI в области конфигурации
1, 0	BYTE: Используется для установки CBE в соответствующее состояние для доступов, которые не являются DWORD PCI

Таблица 4.21.18 – Отображение адреса области ввода-вывода AHB в цикл конфигурации PCI, тип 1

31 16	15 11	10 8	7 2	1 0
AHB ADDRESS MSB	DEVICE	FUNC	REGISTER	BYTE

31–16	Старшие значащие биты адреса AHB: Не используется для отображения адреса цикла конфигурации PCI
15–11	DEVICE: Выбор устройства для доступа на шине
10–8	FUNC: Выбор функции многофункционального устройства
7–2	REGISTER: Используется для индексации DWORD PCI в области конфигурации
1, 0	BYTE: Используется для установки CBE в соответствующее состояние для доступов, которые не являются DWORD PCI

Обработка ошибок

Если доступ считывания, выполняемый ведущим устройством PCI, завершается с остановкой работы ведомого или ведущего устройства, ведомое устройство AHB генерирует ответ AMBA ERROR при установленном бите ER в регистре управления. Если бит EI в регистре управления установлен, генерируется прерывание AMBA в связи с ошибкой. Причина ошибки указывается в поле статуса прерывания в регистре управления.

4.21.5 Интерфейс ведомого устройства PCI

Ведомое устройство PCI занимает в адресуемом пространстве PCI области памяти, соответствующие регистру BAR в области конфигурации PCI. Регистр статусной строки BAR0 определяет размещение адресов в адресуемом пространстве PCI. Размер каждого BAR устанавливается в регистре BAR и предварительной выборки, доступ к которому выполняется через расширенную область конфигурации PCI.

Поддерживаемые команды PCI

Ведомый PCI поддерживает следующие команды:

Считывание/запись конфигурации PCI: пакет и единичный доступ к области конфигурации PCI. Доступ, за исключением доступа, обозначенного в списке определяемых пользователем характеристик в расширенной области конфигурации PCI, не переводится в доступ на шине AHB AMBA.

Считывание памяти: команда считывания в памяти PCI BAR переводится в доступ единичного считывания на шине AHB AMBA.

Множественное считывание памяти, построчное считывание памяти: команда множественного считывания в памяти PCI BAR переводится в пакетный доступ на шине AHB AMBA. При выполнении доступа пакета выполняется предварительная выборка данных для заполнения максимального количества данных, сохраняемых в FIFO.

Запись памяти, запись памяти с инвалидацией: данные команды обрабатываются аналогично друг другу и передаются на шину AHB AMBA, как единичный доступ или пакетный доступ, в зависимости от размера доступа PCI (единичный доступ или пакетный доступ).

Чтение области ввода-вывода: команда чтения в области ввода-вывода PCI BAR переводится в доступ единичного чтения на шине AHB AMBA.

Запись области ввода-вывода: команда записи в область ввода-вывода PCI BAR переводится в доступ единичной записи на шине AHB AMBA.

Реализованные ответы PCI

Ведомый PCI блокирует доступ PCI, выдавая один из следующих ответов:

Повтор: ответ указывает, что ведомый либо извлекает данные для шины AHB AMBA в ходе считывания PCI, либо освобождает FIFO записи для записи PCI. Новый доступ чтения PCI будет всегда завершен с повтором, по крайней мере, однократно, перед тем, как PCI ведомый будет готов доставить данные.

Отключение с данными: завершение операции и передача данных на текущем этапе обработки данных. Ответ выдается, если ведущее устройство PCI запрашивает больше данных и следующее FIFO еще не доступно, или для пакетного доступа PCI с выполнением команды считывания памяти.

Отключение без данных: завершение операции без передачи данных на текущем этапе обработки данных. Ответ выдается при изменении CBE в ходе записи пакета PCI.

Прекращение работы ведомого устройства: указывает, что в текущем доступе произошла внутренняя ошибка, при этом ведомое устройство не в состоянии завершить доступ. Ответ выдается, если система получает сообщение об ошибке на шине AHB AMBA в ходе операции считывания.

Перевод PCI в AHB

Каждый регистр базового адреса PCI (BAR) имеет регистр перевода (отображения, трансляции) для перевода доступа PCI в адресуемое пространство AHB AMBA. Доступ к данному регистру отображения выполняется через расширенную область конфигурации PCI. Количество реализованных бит в данном регистре соответствует размеру (и количеству реализованных битов) регистров BAR.

Сигнал хост-системы PCI

При установленном сигнале хост-системы ведомое устройство PCI выдает соответствующие ответы при выполнении циклов конфигурации, если сигналы IDSEL не установлены (AD[31:11] не установлены). Это выполняется для того, чтобы ведущее устройство PCI, в позиции хост-системы, было способно сконфигурировать своего ведомого PCI.

Обработка ошибок

Ведомый PCI завершает доступ с прекращением работы, если на запрос данных шины AHB выдается сообщение об ошибке. Поскольку записи в ведомое устройство PCI

являются отправленными, при возникновении ошибок записи AHB сообщение об ошибке не выдается.

Если при завершении работы ведущего устройства PCI выдается ответ «повтор», данное ведущее устройство должно повторять выполнение данного доступа, пока доступ не завершится с прекращением работы ведомого. Если ведущее устройство не повторяет выполнение данного доступа, интерфейс ведомого устройства PCI блокируется на данном доступе и не принимает новый. Во избежание этого, ведомое устройство PCI может не учитывать доступ, если его повторное выполнение не происходит в течение 2^{15} циклов тактирования. Данный тайм-аут разрешается через регистр базового адреса области ввода-вывода AHB и конфигурации шины PCI, находящийся в расширенной области конфигурации PCI ядра.

4.21.6 Контроллер DMA

Механизм прямого доступа к памяти (DMA) построен на основе дескрипторов, реализована двухуровневая система дескрипторов.

Канал DMA

Первый уровень – связанный список дескрипторов каналов DMA. В каждом дескрипторе содержится указатель на список дескрипторов данных и указатель на следующий канал DMA. Дескриптор последнего канала DMA всегда должен указывать на первый дескриптор списка каналов DMA, образовывая замкнутый цикл. Дескриптор должен быть выровнен в памяти по границе в 4 слова (0x10) и иметь структуру, приведенную в таблицах 4.21.19 и 4.21.20.

Таблица 4.21.19 – Структура дескриптора канала DMA

Смещение адреса в дескрипторе	Слова дескриптора
0x00	Управление каналом DMA
0x04	Следующий канал DMA (32-битовый адрес, указывающий на следующий дескриптор канала DMA)
0x08	Следующий дескриптор данных в текущем канале DMA (32-битовый адрес, указывающий на следующий дескриптор данных)
0x0C	Зарезервировано

Таблица 4.21.20 – Управление канала DMA

31	30 – 25	24 – 22	21 – 20	19 – 16	15 – 0
EN	–	Идентификатор канала	Тип	–	Количество дескрипторов данных

31	Разрешение дескриптора канала
30–25	Зарезервировано
24–22	Идентификатор канала. Каждому каналу DMA требуется присвоение идентификатора для определения источника прерывания DMA
21, 20	Тип дескриптора. 01_2 = дескриптор канала DMA
19–16	Зарезервировано
15–0	Максимальное количество дескрипторов данных, выполняемых перед переходом в следующий канал DMA. «0» указывает, что все дескрипторы данных должны быть выполнены до перехода в следующий канал DMA

Количество разрешенных каналов DMA сохраняется в поле количества каналов DMA регистра управления DMA, доступного через интерфейс ведомого устройства APB.

Дескриптор данных

Второй уровень дескрипторов – связанный список передаваемых данных. Последний дескриптор данного списка должен быть запрещенным. Чтобы добавить новую передачу данных, запрещенный дескриптор обновляется чтобы отражать передачу данных и указывать на новый запрещенный дескриптор. Слово управления данного дескриптора должно обновляться последним, разрешая использовать корректный дескриптор. Для того чтобы быть уверенным, что механизм DMA считает этот новый дескриптор, необходимо обновить бит разрешения в регистре управления DMA. Дескриптор должен быть выровнен в памяти по границе в 4 слова (0x10) и иметь структуру, указанную в таблицах 4.21.21 и 4.21.22.

Таблица 4.21.21 – Структура дескриптора данных DMA

Смещение адреса в дескрипторе	Слова дескриптора
0x00	Управление данными DMA
0x04	32-битовый стартовый адрес PCI
0x08	32-битовый стартовый адрес AHB
0x0C	Следующий дескриптор данных в текущем канале DMA (32-битовый адрес, указывающий на следующий дескриптор данных)

Таблица 4.21.22 – Управление данных DMA

31	30	29	28	27 – 22	21 20	19	18 – 16	15 – 0
EN	IE	DR	BE	–	Type	ER	–	LEN

31 Разрешение дескриптора данных
30 Разрешение генерации прерывания
29 Направление передачи. «0»: из PCI в AMBA, «1»: из AMBA в PCI
28 Изменение порядка следования байт на шине PCI. «1» определяет шину PCI с порядком следования «сначала младший байт» для данной передачи, «0» определяет шину PCI с порядком следования «сначала старший байт» для данной передачи
27–22 Зарезервировано (должно быть сброшено)
21, 20 Тип дескриптора. 00_2 = дескриптор данных DMA
19 Статус ошибки
18–16 Зарезервировано
15–0 Размер передачи. Количество слов в передаче: LEN + 1

Передача данных

Механизм DMA сначала считывает дескриптор первого канала DMA. Если канал DMA разрешен, считывается и выполняется первый дескриптор данных этого канала. После выполнения передачи дескриптор данных запрещается, статус записывается в слово управления. Если в ходе передачи ошибок не возникло, бит ошибки не устанавливается и поле размера передачи не меняется. Если передача завершена в связи с обнаружением ошибки, в слове управления устанавливается бит ошибки, в поле размера указывается место возникновения ошибки. Если ошибок не возникло, считывается и выполняется следующий дескриптор данных. После считывания неразрешенного дескриптора данных или после выполнения максимального количества дескрипторов данных в дескрипторе канала DMA обновляется указатель на следующий дескриптор данных, а механизм DMA переходит к следующему каналу DMA.

При обнаружении ошибки или при отсутствии разрешенных дескрипторов данных механизм DMA останавливается. Тип ошибки указывается битами 7 – 11 в регистре

управления DMA. Биты типа ошибок должны быть сброшены (через запись «1») перед повторным разрешением DMA.

Прерывание

Контроллер DMA имеет бит разрешения прерывания в регистре управления DMA (доступ к которому выполняется через интерфейс ведомого устройства APB), который разрешает генерацию прерывания.

Каждый дескриптор данных имеет бит разрешения прерывания, определяющий необходимость генерации прерывания системой после выполнения дескриптора.

Механизм DMA использует ту же линию прерываний, что и весь остальной модуль интерфейса PCI, как это указано в поле режима запроса на прерывание (IRQ mode) в регистре статуса и характеристик.

4.21.7 Прерывания

Система выполняет выборку сигналов INTA-D PCI и направляет прерывание на шину APB. Сигналы INTA-D PCI соединяются с одним сигналом запроса на прерывание APB. Поле маски INT хост-системы в регистре управления используется только для выборки допустимых сигналов INT PCI.

Также доступен бит принудительного прерывания PCI в регистре управления для принудительной передачи установленного сигнала INT PCI. Когда приходит сигнал PCI ошибка системы (SERR), устанавливается бит ошибки системы в поле статуса прерывания регистра статуса. Если прерывания разрешены, генерируется прерывание на шине APB.

Номер генерируемого прерывания – 6.

4.21.8 Сброс

В модуле используется синхронный сброс, который сбрасывает подмножество своих внутренних регистров, принадлежащих к системному тактовому домену.

Деактивация сигнала сброса PCI синхронизирована с тактовым сигналом PCI и осуществляется с задержкой в три такта.

Когда модуль функционирует в режиме системного хоста, сигнал сброса AHB задается сигналом сброса PCI.

4.21.9 Арбитраж шины PCI

Для арбитража запросов и подтверждений четырех агентов на шине PCI используется алгоритм циклического обхода (round-robin) с двумя различными уровнями приоритетов, с вложением одного в другой.

Алгоритм арбитража

Алгоритм обхода соответствует содержимому секции 3.4 стандарта PCI (PCI Local Bus Specification, Revision 2.2). Предоставление шины осуществляется циклическим обходом двух петель приоритета, одна из которых вложена в другую. Каждый агент имеет свой номер и уровень приоритета. Приоритет агента задается через регистр управления приоритетами («0» – высокий, «1» – низкий). Номер агента соответствует паре сигналов PCI_ARB_REQ(n)# / PCI_ARB_GNT(n)# (присвоение номеров осуществляется в обратном порядке, за пару PCI_ARB_REQ(0)# / PCI_ARB_GNT(0)# отвечает агент 3). На

рисунке 4.21.3 приведен пример для агентов с номерами A, B уровня 0 и номерами X, Y уровня 1. По сбросу все агенты имеют приоритет уровня 1.

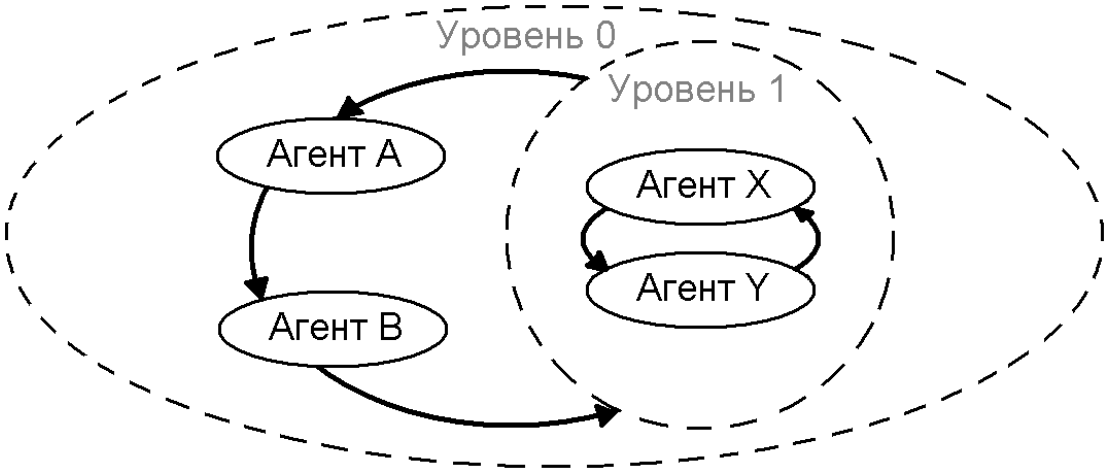


Рисунок 4.21.3 – Пример вложенного циклического арбитража

Все агенты одного уровня имеют равный доступ к шине (последовательное циклическое предоставление ресурсов). Все агенты с уровнем приоритета 1, как группа, имеют доступ равнозначный доступу каждого отдельного агента с уровнем 0. Изменение арбитража происходит, когда сигнал установлен, как только другой мастер запрашивает управление шиной, но только один раз за транзакцию.

Таймаут «неисправный мастер» также может стать причиной изменения арбитража (согласно секции 3.4.1 стандарта). Разрешение отзывается у агента, который еще не начал цикл с 16-ю циклами после запроса и разрешения. Информирование о таком «неисправном» мастере не реализовано.

Цикл перехода (turnover cycle) требуется по стандарту, когда изменение арбитража происходит во время состояния простоя шины. Вопреки стандарту, состоянием простоя шины считается факт нахождения PCI_FRAME# в высоком логическом состоянии более одного цикла.

После сброса шина устанавливается в позицию ожидания (bus parking) агента 0. Шина остается предоставленной последнему владельцу, если никакой другой агент не запрашивает шину. Когда выставлен очередной запрос, изменение арбитража происходит после одного цикла перехода.

Опциональная блокировка ресурсов шины (lock), упомянутая в стандарте PCI, не принимается во внимание модулем арбитра. То есть ситуация, когда сигнал PCI_LOCK# активен, никаким особым образом не обрабатывается.

Управление задержками в PCI реализовано через счетчики задержки каждого агента. Арбитр никак не контролирует задержки и единожды разрешенный агент продолжает транзакцию до того момента, как его разрешение аннулируется и его собственный счетчик задержки завершит счет. Даже если во время транзакции происходит изменение арбитража, действительная передача управления будет осуществлена, только когда текущий владелец шины уберет активный уровень сигнала с вывода PCI_FRAME#.

Регистр управления приоритетами

По адресу 0x80000880 доступен регистр управления приоритетом обхода. Его управляющие поля приведены в таблице 4.21.23. Уровень приоритета («0» – высокий, «1» – низкий) назначается каждому устройству. Исключением является агент 3, который всегда имеет самый низкий приоритет.

Таблица 4.21.23 – Регистр арбитража приоритетов PCI

31 – 4	3	2	1	0
–	P3	P2	P1	P0
0x0000000	1	1	1	1
r	r	rw	rw	rw

31 – 4 Зарезервировано

3 Приоритет линии 3 (P3) – Бит установлен, линия 3 (PCI_ARB_REQ(0)# и PCI_ARB_GNT(0)#) по умолчанию имеет самый низкий приоритет при выполнении обхода

2 Приоритет линии 2 (P2) – Если бит установлен, то приоритет линии 2 (PCI_ARB_REQ(1)# и PCI_ARB_GNT(1)#) при выполнении обхода низкий, если сброшен – высокий

1 Приоритет линии 1 (P1) – Если бит установлен, то приоритет линии 1 (PCI_ARB_REQ(2)# и PCI_ARB_GNT(2)#) при выполнении обхода низкий, если сброшен – высокий

0 Приоритет линии 0 (P0) – Если бит установлен, то приоритет линии 0 (PCI_ARB_REQ(3)# и PCI_ARB_GNT(3)#) при выполнении обхода низкий, если сброшен – высокий

4.21.10 Регистры

Модуль конфигурируется через регистры, размещенные на шине APB, представленные в таблицах 4.21.24 – 4.21.33.

Таблица 4.21.24 – Регистры управления модулем PCI (базовый адрес 0x80000400)

Смещение адреса APB	Регистр
0x00	Регистр управления
0x04	Регистр статуса и характеристик
0x08	Регистр границы пакета предварительной выборки ведущего устройства PCI
0x0C	Регистр отображения доступов AHB в доступы PCI для области ввода-вывода PCI
0x10	Регистр управления и статуса DMA
0x14	Регистр базового адреса (BAR) дескриптора DMA
0x18	Регистр активного канала DMA
0x1C	Зарезервировано
0x20 – 0x34	Регистр базового адреса (BAR) отображения доступа PCI в доступ AHB
0x38 – 0x3C	Зарезервировано
0x40 – 0x7C	Регистр отображения адреса доступа ведущего устройства AHB в адрес доступа к области памяти PCI

Регистр управления

Таблица 4.21.25 – Регистр управления, адрес 0x80000400

31	30	29	28	27	26	25	24	23 … 16
RE	MR	TR	–	SI	PE	ER	EI	Номер шины
0	0	0	0	0	0	0	0	0x00
rw	rw	rw	r	rw	rw	rw	rw	rw

15 … 11	10	9	8	7 … 4	3 … 0
–	IB	CB	DIF	Маска прерывания устройства	Маска прерывания хоста
0x00	0	0	0	0x0	0x0
r	rw	rw	rw	rw	rw

31 Сброс PCI. Если бит установлен, устанавливается сигнал сброса PCI. Для снятия установки сигнала сброса PCI поле должно быть очищено

30 Сброс ведущего устройства PCI. Устанавливается программно для сброса ведущего устройства PCI. Сбрасывается бит аппаратно автоматически.

29 Сброс ведомого устройства PCI. Устанавливается программно для сброса ведомого устройства PCI. Сбрасывается бит аппаратно автоматически.

28 Зарезервировано

27 Если бит установлен, разрешено прерывание в связи с ошибкой системы (SERR).

26 Если бит установлен, разрешен ответ об ошибке AHB в связи с ошибкой по четности.

25 Если бит установлен, разрешен ответ об ошибке AHB в связи с прекращением работы ведущего и ведомого устройств.

24 Если бит установлен, разрешено прерывание в связи с прекращением работы ведущего и ведомого устройств в связи с ошибкой по четности.

23–16 Если бит установлен, генерируются циклы конфигурации типа 1. Данное поле также используется, как поле номера шины для циклов конфигурации типа 1.

15–11 Зарезервировано

10 Если бит установлен, ведущим устройством PCI может быть сгенерирован пакетный доступ для циклов обращения к области ввода-вывода PCI.

9 Если бит установлен, ведущим устройством PCI может быть сгенерирован пакетный доступ для циклов обращения к области конфигурации PCI.

8 Принудительное прерывание устройства. Если бит установлен, выполняется принудительное прерывание PCI.

7–4 Маска прерывания устройства. Если установлен бит[n], dirq[n] не маскировано.

3–0 Маска прерывания хост-системы:
бит[3] = 1: INTD не маскировано
бит[2] = 1: INTC не маскировано
бит[1] = 1: INTB не маскировано
бит[0] = 1: INTA не маскировано

Регистр статуса и характеристик

Таблица 4.21.26 – Регистр статуса и характеристик, адрес 0x80000404

31	30	29	28	27	26	25 24	23	22 21	20	19
Host	MST	TAR	DMA	DI	HI	IRQ mode	Trace	–	CFGDO	CFGER
*	1	1	1	*	*	0x0	0	0x0	0	0
r	r	r	r	r	r	r	r	r	r	r

18 12	11 8	7 5	4 2	1 0
Статус прерывания устройства	Статус прерывания хост-системы	–	FDEPTH	FNUM
*	*	0x0	0x3	0x2
wc	r	r	r	r

31 Если бит сброшен, устройство вставлено в системный слот и работает, как хост-система.

30 Реализован режим ведущего устройства

29 Реализован режим ведомого устройства

28 Реализован DMA

27 Устройство управляет INTA PCI

26 Устройство выполняет выборку PCI INTA…D (для операций хост-системы)

25 – 24 Режим запроса на прерывание APB – 00_2.
00_2: PCI INTA…D, прерывание в связи с ошибкой и прерывание DMA при получении одного и того же сигнала запроса на прерывание IRQ
01_2: PCI INTA…D и прерывание в связи с ошибкой при получении одного и того же сигнала запроса на прерывание. Прерывание DMA при получении сигнала запроса на прерывание IRQ + 1
10_2: PCI INTA…D на прерывание IRQ…IRQ + 3. Прерывание в связи с ошибкой и прерывание DMA при получении сигнала запроса на прерывание
11_2: PCI INTA…D на прерывание IRQ…IRQ + 3. Прерывание в связи с ошибкой при получении сигнала запроса на прерывание. Прерывание DMA при получении сигнала запроса на прерывание IRQ + 4

23 Буфер трассировки PCI не реализован

22 – 21 Зарезервировано

20 Доступ конфигурации PCI выполнен, статус ошибки конфигурации PCI достоверен

19 Ошибка в ходе выполнения доступа конфигурации PCI

18 – 12 Статус прерывания:
бит[6]: доступ ведомого PCI не учитывается в связи с тайм-аутом (повторный доступ не выполнен в ходе 2^{15} циклов тактирования PCI)
бит[5]: ошибка системы
бит[4]: прерывание DMA
бит[3]: ошибка DMA
бит[2]: прекращение работы ведущего устройства
бит[1]: прекращение работы ведомого устройства
бит[0]: ошибка по четности

11 – 8 Статус прерывания хост-системы:
бит[3] = 0: указывает, что INTD установлено
бит[2] = 0: указывает, что INTC установлено
бит[1] = 0: указывает, что INTB установлено
бит[0] = 0: указывает, что INTA установлено

7 – 5 Зарезервировано
4 – 2 Количество слов в каждом FIFO равно 8 (2^{FIFO})
1, 0 Количество FIFO равняется 2

Регистр границы пакета предварительной выборки ведущего устройства PCI

Таблица 4.21.27 – Регистр границы пакета предварительной выборки ведущего устройства PCI, адрес 0x80000408

31 16	15 8	7 0
Немаскированный AHB ведущий	–	Длина пакета
0x0000	0x00	0xFF
rw	r	rw

31–16 Если установлен бит[n], пакет предварительной выборки ведущего устройства AHB n ограничен полем размера пакета
15–8 Зарезервировано
7–0 Максимальное количество данных (минус 1) в пакете (максимальное значение 0xFF соответствует 0x100 данных, что, в свою очередь, соответствует 1-Кбайтовому адресу)

Регистр отображения доступов AHB в доступы PCI для области ввода-вывода PCI

Таблица 4.21.28 – Регистр отображения доступов AHB в доступы PCI для области ввода-вывода PCI, адрес 0x8000040C

31 16	15 0
AHB to PCI IO	–
0x0000	0x0000
rw	r

31–16 Используются, как старшие значащие биты базового адреса для доступа к области ввода-вывода PCI.
15–0 Зарезервировано

Регистр управления и статуса DMA

Таблица 4.21.29 – Регистр управления и статуса DMA, адрес 0x80000410

31	30 20	19 12	11	10	9	8	7	6 4	3	2	1	0
SAFE	–	CHIRQ	MA	TA	PE	AE	DE	Количество каналов DMA	ACTIVE	DIS	IE	EN
1	0x000	0x00	0	0	0	0	0	0x0	0	0	0	0
rw	r	wc	wc	wc	wc	wc	wc	rw	r	rw	rw	rw

31 Обеспечение безопасности для обновления полей управления. Должно быть установлено для обновления полей управления.
30–20 Зарезервировано
19–12 Статус прерывания канала. Устанавливается, если дескриптор сконфигурирован генерировать сигнал прерывания. Бит[0] соответствует каналу с идентификатором 0, бит[1] соответствует каналу с идентификатором 1 и т.д. Биты сбрасываются при записи «1».
11 Статус прекращения работы ведущего устройства во время доступа PCI. Бит сбрасывается при записи «1».
10 Статус прекращения работы ведомого во время доступа PCI. Бит сбрасывается при записи «1».

9	Ошибка по четности во время доступа PCI. Бит сбрасывается при записи «1».
8	Ошибка во время доступа к данным AHB. Бит сбрасывается при записи «1».
7	Ошибка во время доступа к дескриптору. Бит сбрасывается при записи «1» .
6–4	Количество каналов DMA (биты защищены от перезаписи битом [31]).
3	DMA активен (только для считывания).
2	Запрещение/остановка DMA. При записи «1» в данный бит DMA запрещается.
1	Разрешение прерывания (бит защищен от перезаписи битом [31]).
0	Разрешение/запуск DMA. При записи «1» в данный бит DMA разрешается

Регистр базового адреса (BAR) дескриптора DMA

Таблица 4.21.30 – Регистр базового адреса дескриптора DMA, адрес 0x80000414

31	0
Базовый адрес дескриптора DMA	
0x00000000	
rw	

31–0	Базовый адрес таблицы дескрипторов DMA. Когда DMA запущен, данный регистр указывает на активный дескриптор

Регистр активного канала DMA

Таблица 4.21.31 – Регистр активного канала DMA, адрес 0x80000418

31	0
Базовый адрес активного канала DMA	
0x00000000	
rw	

31–0	Базовый адрес активного канала DMA

Регистр базового адреса (BAR) отображения доступа PCI в доступ AHB

Таблица 4.21.32 – Регистр базового адреса отображения доступа PCI в доступ AHB, адреса 0x80000420 – 0x80000434

31	0
PCI to AHB	
0x00000000	
rw	

31–0	32-битовый регистр отображения для каждого PCI BAR. Трансляция адреса доступа к PCI в базовый адрес AHB.

Регистр отображения адреса доступа ведущего устройства AHB в адрес доступа к области памяти PCI

Таблица 4.21.33 – Регистр отображения адреса доступа ведущего устройства AHB в адрес доступа к области памяти PCI, адреса 0x80000440 – 0x8000047C

31	0
AHB to PCI	
0x00000000	
rw	

31–0 32-битовый регистр отображения для каждого ведущего устройства AHB. Переводит доступ из соответствующего ведущего устройства AHB в базовый адрес PCI. Размер адресуемого пространства ведомого устройства AHB определяет количество реализованных битов (начиная с бита 31). Нереализованные биты сбрасываются. Регистр отображения для ведущего устройства AHB 0 расположен при смещении 0x40, для ведущего устройства AHB 1 – при смещении 0x44 и т.д. до ведущего устройства AHB 15 – при смещении 0x7C. Регистры отображения реализованы только для имеющихся ведущих устройств AHB

4.22 Контроллер шины AHB

4.22.1 Общие сведения

Согласно стандарту AMBA 2.0, контроллер AHB AMBA (AHBCTRL) – это совмещенный арбитр AHB, мультиплексор шины и декодер подчиненных компонентов, как показано на рисунке 4.22.1.

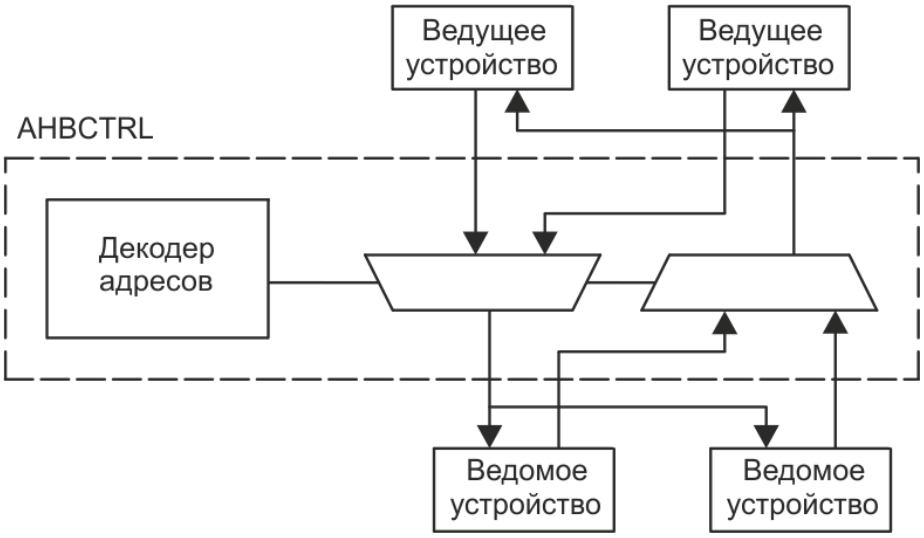


Рисунок 4.22.1 – Блок-схема контроллера AHB

4.22.2 Принцип работы

Арбитраж (контроль доступа к общей шине)

В контроллере AHB реализован алгоритм арбитража с последовательным циклическим предоставлением ресурсов (round-robin). Приоритет сдвигается на один шаг

после каждой передачи AHB. Если ни одно ведущее устройство не запрашивает шину, она остается закрепленной за последним владельцем шины.

Декодирование

Декодирование (генерация HSEL) подчиненных устройств AHB осуществляется через использования технологии plug & plug (см. Приложение Б). Ведомое устройство может занять любое двоичное выровненное адресуемое пространство размером (1 - 4096) Мбайт. Также декодируется особая область ввода-вывода, в которой ведомые устройства могут занимать от 256 байт до 1 Мбайта. Адрес области ввода-вывода по умолчанию – 0xFFF00000. Доступ к неиспользуемым адресам вызовет ошибку шины AHB.

Область ввода-вывода может быть помещена в область памяти, занятую ведомым устройством. Ведомое устройство не будет выбрано при доступе к области ввода-вывода.

Информация plug & play

Устройства содержат ряд информационных слов plug & plug как часть записей AHB, которыми они управляют на шине (для получения дополнительной информации – см. Приложение Б). Эти записи объединены в массив данных, подключенный к модулю контроллера AHB.

Данные plug & plug отображаются в область только для чтения. По умолчанию область отображения приходится на адреса 0xFFFFF000 – 0xFFFFFFFF. Данные по ведущим устройствам находятся в первых 2 Кбайтах блока (0xFFFFF000 – 0xFFFFF800), в то время как данные по ведомым устройствам – во втором 2-Кбайтовом блоке. Каждый сегмент занимает 32 байта, как показано на рисунке 4.22.2, следовательно, в адресуемом пространстве имеется место для 64 ведущих устройств и 64 ведомых устройств. Адрес данных plug & plug конкретного сегмента определяется его индексом на шине. Таким образом, адрес для ведущих устройств – 0xFFFFF000 + (n × 32), и для ведомых устройств – 0xFFFFF800 + (n × 32).

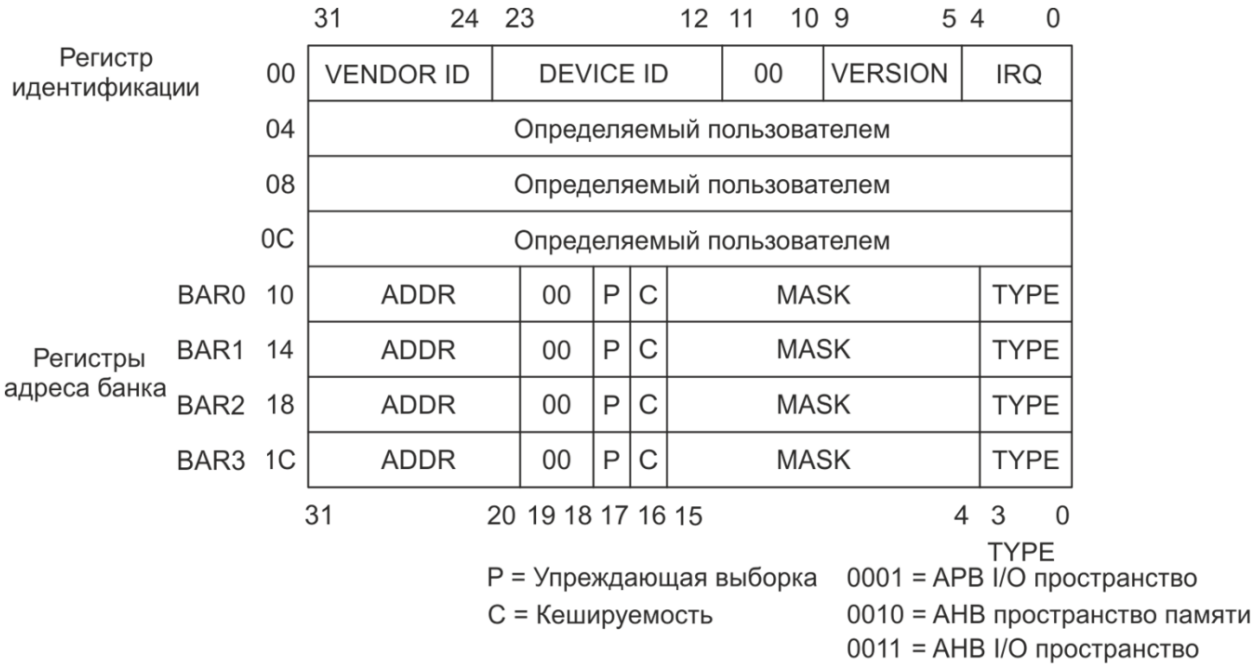


Рисунок 4.22.2 – Формат AHB записи plug & plug

4.23 Контроллер шины APB

4.23.1 Общие сведения

Согласно стандарту AMBA 2.0 мост AHB/APB AMBA (APBCTRL) является устройством управления шиной APB, как показано на рисунке 4.23.1. Контроллер поддерживает до 16 подчиненных устройств.

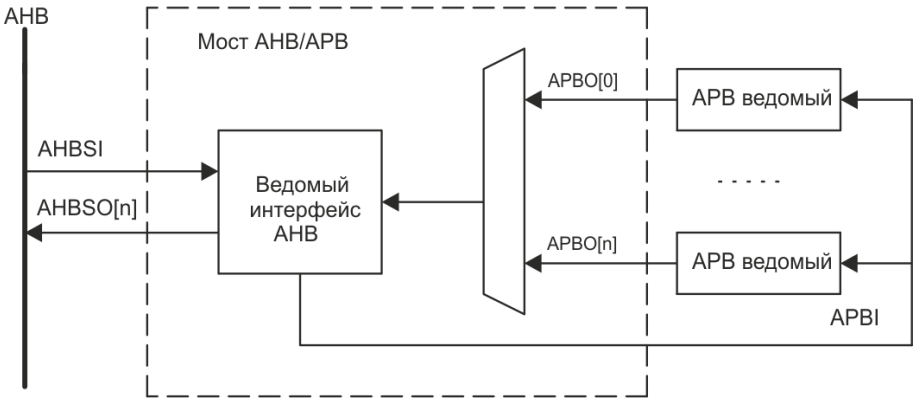


Рисунок 4.23.1 – Блок-схема моста AHB/APB

4.23.2 Принцип работы

Декодирование

Распознавание (генерация PSEL) ведомых устройств AHB осуществляется через использования технологии plug & plug (см. Приложение Б). Ведомое устройство может занимать любое двоичное выровненное адресуемое пространство размером 256 байт – 1 Мбайт. Записи в область, которая не определена, игнорируются, в то время как при считывании из неопределенных областей будет получено произвольное значение. Ни в одном из случаев не генерируется ошибка AHB.

Информация plug & plug

Ведомые устройства APB содержат по два информационных слова plug & plug, которые составляют APB запись, которые и приведены на шине. Эти записи объединены в массив данных, подключенный к мосту APB.

Данные plug & plug отображаются в область только для чтения, которая располагается в старших 4 Кбайтах адресуемого пространства моста. На рисунке 4.23.2 показано, что каждый блок plug & plug занимает 8 байт. Адрес информации plug & plug для определенного модуля определяется его индексом на шине. Если за мостом закреплен адрес 0x80000000 на шине AHB, то адреса записей plug & plug определяются следующим образом – 0x800FF000 + (n × 8).

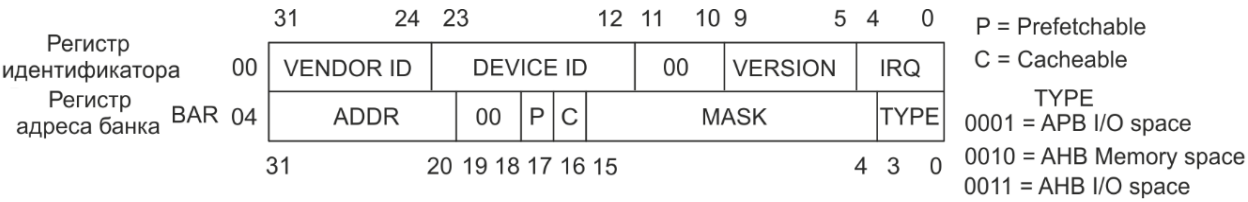


Рисунок 4.23.2 – Формат APB записи plug & plug

4.24 Цепь граничного сканирования

Для проведения тестирования выводов в процессорах 1906ВМ016 и 1906ВМ01А6 предусмотрен режим граничного сканирования. Управление режимом осуществляется с помощью TAP контроллера в соответствии со стандартом IEEE 1149.1. В таблице 4.24.1 приведено функциональное соответствие выводов цепи граничного сканирования.

Таблица 4.24.1 – Функциональное соответствие выводов

Вывод кристалла	Тестовое назначение
JTAG_TDI	TDI
BSD_TDO	TDO
BSD_TRST#	TRST
JTAG_TMS	TMS
JTAG_TCK	TCK

В рабочем режиме модуль управления граничным сканированием должен быть переведен в состояние логического 0, схема контроллера под сбросом. Для выполнения данного условия требуется внешняя притяжка BSD_TRST# к нулю. В таблице 4.24.2 приведены реализованные коды инструкций.

Таблица 4.24.2 – Коды инструкций

Инструкция	Код
CAPTURE	X01
EXTEST	100
SAMPLE	101
PRELOAD	101
BYPASS	111

4.25 Конфигурация процессоров 1906ВМ016, 1906ВМ01А6

Конфигурация процессоров 1906ВМ016, 1906ВМ01А6 приведена в таблицах 4.25.1 – 4.25.4.

Таблица 4.25.1 – Конфигурация ядра LEON4

Опция	Значение
Стартовый адрес	0x00000000
Количество ядер LEON4	1
Поддержка инструкций SPARC V8 MUL/DIV	да
Поддержка инструкций SPARC V8e UMAC/SMAC	да
Поддержка частичной записи WRPSR (SPARC V8e)	да
Поддержка инструкции CASA	да
Тип аппаратного умножителя	16x16
Задержка аппаратного умножителя	4 такта
Поддержка SVT (Single vector trapping)	да
Задержка загрузки	1
Количество аппаратных точек останова / контрольных точек данных (breakpoints/watchpoints)	2
Режим Power-down	да
Стартовый адрес по сбросу	0
Количество регистровых окон	8

Таблица 4.25.2 – Конфигурация кэшей

Опция	Значение
Количество уровней	1
Кэш инструкций	
Ассоциативность (количество каналов)	2
Размер канала (пути)	4 КБ
Размер строки	32 байта (8 слов)
Алгоритм замены по умолчанию	LRU
Блокировка кэша	нет
Кэш данных	
Ассоциативность (количество каналов)	2
Размер канала (пути)	4 КБ
Размер строки	16 байт (4 слова)
Алгоритм замены по умолчанию	LRU
Блокировка кэша	нет
Отслеживание шины AHB (AHB snooping)	да
Быстрое отслеживание (fast snooping)	да
Раздельные метки отслеживания (snoop tag)	да

Таблица 4.25.3 – Модуль управления памятью (MMU)

Опция	Значение
Тип MMU	раздельный (для инструкций и данных)
Алгоритм замены в TLB	LRU
Записей инструкций в TLB	16
Записей данных в TLB	16
Быстрый буфер записи	да

Таблица 4.25.4 – Отладочный модуль (DSU)

Опция	Значение
Размер буфера трассировки инструкций	1 КБ
Размер буфера трассировки шины AHB	4 КБ
Поддерживаемые интерфейсы	RS232, JTAG, Ethernet, SpaceWire

5 Описание архитектуры

5.1 Архитектура SPARC

5.1.1 Особенности архитектуры

SPARC – архитектура системы команд (ISA) ЦПУ, разработанная на базе архитектуры RISC (архитектуры с сокращенным набором команд). Архитектура SPARC обеспечивает целый спектр реализаций микросхем и систем с хорошим соотношением цены и качества и может применяться в различных областях, включая научно-техническую сферу, программирование, работу в режиме реального времени и коммерческую область.

Цели, которые преследовались при разработке

SPARC была спроектирована для эффективной работы оптимизирующих компиляторов и удобного внедрения аппаратной конвейеризации. Реализация SPARC обеспечивает исключительно высокий уровень производительности с максимально коротким сроком внедрения.

Регистровые окна

SPARC, разработанный в Sun Microsystems в 1985 году, основан на проектах RISC I и II, спроектированных в Калифорнийском университете в Беркли с 1980 по 1982 год. Архитектура SPARC «с регистровыми окнами», впервые введенная в проектах Калифорнийского университета в Беркли, позволяет использовать простые, высокопроизводительные компиляторы и значительно сокращает количество инструкций загрузки/сохранения по сравнению с другими RISC, особенно в прикладных программах большого размера. Для таких языков, как C++, где доминирующим является объектно-ориентированное программирование, регистровые окна обеспечивают еще большее сокращение числа выполняемых инструкций. Следует отметить, что регистровые окна управляются программным обеспечением супервизора, а не пользовательскими программами. Супервизор может сохранять минимальное количество регистров (приблизительно 24) во время переключения контекста, таким образом оптимизируя период ожидания переключения контекста.

Единственным отличием SPARC и RISC I & II (Беркли) является то, что SPARC обеспечивает большую свободу компилятору при назначении регистров переменным программы. Архитектура SPARC более гибкая, потому что управление регистровыми окнами не привязано к инструкциям вызова и возврата процедуры (CALL и JMPL), в отличие от машин Беркли. Вместо этого управление регистровыми окнами осуществляется отдельными инструкциями (SAVE и RESTORE).

5.1.2 Технические характеристики

SPARC присущи следующие основные особенности:

- линейное 32-битовое адресуемое пространство;
- упрощенные форматы инструкций – все инструкции имеют размер 32 бита и выравнивание по 32-битовой границе в памяти. Существует всего три базовых формата инструкций, обеспечивающих единообразное размещение кодов операции и полей адреса регистра. Доступ к памяти и области ввода / вывода имеют только инструкции загрузки и сохранения;

- малое количество режимов адресации – адрес памяти задается через «регистр + регистр» или «регистр + непосредственное значение»;

- триады адресов регистров – большинство инструкций работают с двумя регистровыми операндами (или с одним регистром и константой) и помещают результат в третий регистр;

- большой регистровый файл, «реализуемый посредством организации окон». В каждый отдельный момент времени программа «видит» восемь глобальных целочисленных регистров, 24-х регистровое окно, принадлежащее регистровому файлу. Регистровое окно может трактоваться, как способ ускоренного доступа к аргументам процедуры, локальным значениям и адресам возврата;

- отдельный регистровый файл вещественных регистров. В программах файл может трактоваться как набор из 32 регистров одинарной точности (32-разрядных) или 16 регистров двойной точности (64-разрядных), или 8 регистров учетверённой точности (128-разрядных), или как смесь тех или иных;

- отложенная передача управления – процессор всегда производит выборку команды, следующую за командой отложенной передачи управления. Будет ли она выполнена или нет, зависит от состояния «аннулирующего» бита в инструкции передачи управления;

- быстрая обработка системных прерываний – системные прерывания осуществляют векторную передачу управления через таблицу и вызывают выделение нового регистрового окна в регистровом файле;

- инструкции с метками – инструкции сложения / вычитания с метками предполагают, что два младших значащих бита операндов являются битами метки;

- инструкции мультипроцессорной синхронизации – одна инструкция обеспечивает атомарное считывание с последующей установкой разрядов памяти, остальные позволяют выполнить атомарный обмен значения регистра с содержимым памяти;

- сопроцессор – архитектура определяет непосредственный набор инструкций, в дополнение к набору инструкций оперирующих числами с плавающей запятой.

5.1.3 Словарь терминов

Ниже приводятся определения значимых терминов и аббревиатур, используемых в данном техническом описании:

1 Текущее окно – блок из 24 регистров r, на который ссылается указатель текущего окна (CWP).

2 Инструкции операций с плавающей запятой (FPop) – это инструкции, посредством которых выполняются расчеты с плавающей запятой, в соответствии с кодами операций FPop1 и FPop2. Инструкции FPop не включают операции загрузки и сохранения между памятью и FPU.

3 «Игнорируется» – используется для описания поля инструкции, содержимое которого является произвольным и не влияет на выполнение данной инструкции. Содержимое поля «игнорируется» не будет учитываться в последующих версиях архитектуры. Смотрите также «зарезервировано» и «не используется».

4 Реализация – аппаратное или программное обеспечение, соответствует всем спецификациям ISA.

5 Архитектура набора команд (ISA) определяет инструкции, регистры, память инструкций и данных, влияние выполненных инструкций на регистры и память, а также алгоритм управления выполнения инструкций. ISA не определяет частоту тактирования, количество тактов на инструкцию, внутреннюю шину данных и т.д.

6 Счетчик команд (PC) –адрес инструкции, выполняемой в настоящее время IU.

7 Следующее значение счетчика команд (nPC) – адрес следующей инструкции для выполнения (при отсутствии системного прерывания).

8 Привилегированный – инструкция (или регистр), которая может быть выполнена (или доступ, к которой возможен), если процессор находится в режиме супервизора (при PSR[S] = 1).

9 Процессор – совокупность блока целочисленной арифметики (IU), блока арифметики с плавающей запятой (FPU) и сопроцессора (CP), если он представлен.

10 Обозначения rs1, rs2, rd указывают регистры операнды инструкции. Регистры rs1 и rs2 – исходные регистры; rd – регистр назначения.

11 «Зарезервировано» – используется для описания поля инструкции или регистра, зарезервированного для описания в последующих версиях архитектуры. Программному обеспечению разрешается записывать в зарезервированное поле только ноль. Аппаратное обеспечение должно считывать из зарезервированного поля ноль; программное обеспечение, предназначенное для запуска на будущих версиях SPARC, не должно предполагать, что поле будет сброшенным при считывании.

12 Режим супервизора – режим процессора, который активен, если в регистре PSR установлен бит S (PSR[S] = 1).

13 Программное обеспечение супервизора – программное обеспечение, выполняющееся, пока процессор находится в режиме супервизора.

14 Системное прерывание – векторная передача управления программному обеспечению супервизора через таблицу, адрес которой указан в привилегированном регистре IU (Базовом регистре системных прерываний (TBR)).

15 «Не используется» – используется для описания поля инструкции или регистра, которое на настоящий момент не определяется архитектурой. При считывании программным обеспечением значение поля регистра «не используется» не определено. Однако, поскольку поле «не используется» может быть определено в последующих версиях архитектуры, в данное поле программному обеспечению разрешено записывать только ноль. См. также «игнорируется» и «зарезервировано».

16 Режим пользователя – режим процессора, который активен, если в регистре PSR сброшен бит S (PSR[S] = 0).

17 Прикладная программа пользователя – программа, выполняемая процессором в режиме пользователя. Также называется «прикладная программа». Следует отметить, что определения прикладных программ пользователя в настоящем руководстве могут быть неприменимы к программам (например, отладчикам), которые имеют доступ к привилегированному режиму супервизора (например, хранятся в дампе ядра).

18 Инструкции сопроцессора (CPop) – инструкции которые выполняют вычисления в сопроцессоре, определяемые кодами CPop1 и CPop2. CPop не включают в себя инструкции загрузки и сохранения между памятью и сопроцессором.

5.2 Форматы данных

Архитектура SPARC поддерживает три элементарных формата (или типа) данных:
- целочисленные со знаком – 8, 16, 32 и 64 бита;
- целочисленные без знака – 8, 16, 32 и 64 бита;
- с плавающей запятой – 32, 64 и 128 бит.

Размер формата:
- байтовый – 8 бит;
- полуслово – 16 бит;
- слово/одно слово – 32 бита;
- слово с меткой – 32 бита (30-битовое значение плюс 2 бита с меткой);
- двойное слово – 64 бита;
- четверное слово – 128 бит.

Форматы целочисленных данных со знаком кодируются целым числом дополнительного кода. Форматы целочисленных данных без знака являются универсальными и не кодируют конкретный тип данных; они могут обозначать целое число, строковое, дробное, булево значение и т.д. Форматы данных с плавающей запятой соответствуют стандарту IEEE двоичной арифметики с плавающей запятой, стандарту ANSI/IEEE 754-1985. Форматы данных с меткой обозначают слово, в котором два младших значащих бита рассматриваются как биты метки.

На рисунке 5.2.1 представлены форматы целочисленных данных со знаком, форматы целочисленных данных без знака и форматы данных с меткой. На рисунке 5.2.2 представлены форматы данных с плавающей запятой. На рисунках 5.2.1 и 5.2.2 отдельным словам, входящим в состав многословных форматов, присвоены имена.

Организация подформатов данных в регистрах памяти и процессора на базе данных имен представлена в таблице 5.2.1. В таблицах 5.2.2 – 5.2.5 представлены форматы целочисленных данных и данных с плавающей запятой.

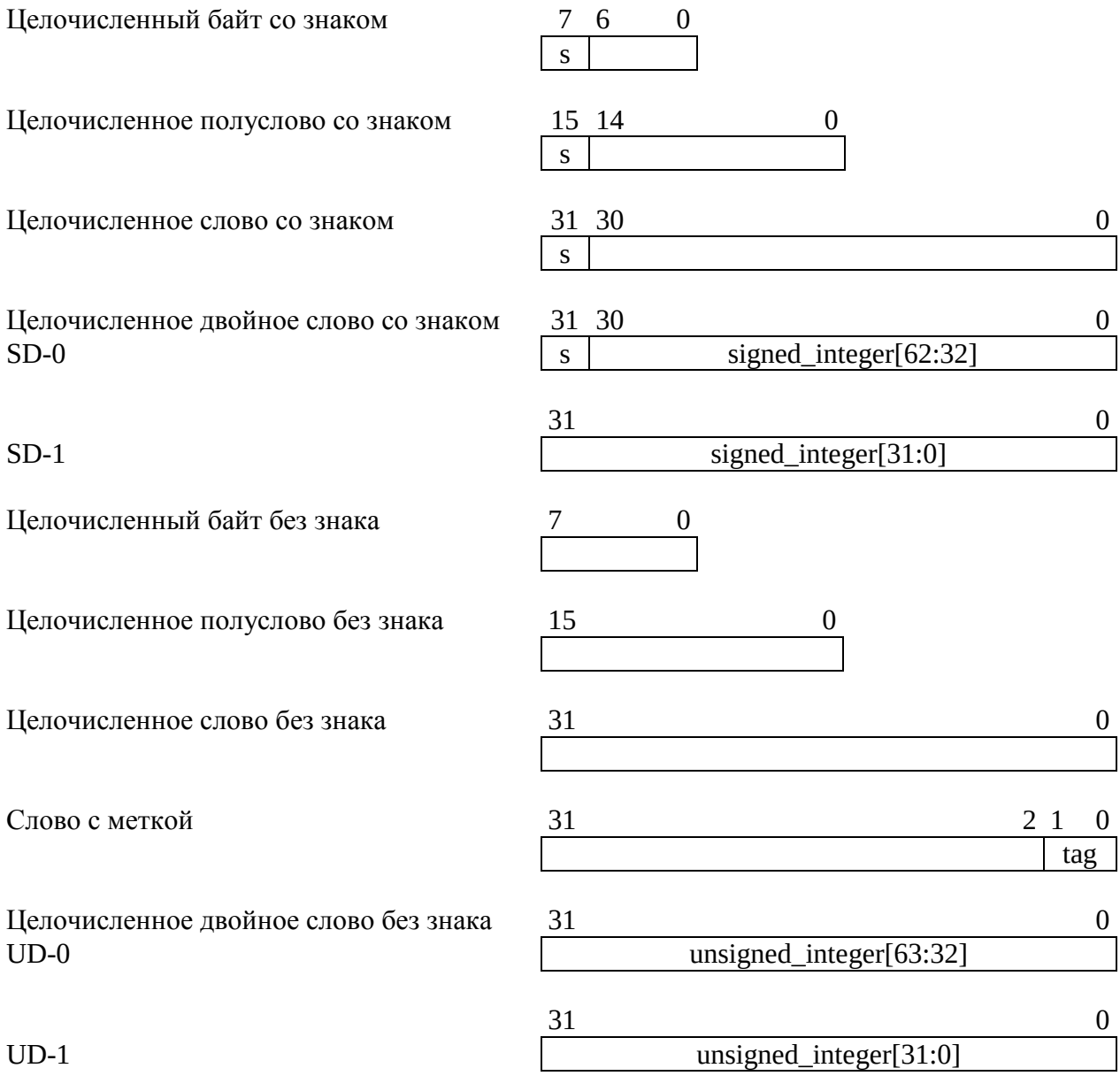


Рисунок 5.2.1 – Форматы целочисленных данных со знаком, целочисленных данных без знака и данных с меткой

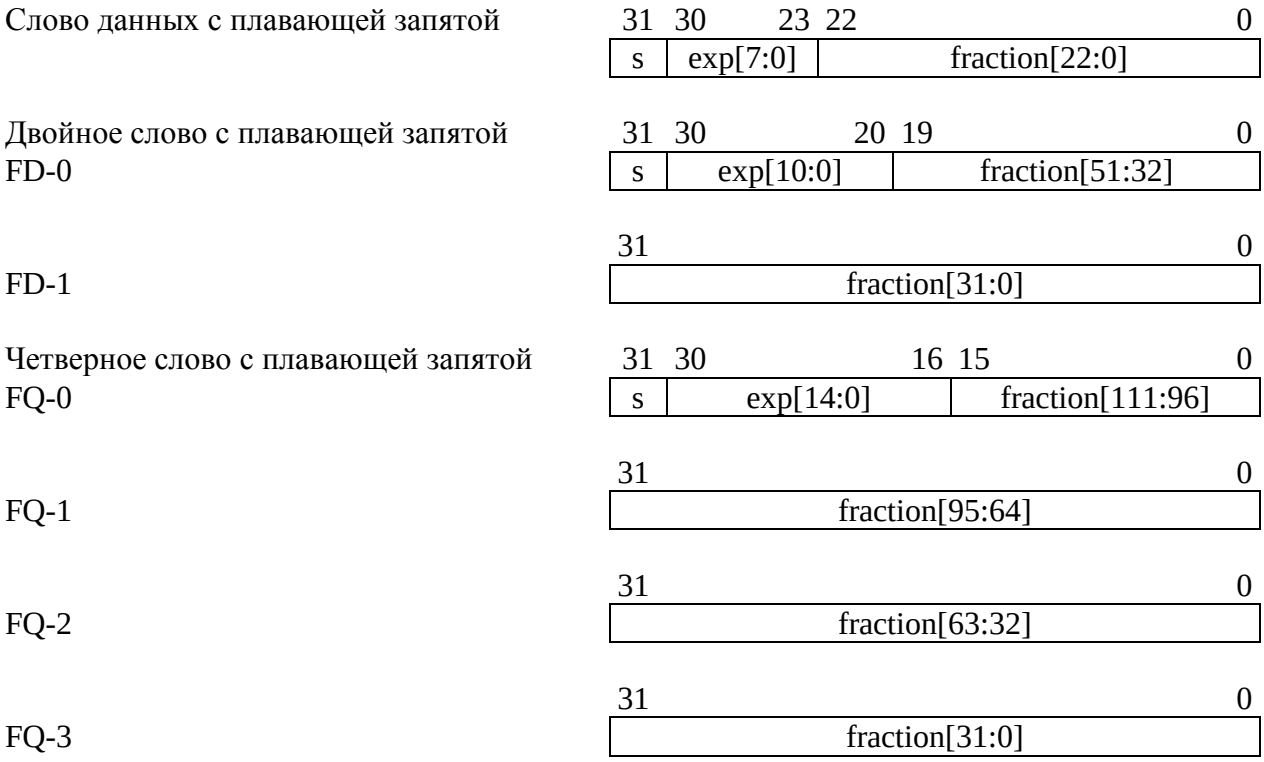


Рисунок 5.2.2– Форматы данных с плавающей запятой

Таблица 5.2.1 – Расположение двойных слов и учетверенных слов в памяти и регистрах

Имя подформата	Поле подформата	Выравнивание по адресу памяти	Адрес памяти	Выравнивание по номеру регистра	Номер регистра
SD-0	signed_integer[63:32]	0 mod 8	n	0 mod 2	r
SD-1	signed_integer[31:0]	4 mod 8	n+4	1 mod 2	r+1
UD-0	unsigned_integer[63:32]	0 mod 8	n	0 mod 2	r
UD-1	unsigned_integer[31:0]	4 mod 8	n+4	1 mod 2	r+1
FD-0	s:exp[10:0]:fraction[51:32]	0 mod 8	n	0 mod 2	r
FD-1	fraction[31:0]	4 mod 8	n+4	1 mod 2	r+1
FQ-0	s:exp[14:0]:fraction[111:96]	0 mod 8	n	0 mod 4	r
FQ-1	fraction[95:64]	4 mod 8	n+4	1 mod 4	r+1
FQ-2	fraction[63:32]	0 mod 8	n+8	2 mod 4	r+2
FQ-3	fraction[31:0]	4 mod 8	n+12	3 mod 4	r+3

Таблица 5.2.2 – Диапазон форматов целочисленных данных со знаком, целочисленных данных без знака и данных с меткой

Тип данных	Размер (биты)	Диапазон
Целочисленный байт со знаком	8	от -2^7 до 2^7-1
Целочисленное полуслово со знаком	16	от -2^{15} до $2^{15}-1$
Целочисленное слово со знаком	32	от -2^{31} до $2^{31}-1$
Целочисленное слово с меткой со знаком	32	от -2^{29} до $2^{29}-1$
Целочисленное двойное слово со знаком	64	от -2^{63} до $2^{63}-1$
Целочисленный байт без знака	8	от 0 до 2^8-1
Целочисленное полуслово без знака	16	от 0 до $2^{16}-1$
Целочисленное слово без знака	32	от 0 до $2^{32}-1$
Целочисленное слово с меткой без знака	32	от 0 до $2^{30}-1$
Целочисленное двойное слово без знака	64	от 0 до $2^{64}-1$

Таблица 5.2.3 – Определение формата одного слова с плавающей запятой

s = знак (1 бит) e = смещенный порядок (8 бит) f = мантисса (23 бита) u = не определено	
нормализованное значение (0 < e < 255):	$(-1)^s \times 2^{e-127} \times 1.f$
субнормальное значение (e = 0):	$(-1)^s \times 2^{-126} \times 0.f$
ноль (e = 0):	$(-1)^s \times 0$
с уведомлением о NaNs (SNaNs):	s = u; e = 255 (max); f = .0uu…uu (по крайней мере, один бит мантиссы должен быть установлен)
без уведомления о NaNs (QNaNs):	s = u; e = 255 (max); f = .1uu…uu
– ∞ (минус бесконечность):	s = 1; e = 255 (max); f = .000…00
+ ∞ (плюс бесконечность):	s = 0; e = 255 (max); f = .000…00

Таблица 5.2.4 – Определение формата двойного слова с плавающей запятой

s = знак (1 бит), e = смещенный порядок (11 бит) f = мантисса (52 бита) u = не определено	
нормализованное значение (0 < e < 2047):	$(-1)^s \times 2^{e-1023} \times 1.f$
субнормальное значение (e = 0):	$(-1)^s \times 2^{-1022} \times 0.f$
ноль (e = 0):	$(-1)^s \times 0$
с уведомлением о NaNs (SNaNs):	s = u; e = 2047 (max); f = .0uu…uu (по крайней мере, один бит мантиссы должен быть установлен)
без уведомления о NaNs (QNaNs):	s = u; e = 2047 (max); f = .1uu…uu
– ∞ (минус бесконечность):	s = 1; e = 2047 (max); f = .000…00
+ ∞ (плюс бесконечность):	s = 0; e = 2047 (max); f = .000…00

Таблица 5.2.5 – Определение формата четверного слова с плавающей запятой

s = знак (1 бит) e = смещенный порядок (15 бит) f = мантисса (112 бит) u = не определено	
нормализованное значение (0 < e < 32767):	$(-1)^s \times 2^{e-16383} \times 1.f$
субнормальное значение (e = 0):	$(-1)^s \times 2^{-16382} \times 0.f$
ноль (e = 0):	$(-1)^s \times 0$
с уведомлением о NaNs (SNaNs):	s = u; e = 32767 (max); f = .0uu…uu (по крайней мере, один бит мантиссы должен быть установлен)
без уведомления о NaNs (QNaNs):	s = u; e = 32767 (max); f = .1uu…uu
– ∞ (минус бесконечность):	s = 1; e = 32767 (max); f = .000…00
+ ∞ (плюс бесконечность):	s = 0; e = 32767 (max); f = .000…00

5.3 Регистры

Процессор SPARC использует два типа регистров: регистры общего назначения или «рабочие» регистры данных и регистры управления / статуса. Регистры общего назначения IU называются регистрами r; регистры общего назначения FPU называются регистрами f.

Регистры управления / статуса IU включают:
- регистр состояния процессора (PSR);
- маску недопустимости окна (WIM);
- базовый регистр системного прерывания (TBR);
- регистр умножения / деления (Y);
- счетчики команд (PC, nPC);
- зависящие от конкретной реализации вспомогательные регистры состояния (ASR);
- зависящую от конкретной реализации очередь отложенных прерываний IU.

Регистры управления / статуса FPU включают:
- регистр состояния блока арифметики с плавающей запятой (FSR);
- зависящую от конкретной реализации очередь отложенных прерываний операций с плавающей запятой (FQ).

5.3.1 Регистры IU

Регистры, реализуемые посредством организации окна

Инструкция имеет доступ к восьми глобальным регистрам и 24-регистровому окну через регистры r. Регистровое окно охватывает 8 входных и 8 локальных регистров отдельного набора регистров вместе с 8 входными регистрами смежного набора регистров, которые адресуются из текущего окна в качестве выходных регистров. Доступ к ресурсам окна показан в таблице 5.3.1. Количество окон или наборов регистров (NWINDOWS), в зависимости от реализации, может быть от 2 до 32 включительно. Здесь и далее по тексту будет рассматриваться вариант NWINDOWS = 8, который реализован в данном процессоре.

Таблица 5.3.1 – Доступ к ресурсам окна

Адрес регистра, реализуемого посредством организации окна	Адрес регистра r
входной[0] – входной[7]	r[24] – r[31]
локальный[0] – локальный[7]	r[16] – r[23]
выходной[0] – выходной[7]	r[8] – r[15]
глобальный[0] – глобальный[7]	r[0] – r[7]

Текущее окно регистров r задается через указатель на текущее окно (CWP) – 5-битовое поле регистра состояния процессора (PSR). CWP увеличивается при выполнении инструкции RESTORE (или RETT) и уменьшается на единицу при выполнении инструкции SAVE или при системном прерывании.

Переполнение или обратное переполнение окон обнаруживается через использование регистра маски недопустимости окна WIM, управляемого программным обеспечением супервизора (операционной системой).

Перекрытие окон

Каждое окно совместно использует свои входные и выходные регистры с двумя смежными окнами. Выходные регистры окна CWP + 1 доступны в качестве входных регистров текущего окна, выходные регистры текущего окна доступны в качестве входных регистров окна CWP – 1. Каждое окно использует отдельные локальные регистры, смотри рисунок 5.3.1.

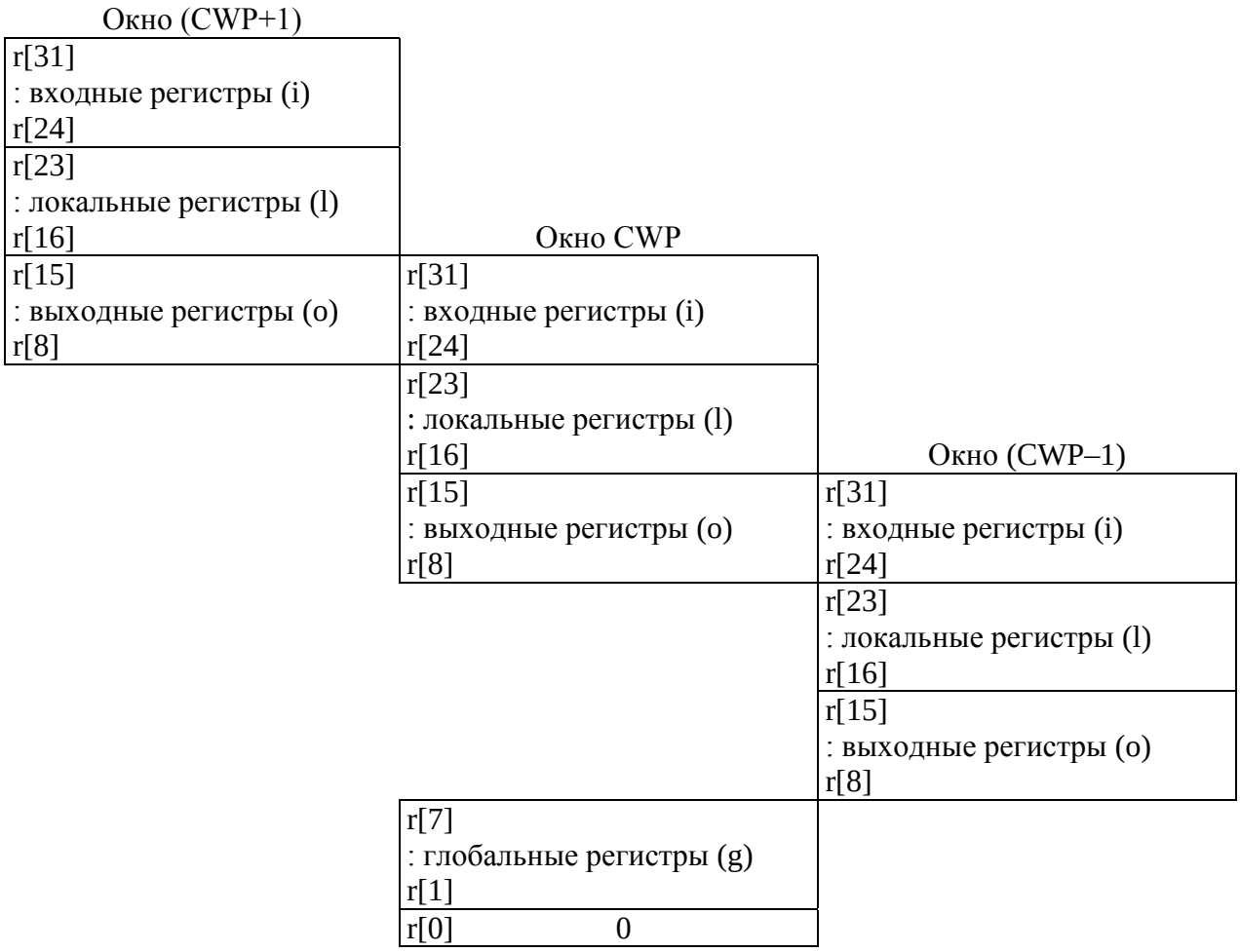


Рисунок 5.3.1 – Три перекрывающихся окна и 8 глобальных регистров

Регистр r с адресом o, где $8 \le o \le 15$, ссылается на такой же регистр, что и (o + 16) после уменьшения CWP на 1 (по модулю 8). Точно так же регистр с адресом i, где $24 \le i \le 31$, ссылается на такой же регистр, что и адрес (i – 16) после увеличения CWP на 1 (по модулю 8).

Поскольку операции над указателем окна CWP выполняются по модулю 8 (NWINDOWS), окно с наибольшим значением имеет перекрытие с окном 0. Выходные регистры окна 0 являются входными регистрами окна 7. Окна имеют непрерывную нумерацию в диапазоне от 0 до 7.

Замечания по программированию

Поскольку инструкции вызова процедуры (CALL и JMPL) не меняют CWP, вызов процедуры выполняется без изменения окна.

В связи с перекрытием окон их количество, доступное программно, на единицу меньше чем количество реализованных окон и равно 7 (NWINDOWS – 1). После заполнения регистрового файла выходные регистры последнего окна являются входными регистрами самого старого окна, которое по-прежнему содержит действительные программные данные.

Не стоит строить предположения о значениях, содержащихся в «локальных» и «выходных» регистрах регистрового окна при повторном входе в окно посредством инструкции SAVE. Если обработка системных прерываний разрешена, и программа выполняет RESTORE, за которым следует SAVE, то «локальные» и «выходные» регистры результирующего окна могут содержать недействительные значения, поскольку между RESTORE и SAVE могло произойти системное прерывание. Однако если системные

прерывания запрещены, то данные локальных и выходных регистров гарантированно останутся действительными.

Компоненты операции с двойными словами

Инструкции, имеющие доступ к двойному слову в регистрах r, выполняются с учетом выравнивания регистра чет-нечет. Младший значащий бит в адресе регистра r данных инструкций зарезервирован и для обеспечения соответствия должен быть сброшен программным обеспечением.

При попытке выполнения инструкции загрузки или сохранения двойного слова со ссылкой на нечетный номер регистра назначения вызовет системное прерывание illegal_instruction.

Специальные регистры r

В архитектуре полностью или частично зафиксировано использование четырех регистров r:

- если осуществляется доступ к r[0] как к входному операнду (rs1 = 0 или rs2 = 0 или rd = 0 для инструкции сохранения), то считывается постоянное значение 0. Если r[0] используется как выходной операнд (rd = 0, кроме инструкций сохранения), записанные данные игнорируются (значение регистров r не изменяется);
- при выполнении инструкции CALL её собственный адрес записывается в r[15] (выходной регистр 7);
- при выполнении системного прерывания счетчики программ PC и nPC копируются в регистры r[17] и r[18] (локальные регистры 1 и 2) нового регистрового окна системного прерывания.

Использование регистров

На рисунке 5.3.2 NWINDOWS = 8. Восемь глобальных регистров на нем не представлены. Наборы регистров обозначены жирным шрифтом. CWP = 0 и WIM[7] = 1. Если процедура, использующая окно w0, выполняет RESTORE, то окно w1 становится текущим. Если процедура, использующая окно w0 выполняет SAVE, то произойдет системное прерывание window_overflow. Обработчик системных прерываний переполнений будет использовать локальные регистры w7.

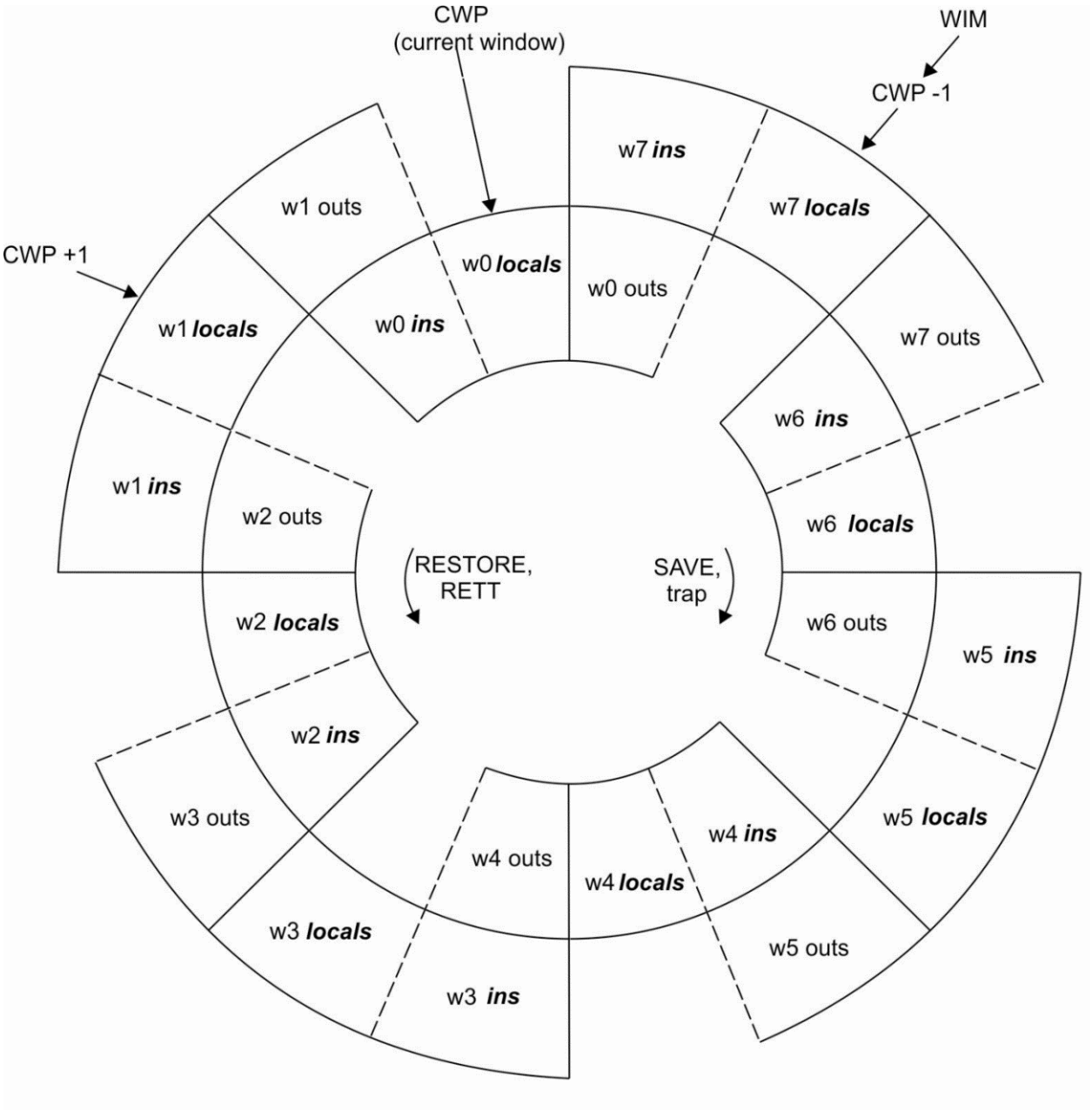


Рисунок 5.3.2 – Регистры r, реализуемые посредством организации окна

5.3.2 Регистры управления/статуса блока целочисленной арифметики (IU)

К регистрам управления/статуса IU относятся регистр состояния процессора (PSR), регистр маски недопустимости окна (WIM), базовый регистр системных прерываний (TBR), регистр умножения/деления (Y), счетчики команд (PC и nPC) и вспомогательные регистры состояния (ASR), присутствующие в зависимости от реализации, а также очередь отложенных прерываний IU.

Регистр состояния процессора (PSR)

32-битовый PSR содержит различные поля, управляющие процессором и содержащие информацию о статусе, см. таблицу 5.3.2. Модификация данного регистра выполняется с помощью инструкций SAVE, RESTORE, Ticc и RETT, а также с помощью всех инструкций, которые модифицируют условные коды. С помощью

привилегированных инструкций RDPSR и WRPSR выполняется непосредственное считывание и запись PSR.

Таблица 5.3.2 – Поля PSR

impl	ver	icc	–	EC	EF	PIL	S	PS	ET	CWP
31–28	27–24	23–20	19–14	13	12	11–8	7	6	5	4–0

PSR предусматривает следующие поля:

PSR_implementation (impl) –биты 31–28 являются жестко заданными и служат для идентификации реализации или класса реализаций архитектуры. Аппаратное обеспечение не изменяет данное поле в ответ на выполнение инструкции WRPSR. Поля PSR.impl и PSR.ver определяют уникальную реализацию или класс реализаций архитектуры.

PSR_version (ver) – биты 27–24 имеют зависимость от реализации. Поле ver может быть жестко заданным, идентифицирующим одну или более конкретную реализацию, или выполнять функцию считываемого или записываемого поля состояния, в зависимости от реализации.

PSR_integer_cond_codes (icc) – биты 23–20 являются кодами условия IU. Модификация данных битов выполняется с помощью арифметических и логических инструкций, имена которых оканчиваются буквами cc (например, ANDcc), а также с помощью инструкции WRPSR. При выполнении инструкций Bicc и Ticc выполняется передача управления на основе значений данных битов, представленных в таблице 5.3.3.

Таблица 5.3.3 – Поля целочисленных условных кодов (icc) PSR

n	z	v	c
23	22	21	20

PSR_negative (n) – бит 23 показывает, был ли отрицательным 32-битный результат работы АЛУ в дополнительном коде для последней команды, которая модифицировала поле icc. Значения: 1 = отрицательный, 0 = не отрицательный.

PSR_zero (z) – бит 22 показывает, был ли нулевым 32-битный результат работы АЛУ для последней инструкции, которая модифицировала поле icc. Значения: 1 = нулевой, 0 = не нулевой.

PSR_overflow (v) – бит 21 показывает, принадлежит ли результат заботы АЛУ диапазону 32-битного дополнительного кода (может ли он быть представлен с его помощью) для последней инструкции, которая модифицировала поле icc. Значения: 1 = переполнение, 0 = без переполнения.

PSR_carry (c) – бит 20 показывает наличие переноса (или заема) произошедшего при выполнении последней инструкции, которая модифицировала поле icc. Данный бит устанавливается при суммировании, если осуществляется перенос в бит 31. Бит переноса устанавливается при вычитании, если осуществляется заем из бита 31. Значения: 1 = перенос, 0 = без переноса.

PSR_reserved – биты (19–14) зарезервированы. При считывании с помощью инструкции RDPSR данные биты сброшены. Для обеспечения совместимости с последующими версиями супервизор программно должен выдавать инструкции WRPSR с нулевыми значениями в данном поле.

PSR_enable_coprocessor (EC) – бит 13 определяет разрешено ли использование сопроцессора (он может присутствовать или нет, в зависимости от реализации). Если не разрешено, при выполнении инструкции сопроцессора возникнет системное прерывание. Значения: 1 = разрешено, 0 = не разрешено. Если в реализации нет аппаратной поддержки сопроцессора, то PSR.EC всегда должен считываться как 0, запись в него должна быть проигнорирована.

PSR_enable_floating-point (EF) – бит 12 определяет разрешение FPU. Если не разрешено, срабатывает системное прерывание при выполнении инструкции с плавающей

запятой. Значения: 1 = разрешено, 0 = не разрешено. Если в реализации нет аппаратной поддержки FPU, то PSR.EF всегда должен считываться как 0, запись в него должна быть проигнорирована.

Замечания по программированию

Программное обеспечение может использовать биты EF и EC для определения наличия FPU или CP в конкретном процессоре. Если процесс не использует FPU / CP, то при переключении контекста сохранение их регистров не требуется.

PSR_proc_interrupt_level (PIL) – биты 11 (старший значащий бит) – 8 (младший значащий бит) определяют уровень прерывания, при превышении которого процессор будет реагировать на прерывание.

PSR_supervisor (S) – бит 7 определяет режим процессора: режим супервизора или пользователя. Значения: 1 = режим супервизора, 0 = режим пользователя.

PSR_previous_supervisor (PS) – бит 6 содержит значение бита S на момент срабатывания последнего системного прерывания.

PSR_enable_traps (ET) – бит 5 определяет, разрешены ли системные прерывания. Системное прерывание автоматически сбрасывает бит ET. Если ET = 0, внешние прерывания игнорируются, остальные исключения приводят к остановке выполнения инструкций IU, что обычно приводит к осуществлению системного прерывания reset, которое возобновляет исполнение инструкций с адреса 0. Значения: 1 = системные прерывания разрешены, 0 = системные прерывания не разрешены.

PSR_current_window_pointer (CWP) – биты 4 (старший значащий бит) – 0 (младший значащий бит) содержат указатель на текущее окно, счетчик, определяющий текущее окно регистров r. Значение CWP аппаратно уменьшается в случае системного прерывания и при выполнении инструкций SAVE и увеличивается при выполнении инструкций RESTORE и RETT (по модулю NWINDOWS).

Регистр маски недопустимости окна (WIM)

Регистр маски недопустимости окна (WIM) управляется программно супервизором и используется аппаратно для определения генерации системных прерываний переполнения или обратного переполнения окна вызванных выполнением инструкции SAVE, RESTORE или RETT. Формат регистра приведен в таблице 5.3.4.

Таблица 5.3.4 – Поля WIM

W31	W30	W29	…	W2	W1	W0
31	30	29		2	1	0

Для каждого реализованного набора регистров или регистрового окна в WIM имеется бит активного статуса. WIM[n] соответствует набору регистров, доступ к которому выполняется при CWP = n.

При выполнении инструкции SAVE, RESTORE или RETT текущее значение CWP сравнивается с WIM. Если инструкция SAVE, RESTORE или RETT приведет к тому, что CWP будет указывать на «недопустимый» набор регистров, т.е. набор регистров, соответствующий бит WIM которого равен 1 (WIM[CWP] = 1), то сработает системное прерывание window_overflow или window_underflow.

Считывание WIM выполняется с помощью привилегированной инструкции RDWIM, запись выполняется с помощью привилегированной инструкции WRWIM. Биты, соответствующие нереализованным окнам, считываются как сброшенные, при этом значения, записанные в нереализованные биты, игнорируются. Если через WRWIM записать значение состоящее из всех единиц, то при последующем считывании RDWIM выдаст битовый вектор, в котором реализованные окна (и только они) будут обозначены единицами.

Базовый регистр системных прерываний (TBR)

Базовый регистр системных прерываний (TBR) содержит три поля, вместе равных адресу, по которому передается управление при срабатывании системного прерывания, как показано в таблице 5.3.5.

Таблица 5.3.5 – Поля TBR

TBA	tt	zero
31–12	11–4	3–0

TBR предусматривает следующие поля:

TBR_trap_base_address (TBA) – биты 31–12 определяют базовый адрес системных прерываний, который устанавливается программным обеспечением супервизора. Он содержит 20 старших значащих битов адреса таблицы системных прерываний. Запись в поле TBA выполняется с помощью инструкции WRTBR.

TBR_trap_type (tt) – биты 11–4 определяют поле типа системного прерывания (tt). Запись в данное 8-битовое поле выполняется аппаратно при срабатывании системного прерывания, при этом значение сохраняется до следующего системного прерывания. Данное поле обуславливает смещение в таблице системных прерываний. При выполнении инструкции WRTBR поле tt не изменяется.

TBR_zero (0) – биты 3–0 сброшены. При выполнении инструкции WRTBR данное поле не изменяется. Для обеспечения совместимости с последующими версиями супервизор программно должен выполнять инструкцию WRTBR только с нулевым значением в данном поле.

Для получения дополнительной информации – см. подраздел «5.4 Системные прерывания».

Регистр умножения/деления (Y)

32-битовый регистр Y содержит старшее значащее слово произведения с удвоенной точностью целочисленного умножения, полученное в результате выполнения инструкции целочисленного умножения (SMUL, SMULcc, UMUL, UMULcc) или подпрограммы, которая использует инструкции поэтапного целочисленного умножения (MULScc). Регистр Y сохраняет старшее значащее слово делимого с удвоенной точностью при выполнении инструкции целочисленного деления (SDIV, SDIVcc, UDIV, UDIVcc).

Запись и считывание регистра Y выполняется с помощью инструкций RDY и WRY.

Счетчики команд (PC, nPC)

32-битовый регистр PC содержит адрес текущей инструкции, выполняемой IU. Регистр nPC содержит адрес следующей инструкции (при условии отсутствия системного прерывания).

В случае передачи управления с задержкой инструкция, непосредственно следующая за инструкцией передачи, называется отложенной инструкцией. Отложенная инструкция выполняется (если инструкция передачи управления не аннулирует ее) до передачи управления адресату. При выполнении отложенной инструкции nPC указывает на адресата инструкции передачи управления, в то время как PC указывает на саму отложенную инструкцию.

PC считывается инструкциями CALL или JMPL. PC и nPC записываются в два локальных регистра при срабатывании системного прерывания.

Вспомогательные регистры состояния (ASR)

Архитектура SPARC предусматривает до 31-го вспомогательного регистра состояния (ASR), пронумерованных от «1» до «31».

ASR от «1» до «15» зарезервированы для дальнейшего использования архитектурой; программно не используются.

ASR от «16» до «31» доступны в зависимости от реализации: для таймеров, счетчиков, диагностических регистров, регистров самотестирования и регистров управления системными прерываниями. Конкретная реализация IU – иметь от 0 до 16 таких ASR. Семантика доступа к ASR зависит от реализации. Привилегированность конкретного вспомогательного регистра состояния зависит от реализации.

Считывание и запись ASR выполняется с помощью инструкций RDASR и WRASR. Инструкция считывания/записи ASR привилегированная, если регистр, к которому выполняется доступ, привилегирован.

Очередь отложенных прерываний IU

Микропроцессор может содержать от нуля и более очередей отложенных прерываний. Такие очереди содержат достаточно состояний для реализации отложенных прерываний, вызванных IU. Следует отметить, что отложенные системные прерывания fp_exception и cp_exception обрабатываются очередями отложенных системных прерываний устройства с плавающей запятой и сопроцессора соответственно.

Считывание и запись очередей отложенных прерываний IU выполняется с помощью привилегированных инструкций выбора загрузки/сохранения в альтернативное адресное пространство или считывания/записи вспомогательных регистров состояния.

Содержимое и функционирование очередей отложенных прерываний зависит от реализации и не видимо для прикладных программ пользователя.

5.3.3 Регистры FPU

FPU содержит тридцать два 32-битовых регистра f с плавающей запятой, которые пронумерованы от f[0] до f[31]. В отличие от регистров r, реализуемых посредством организации окна, в каждый момент времени инструкция имеет доступ к любому из 32 регистров f. Считывание и запись в регистры f выполняется с помощью инструкций FPop (формата FPop1/FPop2), а также инструкций загрузки/сохранения данных с плавающей запятой одинарной/двойной точности (LDF, LDDF, STF, STDF), см. таблицу 5.3.6.

Таблица 5.3.6 – Регистры f

f[31]
f[30]
:
f[1]
f[0]

31 0

Операнды двойной и четверной точности

Один регистр f может сохранять один операнд одинарной точности. Операнд двойной точности требует выровненную по адресу пару регистров f, операнд четверной точности требует четыре выровненных по адресу регистра f. Таким образом, в каждый момент времени регистры f могут хранить до 32 операндов одинарной точности, до 16 операндов двойной точности или до восьми операндов четверной точности.

Инструкции, имеющие доступ к данным с плавающей запятой удвоенной точности в регистрах f, предполагают выравнивание адреса. Младший значащий бит спецификатора адреса двойного слова регистров f зарезервирован и должен быть сброшен программными средствами. Аналогично два младших значащих бита спецификатора адреса четверного слова регистров f зарезервированы и должны быть программно сброшены, см. таблицу 5.3.7.

Таблица 5.3.7 – Таблица данных с плавающей запятой двойной и четверной точности в регистрах f

Имя подформата	Поля формата	Адрес регистра f
FD-0	s:exp[10:0]:fraction[51:32]	0 по модулю 2
FD-1	fraction[31:0]	1 по модулю 2
FQ-0	s:exp[14:0]:fraction[111:96]	0 по модулю 4
FQ-1	fraction[95:64]	1 по модулю 4
FQ-2	fraction[63:32]	2 по модулю 4
FQ-3	fraction[31:0]	3 по модулю 4

Рекомендуется (но не является обязательным), в ответ на попытку выполнения инструкции, использующей не выровненное значение регистра f в качестве операнда, генерация системного прерывания fp_exception с FSR.ftt = 6 (invalid_fp_register). Данная рекомендация применяется для операнда двойной точности, находящегося в регистре, номер которого по модулю 2 не равен нулю. Также данная рекомендация применяется для операнда четверной точности, находящегося в регистре, номер которого по модулю 4 не равен нулю.

5.3.4 Регистры управления/статуса FPU

К регистрам управления/статуса FPU относятся 32-битовые регистр состояний операций с плавающей запятой (FSR), в котором содержится информация о режиме и статусе FPU, а также, присутствующую или нет в зависимости от реализации, очередь отложенных прерываний операций с плавающей запятой (FQ).

Регистр состояния операций с плавающей запятой (FSR)

Поля регистра FSR содержат информацию о режиме и состоянии FPU (см. таблицу 5.3.8). Считывание и запись в FSR выполняется с помощью инструкций STFSR и LDFSR.

Таблица 5.3.8 – Поля FSR

RD	u	TEM	NS	res	ver	ftt	qne	u	fcc	aexc	cexc
31–30	29–28	27–23	22	21–20	19–17	16–14	13	12	11–10	9–5	4–0

FSR предусматривает следующие поля:

FSR_rounding_direction (RD) – биты 31, 30 определяют направление округления результатов операций с плавающей запятой в соответствии со стандартом ANSI/IEEE 754-1985 (как показано в таблице 5.3.9).

Таблица 5.3.9 – Поле направления округления (RD) FSR

RD	Округление до:
0	ближайшего значения (четного, с привязкой)
1	0
2	+ ∞
3	– ∞

FSR_unused (u) – биты 29, 28 и 12 не используются. Для обеспечения совместимости с последующими версиями программному обеспечению необходимо использовать инструкцию LDFSR с нулевыми значениями в данных битах.

FSR_trap_enable_mask (TEM) – биты 27–23 являются битами разрешения для каждого из пяти исключений с плавающей запятой, которые могут быть выставлены в поле current_exception (cexc). Если инструкция операции с плавающей запятой генерирует одно или несколько исключений, а бит TEM, соответствующий одному или нескольким из исключений, установлен, срабатывает системное прерывание fp_exception. Нулевое значение TEM предотвращает генерацию системного прерывания при конкретном типе исключений.

FSR_nonstandard_fp (NS) – если бит 22 установлен, то результат работы FPU зависит от конкретной реализации, и может не соответствовать стандарту ANSI/IEEE 754-1985. Например, для повышения производительности субнормальные операнды или результаты с плавающей запятой могут конвертироваться в ноль, если установлен бит NS.

FSR_reserved (res) – биты 21 и 20 зарезервированы. При считывании с помощью инструкции STFSR данные биты читаются как «0». Для обеспечения совместимости с последующими версиями программному обеспечению необходимо использовать инструкцию LDFSR с нулевыми значениями в данных битах.

FSR_version (ver) – биты 19 – 17 идентифицируют одну или более конкретных реализаций архитектуры FPU. Для каждой реализации SPARC IU (в соответствии со значениями полей PSR.impl и PSR.vers), может быть одна или более реализация FPU, или реализации FPU могут отсутствовать. Данное поле идентифицирует конкретную реализацию FPU. Номер версии 7 зарезервирован для обозначения того, что аппаратно реализованный контроллер операций с плавающей запятой отсутствует.

FSR_floating-point_trap_type (ftt) – биты 16 – 14 идентифицируют типы системных прерываний при выполнении операций с плавающей запятой. После возникновения исключения операции с плавающей запятой, поле ftt кодирует тип данного исключения, пока не будет выполнена инструкция STFSR или FPop.

Считывание поля ftt выполняется с помощью инструкции STFSR. Инструкция LDFSR не оказывает влияния на ftt.

Программное обеспечение, которое обрабатывает системные прерывания с плавающей запятой, в режиме супервизора для определения типа системного прерывания с плавающей запятой должно выполнить STFSR. Сброс ftt в явной форме с помощью STFSR зависит от реализации; если ftt не сбрасывается с помощью STFSR, то обработчик системного прерывания должен гарантировать, что при выполнении последующего STFSR из режима пользователя прочитанное значение ftt будет нулевым.

Замечания по программированию

Инструкция LDFSR не может быть использована для сброса ftt, поскольку она не изменяет значение данного поля. Однако выполнение команды FPop не генерирующей системного прерывания, например «fmovs %f0,%f0», до возврата в режим пользователя приведет к обнулению ftt. Значение ftt остается действительным, пока не будет выполнена следующая инструкция FPop.

Данное поле кодирует тип ошибок в соответствии с таблицей 5.3.10. Следует отметить, что значение 7 зарезервировано для последующих версий.

Таблица 5.3.10 – Поле типов системных прерываний с плавающей запятой (ftt) FSR

ftt	Тип системного прерывания
0	нет
1	IEEE_754_exception
2	unfinished_FPop
3	unimplemented_FPop
4	sequence_error
5	hardware_error
6	invalid_fp_register
7	зарезервировано

При нормальном ходе выполнения вычислений не предполагается возникновение системных прерываний типа sequence_error и hardware_error. С точки зрения пользовательских приложений, после данных прерываний восстановление работы невозможно.

Напротив, при нормальном ходе вычислений ожидается эпизодическое срабатывание системных прерываний типа IEEE_754_exception, unfinished_FPop и unimplemented_FPop. Программное обеспечение супервизора должно быть готово восстановить работу после них. Когда происходит системное прерывание операции с плавающей запятой (с точки зрения пользовательского обработчика сигнала (системного прерывания)):

- значение aexc не изменяется;
- значение cexc не изменяется, кроме тех случаев, когда при EEE_754_exception устанавливается именно тот один бит, который соответствует захватываемому исключению. Исключения операций с плавающей запятой unfinished_FPop, unimplemented_FPop и sequence_error не оказывают влияния на cexc;
- исходные регистры f не изменяются (обычно реализована и неизменность регистров назначения f);
- значение fcc не изменяется.

Все выше обозначенное относится к результату, доступному для пользовательского обработчика сигнала при сигнализировании об IEEE исключении, непосредственно после IEEE_754_exception или после восстановления работы в случае unfinished_FPop или unimplemented_FPop. В любом случае, в поле cexc, как видно обработчику системного прерывания, будет отображать исключение, вызвавшее системное прерывание.

В случае системных прерываний unfinished_FPop и unimplemented_FPop, которые впоследствии не генерируют IEEE исключения, от программного обеспечения, восстанавливающего работу, ожидаются действия по определению cexc, aexc и либо регистр назначения f, либо fcc, соответственно.

ftt = IEEE_754_exception – системное прерывание типа IEEE_754_exception операции с плавающей запятой обозначает, что при выполнении операции с плавающей запятой произошла исключительная ситуация в соответствии со стандартом ANSI/IEEE 754-1985. Тип системного прерывания кодируется в поле cexc. Следует отметить, что aexc, fcc и регистр назначения f не изменяются системным прерыванием IEEE_754_exception.

ftt = unfinished_FPop – Unfinished_FPop обозначает, что FPU в данной реализации не может сгенерировать правильные результаты или исключения в соответствии со стандартом ANSI/IEEE 754-1985. В данном случае поле cexc не изменяется.

ftt = unimplemented_FPop – Unimplemented_FPop обозначает, что реализованный FPU декодировал FPop, которая не реализована. В данном случае поле cexc не меняется.

Замечания по программированию

В случае срабатывания системного прерывания операции с плавающей запятой типа unfinished_FPop или unimplemented_FPop программное обеспечение обязано

эмулировать или повторно выполнить инструкцию, вызвавшую исключение, а также обновить FSR, регистр(-ы) назначения f и fcc.

ftt = sequence_error – Sequence_error обозначает одно из трех аномальных условий ошибки в FPU, каждое из которых вызывается некорректной работой программного обеспечения супервизора:

- попытка выполнения инструкции STDFQ в реализации, в которой отсутствует очередь отложенных прерываний с плавающей запятой (FQ);
- попытка выполнения инструкции с плавающей запятой, которую FPU не может принять. Этот тип sequence_error возникает из-за логической ошибки в программном обеспечении супервизора, которая вызвала неполную обработку предыдущего системного прерывания с плавающей запятой (например, очередь с плавающей запятой не была опустошена после предыдущего исключения с плавающей запятой);
- попытка выполнения инструкции STDFQ при пустой очереди отложенных прерываний с плавающей запятой (FQ), т.е. при FSR.qne = 0. (Следует отметить, что генерация sequence_error в данном случае рекомендуется, но не является обязательной).

Замечания по программированию

Если в процессе выполнения пользовательского кода возникает исключение fp_exception с типом sequence_error (в виду вышеперечисленных условий), восстановление состояния достаточного для продолжения выполнения пользовательского приложения может оказаться невозможным.

ftt = hardware_error – hardware_error обозначает, что FPU обнаружил серьезную внутреннюю ошибку, например, недопустимое состояние или ошибку по четности при выполнении доступа к регистру f.

Если в процессе выполнения пользовательского кода возникает исключение hardware_error, восстановление состояния достаточного для продолжения выполнения пользовательского приложения может оказаться невозможным.

ftt = invalid_fp_register – тип системного прерывания invalid_fp_register обозначает невыравнивание одного (или более) операндов FPop, т.е. номер регистра с удвоенной точностью – не 0 по модулю 2 или номер регистра с учетверенной точностью – не 0 по модулю 4. В данном случае рекомендуется генерация системного прерывания fp_exception с FSR.ftt = invalid_fp_register, но в реализации может быть принято решение не генерировать данное системное прерывание.

FSR_FQ_not_empty (qne) – бит 13 указывает, является ли необязательная очередь отложенных системных прерываний с плавающей запятой (FQ) пустой после отложенного системного прерывания fp_exception или после выполнения инструкции сохранения двойного слова в очередь операций с плавающей запятой (STDFQ). Если qne = 0, то очередь пуста; если qne = 1, очередь не пуста.

Считывание бита qne может быть выполнено с помощью инструкции STFSR. Инструкция LDFSR не изменяет qne. Однако выполнение следующих одна за другой инструкций STDFQ приведет (в итоге) к опустошению FQ (qne = 0). Если FQ не реализован, данный бит считывается, как сброшенный. Программное обеспечение супервизора должно обеспечивать, чтобы в пользовательском режиме данный бит всегда считывался, как сброшенный.

FSR_fp_condition_codes (fcc) – биты 11 и 10 содержат код условия FPU. Обновление данных битов выполняется с помощью инструкций сравнения с плавающей запятой (FCMP и FCMPE). Считывание и запись выполняется с помощью инструкций STFSR и LDFSR соответственно. Передача управления инструкциями FBfcc базируется на значении данного поля.

В таблице 5.3.11 f_{rs1} и f_{rs2} соответствуют регистру f одинарной, удвоенной или учетверенной точности, указанных в полях rs1 и rs2 инструкции. Знак вопроса «?» обозначает неупорядоченное отношение, которое имеет место быть, если либо f_{rs1}, либо f_{rs2} является NaN с уведомлением (SNaN) или NaN без уведомления (QNaN). Следует

отметить, что fcc не изменяется, если FCMP или FCMPE генерирует системное прерывание IEEE_754_exception.

Таблица 5.3.11 – Поле код условия операций с плавающей запятой (fcc) регистра FSR

fcc	Отношение
0	$f_{rs1} = f_{rs2}$
1	$f_{rs1} < f_{rs2}$
2	$f_{rs1} > f_{rs2}$
3	f_{rs1} ? f_{rs2} (не упорядочено)

FSR_accrued_exception (aexc) – биты 9–5 аккумулируют исключения операций с плавающей запятой стандарта IEEE_754, пока системные прерывания fp_exception запрещены через механизм поля TEM. Формат записи приведен в таблице 5.3.13. После выполнения FPop над полями TEM и cexc выполняется операция логического И. Если результат ненулевой, генерируется системное прерывание fp_exception; в противном случае, новое поле cexc через операцию логического ИЛИ заносится в поле aexc. Таким образом, пока системное прерывание маскировано, исключения аккумулируют в поле aexc.

FSR_current_exception (cexc) – биты 4–0 обозначают, что последняя выполненная инструкция FPop сгенерировала одно или более исключений операций с плавающей запятой стандарта IEEE_754. Отсутствие исключения сбрасывает соответствующий бит. Формат записи приведен в таблице 5.3.14.

Биты cexc устанавливаются при выполнении FPop, которые либо не вызывают системное прерывание, либо вызывают системное прерывание fp_exception с FSR.ftt = IEEE_754_exception. Рекомендуется, чтобы IEEE_754_exception, которые генерируют системные прерывания, вызвали установку только одного определенного бита в FSR.cexc, соответствующего обнаруженному исключению IEEE 754. Если выполнение FPop вызывает системное прерывание, отличное от fp_exception, обусловленного стандартом IEEE 754, то FSR.cexc остается неизменным.

Поля исключений с плавающей запятой

Поля текущих (cexc) и накопленных (aexc) исключений, а также маска разрешения системного прерывания (TEM), подразумевают порядок условий исключений операций с плавающей запятой (согласно стандарту ANSI/IEEE 754-1985), отображенный в таблицах 5.3.12 – 5.3.14.

Таблица 5.3.12 – Поля маски разрешения системного прерывания (TEM) регистра FSR

NVM	OFM	UFM	DZM	NXM
27	26	25	24	23

Таблица 5.3.13 – Поля накопленных битов исключений (aexc) регистра FSR

nva	ofa	ufa	dza	nxa
9	8	7	6	5

Таблица 5.3.14 – Поля текущих битов исключений (cexc) регистра FSR

nvc	ofc	ufc	dzc	nxc
4	3	2	1	0

FSR_invalid (nvc, nva) – операнд не подходит для выполнения операции. Например, деление 0 на 0 и вычитание ∞ из ∞ не допустимы. 1 = недопустимый операнд, 0 = допустимый операнд(-ы).

FSR_overflow (ofc, ofa) – округленный результат по абсолютному значению превышает максимальное нормализованное значение для заданного формата. 1 = переполнение, 0 = переполнения не наблюдается.

FSR_underflow (ufc, ufa) – округленный результат неточен и по абсолютному значению меньше минимального нормализованного значения для заданного формата. 1 = обратное переполнение, 0 = обратного переполнения не наблюдается.

Обратное переполнение ни при каких обстоятельствах не будет установлено, если точный неокругленный результат равен нулю. В противном случае возможны два варианта:

Если UFM = 0: биты ufc и ufa будут установлены, если точный неокругленный результат операции по абсолютному значению меньше наименьшего возможного нормализованного значения и корректно округленный результат неточен. Данные биты будут установлены, если точный неокругленный результат операции по абсолютному значению меньше наименьшего возможного нормализованного значения, но точный округленный результат является наименьшим нормализованным числом. Также всегда устанавливаются nxc и nxa.

Если UFM = 1: системное прерывание IEEE-754_exception произойдет, если точный неокругленный результат операции по абсолютному значению меньше наименьшего возможного нормализованного значения и корректно округленный результат неточен. Системное прерывание произойдет, если точный неокругленный результат операции по абсолютному значению меньше наименьшего возможного нормализованного значения, но точный округленный результат является наименьшим нормализованным числом.

FSR_division-by-zero (dzc, dza) – деление X на 0, где X субнормальное или нормализованное число. Следует отметить, что при делении 0 на 0 бит dzc не устанавливается. 1 = деление на ноль, 0 = без деления на ноль.

FSR_inexact (nxc, nxa) – округленный результат операции отличается от бесконечно точного результата. 1 = неточный результат, 0 = точный результат.

Соответствие FSR

В конкретной реализации поля TEM, cexc и aexc могут быть интегрированы в устройство одним из двух способов:

1) Реализовать все три поля в соответствии со стандартом ANSI/IEEE 754-1985;
2) Реализовать биты NXM, nxa и nxc этих трех полей в соответствии со стандартом ANSI/IEEE 754-1985. Интегрировать каждый из оставшихся битов трех полей:
   a. В соответствии со стандартом ANSI/IEEE;
   б. Как бит состояния, который может быть установлен программным обеспечением, которое вычисляет значение ANSI/IEEE бит. Для каждого бита, реализованного подобным образом, во-первых, исключение IEEE, соответствующее биту статуса, обязано всегда вызывать исключение (в особенности это относится к unfinished_FPop). В ходе выполнения обработчика системного прерывания бит в поле статуса может быть записан через задание соответствующего значения инструкции LDFSR. Во-вторых, бит состояния должен быть реализован таким образом, чтобы если его конкретное значение записывается с помощью инструкции LDFSR, то он будет считан с тем же значением при последующем выполнении STFSR.

Замечания по программированию

Программное обеспечение должно быть в состоянии имитировать весь FPU для обеспечения должной обработки системных прерываний операций с плавающей запятой исключений unimplemented_FPop, unfinished_FPop и IEEE_754_exception. Таким образом, прикладная программа пользователя всегда «видит» FSR, полностью соответствующий стандарту ANSI/IEEE 754-1985.

Очередь отложенных прерываний операций с плавающей запятой (FQ)

Очередь отложенных прерываний операций с плавающей запятой (FQ), если она представлена в реализации, содержит информацию о состоянии, достаточную для выполнения восстановления работы после отложенных системных прерываний операций с плавающей запятой.

Если инструкции с плавающей запятой выполняются параллельно с (асинхронно) целочисленными инструкциями в конкретной реализации, то в реализации обязана присутствовать очередь операций с плавающей запятой. Если инструкции с плавающей запятой выполняются синхронно с целочисленными инструкциями, реализация данной очереди является необязательной.

Считывание FQ можно выполнить при помощи привилегированной инструкции сохранения двойного слова данных очереди операций с плавающей запятой (STDFQ). В определенных реализациях может быть доступно считывание или запись при помощи привилегированных инструкций загрузки/сохранения удвоенных данных в альтернативное адресное пространство (LDDA, STDA) или при помощи инструкций считывания/записи вспомогательных регистров состояния (RDASR, WRASR).

Содержимое и операции с FQ зависят от реализации. Однако, если FQ представлен в устройстве, программное обеспечение супервизора должно быть в состоянии установить код операции (opf), вызвавшей исключение, её операнды и адрес из записи FQ. Данное условие также должно распространяться и на все остальные ожидающие обработки операции с плавающей запятой в очереди.

Если реализация не содержит FQ, бит qne в регистре FSR всегда сброшен, а выполнение инструкции STDFQ вызывает системное прерывание fp_exception с FSR.ftt = 4 (sequence_error).

5.4 Системные прерывания

Системное прерывание – это векторная передача управления супервизору через специальную таблицу системных прерываний, в которой содержатся первые 4 инструкции каждого обработчика системных прерываний. Базовый адрес таблицы устанавливается супервизором путем записи поля базового адреса таблицы системных прерываний (TBA) регистра состояния IU, называемого базовым регистром системных прерываний (TBR). Смещение в пределах таблицы определяется типом системного прерывания. Половина таблицы зарезервирована для системных прерываний аппаратного обеспечения, вторая половина зарезервирована для системных прерываний программного обеспечения, генерируемых инструкциями системных прерываний программного обеспечения (Ticc).

Системное прерывание аналогично непредвиденному вызову процедуры. Оно декрементирует указатель на текущее окно (CWP) до следующего регистрового окна, при этом аппаратное обеспечение записывает значения счетчиков команд (PC, nPC) на которых произошло прерывание в два локальных регистра нового окна. Обычно, обработчики системных прерываний также сохраняют значение PSR в другом локальном регистре, оставшиеся 5 локальных регистров нового окна можно свободно использовать.

Системное прерывание может быть вызвано инструкцией, исключением или внешним запросом на прерывание, напрямую не связанным с конкретной инструкцией. Перед выполнением каждой инструкции IU выбирает исключение или запрос на прерывание с наивысшим приоритетом и, если таковые имеются, вызывает системное прерывание.

Ошибка, «после которой невозможно восстановление в автоматическом режиме», вызывается при обнаружении неправильного функционирования аппаратного обеспечения, которое по своим свойствам препятствует восстановлению состояния процессора на момент возникновения ошибки. Т.к. состояние процессора не

восстанавливается, выполнение операции после такой ошибки не возобновляется. Ошибки, после которых невозможно восстановление в автоматическом режиме, обрабатываются в зависимости от реализации. Пример такой ошибки – ошибка по четности на шине.

5.4.1 Категории прерываний

Исключение или запрос на прерывание может вызвать следующие три категории системных прерываний:

- строгое;
- отложенное;
- прерывающее.

Строгое системное прерывание

Строгое прерывание вызывается конкретной инструкцией и происходит прежде, чем произойдет любое программно-видимое изменение состояния, вызванное системным прерыванием. При возникновении строгого прерывания в обязательном порядке соблюдаются следующие условия:

- PC, сохраненный в r[17] (локальный регистр 1), указывает на инструкцию, вызвавшую системное прерывание; nPC, сохраненный в r[18] (локальный регистр 2), указывает на следующую выполняемую инструкцию;
- выполнены инструкции, предшествующие инструкции, вызвавшей системное прерывание;
- инструкции расположенные после инструкции, вызвавшей системное прерывание, остаются невыполненными.

Отложенное системное прерывание

Отложенное прерывание также вызывается конкретной инструкцией, но, в отличие от строгого прерывания, оно может произойти после программно-видимого изменения состояния. Такое изменение состояния может быть обусловлено либо самой инструкцией, вызвавшей системное прерывание, либо одной или несколькими инструкциями, следующими за ней.

Отложенное прерывание может возникнуть через одну или более инструкций, выполненных после инструкции, вызывающей системное прерывание. Однако, отложенное прерывание обязано произойти до выполнения любой из инструкций, которая зависит от инструкции, вызывающей системное прерывание. То есть выполнение отложенного прерывания не может откладываться дольше, чем до инструкции, в которой заявлены исходные регистры, регистры назначения, коды условия или любые другие программно-видимые состояния устройства, которые могли бы быть модифицированы инструкцией, вызвавшей системное прерывание.

Отложенное прерывание не может быть отложено дольше, чем до строгого прерывания, кроме исключения операции с плавающей запятой или исключения сопроцессора, на которые данное утверждение не распространяется.

Наряду со спецификой конкретной реализации отложенных системных прерываний, в обязательном порядке должны существовать:

- инструкция, вызывающая исключение отложенного системного прерывания, которое потенциально может ожидать выполнения через механизм системного прерывания;
- возможность возобновления выполнения прерванного потока инструкций;

- привилегированные инструкции доступа к состоянию для эмулирования супервизором инструкции, вызывающей отложенное прерывание, и возобновления выполнения прерванного потока инструкций.

Следует отметить, что возможность возобновления выполнения может подразумевать эмулирование невыполненных к моменту отложенного прерывания инструкций (т.е. инструкций в очереди отложенных прерываний).

Сам факт наличия отложенных системных прерываний (и связанных с ними очередей отложенных системных прерываний) зависит от реализации. Если есть необходимость полностью исключить появление отложенных системных прерываний, то необходимо, чтобы все инструкции с плавающей запятой (и инструкций сопроцессора, если таковые имеются) и исключения ими сгенерированные осуществлялись синхронно с выполнением целочисленных инструкций. В таких реализациях неоправданно наличие очереди отложенных прерываний (например, FQ или CQ).

Прерывающее системное прерывание

Прерывающее системное прерывание не является ни строгим системным прерыванием, ни отложенным прерыванием. Управление прерывающими системными прерываниями осуществляется через поле уровня прерывания процессора (PIL) и поле разрешения системных прерываний (ET) регистра PSR.

Прерывающее системное прерывание может быть вызвано:

- внешним запросом на прерывание, не связанным непосредственно с предыдущей выполненной инструкцией;
- исключением, не связанным непосредственно с предыдущей выполненной инструкцией, или
- исключением, вызванным предыдущей выполненной инструкцией.

Подробнее каждый из трех приведенных случаев рассмотрен ниже.

Внешний запрос на прерывание может быть вызван появлением внешнего сигнала, напрямую не связанного с конкретным состоянием процессора или памяти. Например, установкой сигнала «Операция ввода-вывода завершена».

Прерывающее системное прерывание, вызванное исключением, не связанным непосредственно с предыдущей выполненной инструкцией, может быть вызвано обнаружением произвольного, зависящего от конкретной реализации, состояния процессора или системы. Например, исключение, сгенерированное управляющей логикой точки останова, зависящее от выбранной из памяти, но не выполненной инструкции или от истории выполнения операций процессора.

Прерывающее системное прерывание, вызванное исключением, обусловленным предыдущей выполненной инструкцией. Оно аналогично отложенному системному прерыванию в части того, что оно происходит после того как инструкции, следующие за инструкцией вызвавшей прерывание, изменят состояние процессора или памяти. Разница заключается в том, что инструкция, вызывающая такое прерывающее системное прерывание, не всегда возможно эмулировать, т.к. в реализации не сохраняется необходимое для этого состояние. Например, ошибка доступа к данным, защищенным кодами коррекции, сообщение о которой появилось после выполнения соответствующей инструкции загрузки.

5.4.2 Модели системных прерываний

Поскольку прикладная пользовательская программа «не видит» системные прерывания, пока через программное обеспечение супервизора не установлен пользовательский обработчик системных прерываний, и рассмотрение зависящих от конкретной реализации ошибок, «после которых невозможно восстановление в

автоматическом режиме», может отличаться в разных системах, архитектура SPARC позволяет в реализации иметь альтернативные модели системных прерываний для отдельных типов исключений.

В частности, конкретное исключение может привести к срабатыванию строгого, отложенного или прерывающего прерываний, в зависимости от реализации. (Следует отметить, что внешние запросы на прерывание, по определению, всегда вызывают прерывающие системные прерывания).

Однако, для поддержки пользовательских обработчиков системных прерываний и виртуализируемых инструкций, архитектура SPARC определяет модель системных прерываний по умолчанию, которая должна присутствовать во всех реализациях.

Модель системных прерываний по умолчанию

В SPARC, согласно модели системных прерываний по умолчанию, все системные прерывания должны являться строгими, за исключением четырех случаев:

- исключений операций с плавающей запятой и сопроцессора (fp_exception, cp_exception), которые могут быть отложенными;
- исключений, «после которых невозможно восстановление в автоматическом режиме», которые могут быть отложенными или прерывающими, в зависимости от реализации;
- исключений, вызванных после выполнения первичного доступа инструкцией загрузки / сохранения с множественным доступом (загрузка / сохранение двойного слова данных, атомарная загрузка / сохранение и SWAP), которые могут быть прерывающими, если они вызваны исключением, «после которого не возможно восстановление в автоматическом режиме». Таким образом, системное прерывание при выполнении второго доступа к памяти может возникать после изменения состояния процессора или памяти, вызванного выполнением первого доступа;
- исключений, вызванных событиями, не связанными с потоком инструкций, которые являются прерывающими и поэтому не могут принадлежать к классу строгих.

Данные условия зависят от реализации.

Если на текущий момент другое исключение с плавающей запятой является отложенным, то попытка выполнить инструкцию с плавающей запятой (FPop, FBfcc или загрузка / сохранение с плавающей запятой) вызовет или станет причиной возникновения ожидающего выполнения системного прерывания fp_exception. Аналогично, если исключение сопроцессора в настоящее время отложено, попытка выполнить другую команду сопроцессора (CPop, CBccc или загрузка / сохранение сопроцессора) вызовет ожидающее выполнения системное прерывание cp_exception.

Замечание по реализации

Чтобы обеспечить возможность завершать пользовательский процесс при возникновении исключения, «после которого не возможно восстановление в автоматическом режиме», которое может вызвать отложенное или прерывающее системное прерывание, в реализации должна присутствовать одна или несколько инструкций, вызывающих обработку ожидающего выполнения условия, как системного прерывания. Например, ожидающее обработки исключение операции с плавающей запятой может быть трансформировано в вызов системного прерывания fp_exception путем выполнения любой FPop, загрузки / сохранения с плавающей запятой, включая STFSR, или команды FBfcc.

Усовершенствованная модель системных прерываний

Прикладные программы пользователя не «видят» системные прерывания до тех пор, пока они в явном виде не запрашивают права на их обработку. Таким образом, архитектура SPARC предусматривает зависящие от реализации улучшения производительности к модели системных прерываний по умолчанию, и позволяет создавать реализации с усовершенствованной моделью системных прерываний, в которой некоторые системные прерывания могут являться отложенными или прерывающими, вместо того, чтобы быть строгими.

В частности, должно ли конкретное системное прерывание быть строгим или нет, зависит от того, запрашивает ли пользовательская прикладная программа у супервизора, возможность обработки исключения, генерирующего системное прерывание. Запрос на регистрацию пользовательского обработчика системного прерывания подразумевает, что когда происходит данное конкретное системное прерывание, супервизор предаст управление пользовательской прикладной программе по заранее определенному адресу, указанному программой.

В усовершенствованной модели системных прерываний SPARC утверждено, что отдельное системное прерывание обязано придерживаться модели системных прерываний по умолчанию, если пользовательская прикладная программа через супервизор устанавливает собственный обработчик этого системного прерывания. Таким образом, аппаратное обеспечение должно быть в состоянии обработать исключение как системное прерывание одного из следующих типов:

- строгое прерывание;
- отложенное прерывание в случае исключений операций с плавающей запятой или сопроцессора;
- отложенное или прерывающее системное прерывание в случае зависящих от реализации исключений, «после которых невозможно восстановление в автоматическом режиме».

Если пользовательская прикладная программа не устанавливает собственный обработчик системного прерывания (через программное обеспечение супервизора) для данного системного прерывания, такое системное прерывание может являться отложенным или прерывающим. Тип конкретного системного прерывания (строгое, отложенное или прерывающее) должен оставаться неизменным для всех вхождений связанного исключения для конкретной программы или процесса. Поэтому, если пользовательская программа для данного системного прерывания устанавливает свой собственный обработчик, она должна сделать это до первого срабатывания этого системного прерывания в программе.

Инструкция, осуществляющая доступ к состоянию аппаратного обеспечения, которое контролирует, является ли конкретное прерывание строгим или нет, конкретное системное прерывание, должна быть привилегированной (например, инструкция загрузки / сохранения в альтернативное адресное пространство, привилегированная инструкция считывания / записи ASR, считывания / записи поля ver регистра PSR).

Реализация конкретного системного прерывания, согласно усовершенствованной модели системных прерываний, является зависимой от супервизора и реализации.

Замечания по программированию

Запрос супервизору о регистрации пользовательского обработчика системного прерывания может оговаривать, что соответствующее исключение обрабатывается пользовательской программой невозобновляемым образом. В этом случае супервизор имеет право не требовать, чтобы соответствующее системное прерывание было строгим.

После строгого системного прерывания супервизор:

- возвращается к инструкции, вызвавшей системное прерывание, и повторно ее выполняет (PC ← предыдущий PC, nPC ← предыдущий nPC), или
- эмулирует инструкцию, вызвавшую системное прерывание, и возвращается к инструкции (по времени), которая выполнялась после инструкции, вызвавшей системное прерывание (PC ← предыдущий nPC, nPC ← предыдущий nPC + 4), или
- завершает программу или процесс, связанный с системным прерыванием.

После отложенного системного прерывания супервизор:

- эмулирует инструкцию, вызвавшую ошибку, эмулирует или инициирует выполнение других отложенных инструкций, находившихся в соответствующей очереди отложенных системных прерываний, а также возвращает управление инструкции, при выполнении которой сработало отложенное прерывание, или
- завершает программу или процесс, связанный с системным прерыванием.

После прерывающего системного прерывания супервизор:

- возвращается к инструкции, при выполнении которой сработало системное прерывание (PC ← предыдущий PC, nPC ← предыдущий nPC), или
- завершает программу или процесс, связанный с системным прерыванием.

При наличии пользовательского обработчика системных прерываний супервизор может передать ему запись, содержащую адрес инструкции, вызвавшей системное прерывание, адреса PC и nPC системного прерывания, а также любое связанное состояние, необходимое для эмуляции инструкции. Пользовательский обработчик системного прерывания может возобновить выполнение по альтернативному адресу или возвратить управление супервизору. Супервизор может обрабатывать любые другие оставшиеся записи в соответствующей очереди отложенных системных прерываний и, в конечном итоге, может вернуться к инструкциям по адресам PC и nPC системного прерывания.

Управление системными прерываниями

Управление системными прерываниями осуществляется через несколько регистров: исключения и запросы прерываний с помощью поля разрешения системных прерываний (ET) в PSR, запросы на прерывание с помощью поля уровня прерывания процессора (PIL) в PSR и исключения операций с плавающей запятой с помощью маски разрешения системных прерываний (TEM) в FSR.

Управление через ET и PIL

Бит ET в PSR должен быть установлен для штатного срабатывания системных прерываний. Пока ET = 1, IU – между выполнением инструкций – устанавливает приоритеты ожидающих выполнения исключений и запросов прерываний в соответствии с таблицей 5.4.1. В каждый определенный момент времени как системное прерывание обрабатывается исключение или запрос прерывания с наивысшим приоритетом. Если ожидающих обработки исключений или запросов прерываний несколько, SPARC предполагает, что запросы прерываний с более низким приоритетом будут сохраняться, а исключения с более низким приоритетом будут возникать вновь, если инструкция, инициировавшая исключение, будет повторно выполнена.

В случае запросов на прерывание IU сравнивает уровень запроса на прерывание (bp_IRL) со значением поля уровня прерывания процессора (PIL) регистра PSR. Если bp_IRL превышает PIL, или bp_IRL = 15 (немаскируемый), процессор принимает запрос на прерывание в связи с возникновением системного прерывания – при этом предполагается, что ошибки высокого приоритета, ожидающие выполнения,

отсутствуют. Быстрота реакции процессора на запрос прерывания, а также метод удаления запроса на прерывание, зависят от реализации.

Пока ET = 0:

- прерывающие системные прерывания не срабатывают, все запросы на прерывание игнорируются, даже если bp_IRL = 15;
- если срабатывает строгое прерывание или при попытке выполнения инструкции, вызывающей отложенное прерывание, при наличии ожидающего выполнения исключения, вызвавшего отложенное системное прерывание, процессор останавливается и переходит в состояние error_mode;
- если срабатывает отложенное системное прерывание, вызванное инструкцией, которая начала выполняться при ET = 0, отложенное системное прерывание останавливает процессор и переводит его в состояние error_mode;
- игнорируется любое исключение, породившее отложенное системное прерывание, вызванное инструкцией, которая начала выполняться при ET = 1.

Управление через TEM

Управление исключениями операций с плавающей запятой IEEE_754_exceptions выполняется через поле маски разрешения системного прерывания (TEM) FSR, доступное пользователю. Если конкретный бит TEM установлен, соответствующее исключение IEEE_754_exception вызывает системное прерывание fp_exception.

Если конкретный бит TEM сброшен, соответствующее исключение IEEE_754_exception не вызывает системное прерывание fp_exception. Вместо этого факт возникновения исключения IEEE_754_exception записывается в поле накопленных исключений FSR (aexc).

Если исключение операции с плавающей запятой типа IEEE_754_exception вызывает системное прерывание fp_exception, регистр назначения f, поля fcc и aexc не меняются. Однако, если исключение IEEE_754_exception не вызывает системного прерывания, то регистр f, поля fcc и aexc обновляются до их новых значений.

5.4.3 Идентификация системных прерываний

Супервизор инициализирует поле базового адреса системных прерываний (TBA) базового регистра системных прерываний (TBR) двадцатью старшими битами адреса таблицы системных прерываний.

Тип системного прерывания (tt)

При возникновении системного прерывания (за исключением внешнего запроса на сброс) значение, идентифицирующее системное прерывание, записывается аппаратным обеспечением в 8-битовое поле tt регистра TBR. После этого управление передается в таблицу системных прерываний супервизора на адрес, содержащийся в 32-битовом TBR. Поскольку младшие 4 бита TBR сброшены, каждая запись в таблице системных прерываний содержит первые 16 байт (4 слова) соответствующего сортировщика системных прерываний.

Поле tt предусматривает 256 типов системных прерываний – половина для системных прерываний аппаратного обеспечения и половина для системных прерываний программного обеспечения. Значения 0–0x7F зарезервированы для системных прерываний аппаратного обеспечения. Значения 0x80–0xFF зарезервированы для системных прерываний программного обеспечения (системные прерывания, вызванные выполнением инструкцией Ticc). Распределение значений tt по системным прерываниям приведено в таблице 5.4.1. Следует отметить, что поле tt остается действительным до возникновения следующего системного прерывания.

Поскольку распределение значений исключений и запросов на прерывание конкретным адресам вектора системных прерываний и по уровням приоритетов невидимо для прикладной программы пользователя, реализации определяют дополнительные системные прерывания аппаратного обеспечения.

В частности, значения tt 0x60–0x7F зарезервированы для исключений, в зависимости от реализации.

Значения в диапазоне от 0 до 0x5F, не присвоенные в таблице 5.4.1, зарезервированы для последующих версий архитектуры.

Режим ошибки

Если при ET = 0 возникает системное прерывание, процессор входит в режим error_mode. В этом случае реализация должна предоставлять максимальное количество данных о состоянии процессора. Стандартные действия в случае возникновения системного прерывания (например, уменьшение CWP и сохранение информации о состоянии в локальных регистрах) не должны выполняться при входе в режим error_mode. В частности, запись в поле tt регистра TBR выполняется при переходе в режиме error_mode только в случае выполнения инструкции RETT, вызывающей системное прерывание при ET = 0. В этом случае запись в поле tt выполняется для обозначения типа исключения, вызванного инструкцией RETT.

Действия после входа в режим error_mode зависят от реализации; обычно процессор запускает внешний сброс, вызывая системное прерывание при сбросе (см. ниже).

Системное прерывание при сбросе

Системное прерывание при сбросе (reset) запускается внешним запросом на сброс и вызывает передачу управления по адресу 0.

Программное обеспечение супервизора не должно предполагать, что после системного прерывания reset будет инициализировано какое-либо конкретное состояние процессора или памяти, за исключением битов ET и S регистра PSR.

При возникновении системного прерывания reset запись в поле tt не выполняется, при этом в данном поле отображается значение предыдущего системного прерывания. В случае внешнего запроса на сброс, вызванного подачей питания на схему, значение поля tt не определено.

Приоритеты системного прерывания

В таблице 5.4.1 для исключений приводятся присвоенные им значения tt, а также указывается относительная приоритетность исключений и запросов на прерывание. Приоритет 1 является наивысшим, приоритет 31 – самым низким; то есть, если X < Y, обрабатывается ожидающее выполнения исключение или запрос на прерывание приоритета X, а не ожидающее выполнения исключение или запрос на прерывание приоритета Y.

Следует отметить, что конкретные приоритеты системных прерываний зависят от реализации, поскольку последующие версии архитектуры могут определять новые системные прерывания, при этом разные реализации могут определять разные системные прерывания. Однако, значения tt для исключений и запросов на прерывание, представленные в таблице 5.4.1, сохраняются для каждой реализации.

Таблица 5.4.1 – Приоритет исключений и запросов на прерывание и значения tt

Исключение или запрос на прерывание	Приоритет	tt
reset	1	(см. по тексту)
data_store_error	2	0x2B
instruction_access_MMU_miss	2	0x3C
instruction_access_error	3	0x21
r_register_access_error	4	0x20
instruction_access_exception	5	0x01
privileged_instruction	6	0x03
illegal_instruction	7	0x02
fp_disabled	8	0x04
cp_disabled	8	0x24
unimplemented_FLUSH	8	0x25
watchpoint_detected	8	0x0B
window_overflow	9	0x05
window_underflow	9	0x06
mem_address_not_aligned	10	0x07
fp_exception	11	0x08
cp_exception	11	0x28
data_access_error	12	0x29
data_access_mmu_miss	12	0x2C
data_access_exception	13	0x09
tag_overflow	14	0x0A
division_by_zero	15	0x2A
trap_instruction	16	0x80–0xFF
interrupt_level_15	17	0x1F
interrupt_level_14	18	0x1E
interrupt_level_13	19	0x1D
interrupt_level_12	20	0x1C
interrupt_level_11	21	0x1B
interrupt_level_10	22	0x1A
interrupt_level_9	23	0x19
interrupt_level_8	24	0x18
interrupt_level_7	25	0x17
interrupt_level_6	26	0x16
interrupt_level_5	27	0x15
interrupt_level_4	28	0x14
interrupt_level_3	29	0x13
interrupt_level_2	30	0x12
interrupt_level_1	31	0x11
исключения, в зависимости от реализации	зависит от реализации	0x60–0x7F

5.4.4 Описание системного прерывания

Если ET = 1 при возникновении системного прерывания:
- Запрещается выполнение системных прерываний: ET ← 0.
- Сохраняется текущий режим пользователь / супервизор: PS ← S.
- Режим пользователь / супервизор изменяется на режим супервизора: S ← 1.

- Осуществляется переход от текущего регистрового окна к новому окну: CWP ← ((CWP – 1) по модулю NWINDOWS) [без проверки на предмет переполнения окна].

- Счетчики команд системного прерывания сохраняются в локальные регистры 1 и 2 нового окна: r[17] ← PC, r[18] ← nPC.

- В поле tt записывается конкретное значение, обозначающее исключение или запрос прерывания, кроме случаев, описанных выше в подпунктах «Системное прерывание при сбросе» и «Режим ошибки».

- Если системное прерывание является системным прерыванием при сбросе, управление передается по адресу «0»: PC ← 0, nPC ← 4.

Если системное прерывание не является системным прерыванием при сбросе, управление передается в таблицу системных прерываний: PC ← TBR, nPC ← TBR + 4.

Если ET = 0 и возникает строгое прерывание, процессор входит в состояние error_mode и останавливается. Если ET = 0 и возникает либо запрос на прерывание, либо прерывающая или отложенная исключительная ситуация, то они игнорируются.

Замечание по реализации

При входе в режим error_mode изменению подлежит минимальное количество данных по состоянию процессора. Если процессор в режиме error_mode, его перезапуск выполняется с помощью внешнего запроса на сброс, что вызывает системное прерывание reset.

Замечание по программированию

Обработчик системных прерываний должен осуществлять с полями регистра PSR только обратимые изменения (ET, CWP) или не должен производить над PSR действий до его сохранения.

5.4.5 Описание отдельных исключений / прерываний

В таблице 5.4.2 приводится описание различных исключений и запросов на прерывание, а также условия, при которых они возникают.

Поскольку точные условия для некоторых из этих исключений зависят от реализации, некоторые определения неточны. В частности, определение терминов «безусловная ошибка» и «блокирующая ошибка» зависит от реализации. Более того, генерация процессором SPARC исключений данных типов зависит от реализации.

Таблица 5.4.2 – Описание исключений

Наименование исключения	Описание исключения
reset	Данное системное прерывание вызывается внешним запросом на сброс. При этом процессор начинает выполнение инструкций по виртуальному адресу 0. Программное обеспечение супервизора не может с уверенностью полагать, что после системного прерывания reset процессор или память будут инициализированы, за исключением битов ET и S регистра PSR.
data_store_error	Возникла безусловная ошибка при сохранении данных в память (например, ошибка по четности шины при сохранении данных из буфера сохранения)

Продолжение таблицы 5.4.2

Наименование исключения	Описание исключения
instruction_access_MMU_miss	Неудачное обращение в MMU при доступе к инструкции из памяти. Например, PDC или TLB не содержали трансляцию виртуального адреса
instruction_access_error	Возникла безусловная ошибка при доступе к инструкции (например, ошибка, выявленная контролем по чётности, при доступе к кэшу инструкции)
r_register_access_error	Возникла безусловная ошибка при доступе к регистру r (например, ошибка, выявленная контролем по чётности, при считывании регистра r)
instruction_access_exception	Возникла блокирующая ошибка при доступе к инструкции (например, MMU указывает о недействительности переадресации страниц или наличии защиты от считывания)
privileged_instruction	Попытка выполнения привилегированной инструкции, когда бит S регистра PSR сброшен
illegal_instruction	Попытка выполнения инструкции с нереализованным кодом операции, или инструкции UNIMP, или инструкции, которая в результате должна привести к недопустимому состоянию процессора (например, запись недопустимого CWP в PSR). Следует отметить, что нереализованная инструкция FPop и нереализованная инструкция CPop генерируют системные прерывания fp_exception и cp_exception, а также что разработчик может организовать генерацию системного прерывания unimplemented_FLUSH при выполнении нереализованной инструкции FLUSH вместо системного прерывания illegal_instruction
unimplemented_FLUSH	Была предпринята попытка выполнить инструкцию FLUSH, семантика которой не полностью реализована в аппаратном обеспечении. Использование данного системного прерывания зависит от реализации
watchpoint_detected	Адрес памяти извлечения инструкции или адрес памяти загруженных/сохраненных данных соответствует содержимому предварительно загруженного, в зависимости от реализации, регистра «точки наблюдения». Генерация процессором SPARC ошибок watchpoint_detected зависит от реализации
fp_disabled	Попытка выполнения инструкции FPop, FBfcc или инструкции загрузки/сохранения с плавающей запятой при сброшенном бите EF регистра PSR или при отсутствии FPU
cp_disabled	Попытка выполнения инструкции CPop, CBccc или инструкции загрузки/сохранения сопроцессора при отсутствии сопроцессора
window_overflow	Переполнение окна. Инструкция SAVE пыталась заставить CWP указывать на окно, отмеченное в WIM как недопустимое
window_underflow	Обратное переполнение окна. Инструкция RESTORE или RETT пыталась заставить CWP указывать на окно, отмеченное в WIM как недопустимое

Окончание таблицы 5.4.2

Наименование исключения	Описание исключения
mem_address_not_aligned	Инструкция загрузки/сохранения сгенерировала адрес памяти, который не был правильно выровнен в соответствии с инструкцией, либо инструкция JMPL или RETT сгенерировала адрес, не выравниваемый по границе слова
fp_exception	Инструкция FPop сгенерировала исключение IEEE_754_exception, при этом соответствующий бит маски разрешения системного прерывания (TEM) был установлен или FPop не реализована, или FPop не выполнена, или возник ряд ошибок или аппаратная ошибка в FPU. Тип исключения операции с плавающей запятой кодируется в поле ftt FSR
cp_exception	Инструкция сопроцессора сгенерировала исключение
data_access_error	Возникла безусловная ошибка при доступе к загруженным/сохраненным данным из/в память (например, ошибка, выявленная контролем по чётности, при доступе к кэшу данных, или неисправимая ошибка в памяти, использующей корректирующие кодирование)
data_access_MMU_miss	Неудачное обращение в MMU при доступе загрузки/сохранения из/в память. Например, PDC или TLB не содержали трансляцию виртуального адреса
data_access_exception	Возникла блокирующая ошибка при попытке доступа загрузки/сохранения данных (например, MMU указывает о недействительности переадресации страниц или наличии защиты от считывания)
tag_overflow	Выполнена инструкция TADDccTV или TSUBccTV, при этом возникло арифметическое переполнение или, по крайней мере, один из битов метки компонентов операции был не равен нулю
division_by_zero	Попытка целочисленной инструкции деления выполнить деление на ноль
trap_instruction	Выполнена инструкция Ticc, при этом условие возникновения системного прерывания выполняется
interrupt_level_n	В IU был передан на рассмотрение внешний запрос на прерывание (bp_IRL) с уровнем n, в то время как ET = 1 и ((bp_IRL = 15) или (bp_IRL > PIL))

6 Введение в систему команд

Инструкции, извлекаемые процессором из памяти, могут быть выполнены, аннулированы или прерваны. Кодирование инструкций осуществляется в трех 32-битовых форматах с подразделением на шесть основных категорий. Есть 72 базовые инструкции операций. Их подробное описание приведено в приложении А.

6.1 Выполнение инструкций

Считывание инструкции выполняется из памяти по адресу, указанному счетчиком команд (PC). После этого инструкция выполняется или не выполняется, в зависимости от того, является предыдущая инструкция аннулирующей или нет (см. ниже). Инструкция может генерировать системное прерывание при обнаружении исключительной ситуации, вызванной самой инструкцией (строгое системное прерывание), предыдущей инструкцией (отложенное системное прерывание), внешним прерыванием (прерывающее системное прерывание) или внешним запросом на сброс. Если инструкция выполняется, может измениться состояние процессора и/или памяти, видимое для программы.

Если инструкция прерывается, управление направляется в таблицу системных прерываний по адресу, указанному в базовом регистре системных прерываний (TBR). Если инструкция не прерывается, следующее значение счетчика команд (nPC) копируется в PC, при этом nPC увеличивается на 4 (переполнение, если оно имеет место быть, игнорируется). Если инструкция является инструкцией передачи управления, процессор записывает адрес назначения в nPC. Таким образом, два счетчика команд обеспечивают функционирование модели отсроченного ветвления.

Для доступа к каждой инструкции и ко всем обычным данным IU добавляет к 32-битовому адресу памяти 8-битовый идентификатор адресного пространства (ASI). ASI кодирует режим процессора: режим супервизора или режим пользователя; а также осуществляется ли доступ к инструкции или доступ к данным. Также реализованы привилегированные инструкции загрузки/сохранения в альтернативное адресное пространство (см. ниже), которые обеспечивают доступ к данным по адресу с произвольным ASI.

Замечание по реализации

Время выполнения инструкции и степень параллелизма их выполнения зависят от реализации. Взаимоотношения между PC, nPC и аппаратным обеспечением, которое извлекает и декодирует инструкции, также зависят от реализации. При отсутствии системных прерываний изменения регистров и состояния памяти, видимые для программы, должны быть аналогичны изменениям при выполнении программой операций в соответствии с последовательной моделью.

6.1.1 Форматы инструкций

Кодирование инструкций выполняется в трех основных 32-битовых форматах, приведенных на рисунке 6.1.1. Также существует несколько вспомогательных форматов – см. приложение А «Описание инструкций».

Формат 1 (op = 1): CALL

op	disp30
31 30	29 0

Формат 2 (op = 0): SETHI и операции ветвления (Bicc, FBfcc, CBccc)

op	rd		op2	imm22
op	a	cond	op2	disp22
31 30	29 28	25	24 22	21 0

Формат 3 (op = 2 or 3): Остальные инструкции

op	rd	op3	rs1	i = 0	asi	rs2
op	rd	op3	rs1	i = 1	simm13	
op	rd	op3	rs1	opf		rs2
31 30	29 25	24 19	18 14	13	12 5	4 0

Рисунок 6.1.1 – Форматы инструкций

6.1.2 Поля инструкций

Поля инструкций интерпретируются следующим образом:

«op» и «op2»

С помощью этих двух- и трех-битовых полей кодируются три основных формата и инструкции формата 2, согласно таблицам 6.1.1 и 6.1.2.

Таблица 6.1.1 – Кодирование op (все форматы)

Формат	op	Инструкции
1	1	CALL
2	0	Bicc, FBfcc, CBccc, SETHI
3	3	Инструкции памяти
	2	Арифметика, логика, смещение и др.

Таблица 6.1.2 – Кодирование op2 (формат 2)

op2	Инструкции
0	UNIMP
1	не реализовано
2	Bicc
3	не реализовано
4	SETHI
5	не реализовано
6	FBfcc
7	CBccc

«rd»

Данное 5-битовое поле является адресом назначения (или исходным адресом) регистров r, f или сопроцессора в арифметических инструкциях/инструкциях загрузки (или сохранения). При выполнении инструкции чтения/записи двойного слова (или данных учетверенной точности), один младший значащий бит (или два) не используется и должен быть передан в сброшенном состоянии.

«a»

Бит «a» в инструкции перехода аннулирует выполнение следующей инструкции, если переход условный и не выполняется, если он безусловный и выполняется.

«cond»

Данное 4-битовое поле определяет код(-ы) условия(-й), проверяемый(-е) в инструкции перехода.

«imm22»

Данное 22-битовое поле является константой, которую инструкция SETHI помещает в старшие разряды регистра назначения.

«disp22» и «disp30»

Данные 30- и 22-битовые поля являются выравненными по границе слова значениями со знаком, которые модифицируют PC в инструкциях вызова процедуры или перехода соответственно.

«op3»

Данное 6-битовое поле (совместно с первым битом поля op) кодирует инструкции формата 3.

«i»

Бит i выбирает второй компонент операции ALU при выполнении (целочисленных) арифметических инструкций и инструкций загрузки/сохранения. Если i = 0, компонент операции – r[rs2]. Если i = 1, компонент операции – simm13 с добавлением знака в биты с тринадцатого по тридцать второй.

«asi»

Данное 8-битовое поле является идентификатором адресуемого пространства, который задается при выполнении инструкции загрузки/сохранения.

«rs1»

Данное 5-битовое поле является адресом первого исходного компонента операции регистров r, f или сопроцессора. При выполнении инструкции чтения/записи двойного слова (или данных учетверенной точности), один младший значащий бит (или два) не используется и должен быть передан в сброшенном состоянии.

«rs2»

Данное 5-битовое поле является адресом второго исходного компонента операции регистров r, f или сопроцессора при i = 0. При выполнении инструкции чтения/записи двойного слова (или данных учетверенной точности), один младший значащий бит (или два) не используется и должен быть передан в сброшенном состоянии.

«simm13»

Данное поле является 13-битовой константой со знаком, которое используется как второй компонент операции ALU при выполнении (целочисленной) арифметической инструкции или инструкции загрузки/сохранения при i = 1.

«opf»

Данное 9-битовое поле кодирует инструкцию операции с плавающей запятой (FPop) или инструкцию операции сопроцессора (CPop).

6.2 Категории инструкций

Инструкции SPARC можно сгруппировать по шести категориям: загрузка / сохранение, целочисленная арифметика, передача управления (CTI), считывание / запись регистра управления, операции с плавающей запятой и операции сопроцессора.

6.2.1 Инструкции загрузки/сохранения

Инструкции загрузки/сохранения – это единственные инструкции, имеющие доступ к памяти. Инструкции загрузки и сохранения используют два регистра r или один регистр r и simm13 для расчета 32-битового адреса памяти с выравниванием по границе байта. IU добавляет к данному адресу ASI, который кодирует режим процессора: режим супервизора или режим пользователя, а также доступ: доступ к инструкции или доступ к данным.

В поле назначения инструкции загрузки/сохранения указывается регистр r, f или сопроцессора, из которого данные передаются для сохранения или в который принимаются загруженные данные.

Целочисленные инструкции загрузки и сохранения поддерживают байтовый (8-битовый) доступ, доступ к полуслову (16-битовый), слову (32-битовый) и двойному слову (64-битовый). Инструкции загрузки и сохранения с плавающей запятой и сопроцессора поддерживают доступы к слову и двойному слову.

Замечание по программированию

Когда i = 1 и rs1 = 0 доступ к любой области младших или старших 4 Кбайт адресуемого пространства выполняется без использования регистра, хранящего адрес.

Ограничения по выравниванию

Доступы к полусловам должны быть выровнены по границе в 2 байта, доступы к словам (включая выборку инструкций) должны быть выровнены по границе в 4 байта, доступы к двойным словам должны быть выровнены по границе в 8 байт. Инструкция загрузки или сохранения по не выровненному адресу генерирует системное прерывание mem_address_not_aligned.

Принятые правила адресации

Архитектура SPARC использует порядок следования «сначала старший байт»: адрес двойного слова, слова или полуслова является адресом старшего значащего байта в адресе. Увеличение адреса обычно обозначает уменьшение значимости сегмента, к которому выполняется доступ. Правила адресации представлены на рисунке 6.2.1 и определяются следующим образом:

- Байты. Для байтовой инструкции загрузки/сохранения доступ к старшему значащему байту слова (биты 31 – 24) выполняется, если биты адреса [1:0] = 0, доступ к младшему значащему байту (биты 7 – 0) выполняется, если биты адреса [1:0] = 3.
- Полуслова. Для инструкции загрузки/сохранения полуслова доступ к старшему значащему полуслову (биты 31 – 16) выполняется, если биты адреса [1:0] = 0, доступ к младшему значащему полуслову выполняется, если биты адреса [1:0] = 2.
- Двойное слово. Для инструкции загрузки/сохранения двойного слова доступ к старшему значащему слову (биты 63 – 32) выполняется, если биты адреса [2:0] = 0, доступ к младшему значащему слову (биты 31 – 0) выполняется, если биты адреса [2:0] = 4.

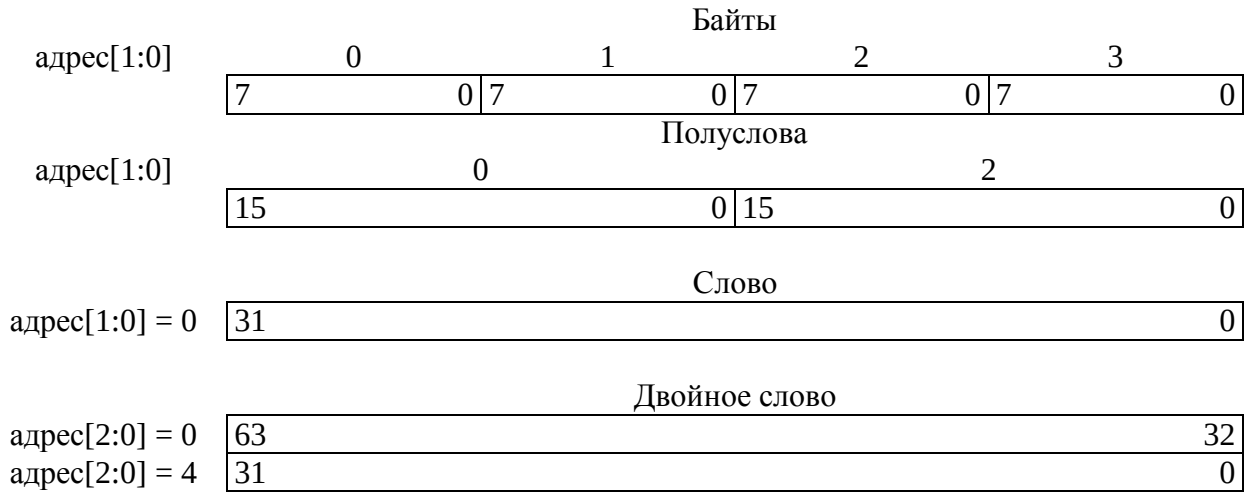


Рисунок 6.2.1 – Принятые правила адресации

Идентификаторы адресного пространства (ASI)

Стандартная инструкция загрузки/сохранения обеспечивает ASI – 0x0A или 0x0B для доступа к данным, в зависимости от режима процессора (пользователь/супервизор). Однако привилегированные инструкции загрузки из альтернативного адресного пространства и сохранения в альтернативное адресное пространство могут в явном виде задавать идентификатор адресного пространства, используя поле asi инструкций.

Назначение ASI приведено в таблице 6.2.1.

Таблица 6.2.1 – Идентификаторы адресных пространств

ASI	Адресное пространство
0x00 – 0x07	зависит от реализации
0x08	инструкция пользователя
0x09	инструкция супервизора
0x0A	данные пользователя
0x0B	данные супервизора
0x0C – 0xFF	зависит от реализации

Замечание по реализации

Назначение альтернативных адресных пространств зависит от реализации. Осуществляется ли декодирование всех восьми доступных бит ASI, также зависит от реализации.

Отдельная память инструкций

В архитектуре SPARC оговаривается, что инструкции сохранения имеют доступ к памяти, из которых была осуществлена их выборка. Тем не менее, реализация может явно разделять инструкции и данные на независимые памяти инструкций и данных (кэши), обычно такая архитектура называется «гарвардской» или архитектурой с «раздельными кэшами инструкций и данных».

Если программа использует самомодифицирующийся код, он должен выполнять FLUSH инструкцию для каждого модифицированного слова инструкции (или организовывать обращения к супервизору посредством системных прерываний, приводящие к тому же эффекту), обеспечивая сброс модифицированной инструкции в память. В большинстве случаев после сохранения инструкции должно следовать выполнение FLUSH, прежде чем будет осуществлена безопасная выборка новой инструкции из потока инструкций (см. «Определение инструкций» из приложения А).

Ввод-вывод

Архитектура SPARC предполагает, что доступ к регистрам области ввода-вывода выполняется с помощью инструкций загрузки/сохранения в альтернативные адресные пространства, стандартных инструкций загрузки/сохранения, инструкций сопроцессора или инструкций считывания/записи вспомогательных регистров состояния (RDASR, WRASR). В первом случае доступ к регистрам области ввода-вывода может осуществлять только супервизор. В последнем случае возможность выполнения доступа к регистрам области ввода-вывода из пользовательского кода зависит от реализации. Адреса и содержимое регистров области ввода-вывода зависят от реализации.

6.2.2 Целочисленные арифметические инструкции

Инструкции целочисленной арифметики обычно используют триаду адресов регистров, они вычисляют результат, являющийся функцией двух исходных операндов, и либо записывают результат в регистр назначения r[rd], либо отбрасывают его. Одним из исходных операндов всегда является r[rs1]. Второй исходный операнд зависит от состояния бита i в инструкции: если i = 0, компонент операции – r[rs2], если i = 1, то simm13 с дополнением разрядом знака до размера в 32 бита.

При считывании r[0] выводится нулевое значение. Если в поле назначения указывается запись в r[0], регистры r не изменяются и результат операции отбрасывается.

Установка кодов условий

Большинство данных инструкций доступны в двух версиях: в первой из них целочисленные коды результата (icc) устанавливаются как побочное действие, во второй версии коды результата не изменяются. Не требуется какой-либо специальной инструкции сравнения целочисленных значений, поскольку данный результат легко достигается использованием инструкции «вычитания с установкой кодов условий» (SUBcc).

Инструкции сдвига

При выполнении инструкции сдвига содержимое регистра r смещается влево или вправо на постоянную или переменную величину. Выполнение инструкции сдвига не оказывает влияния на коды условий.

Установка старших 22 битов

При выполнении инструкции «установки старших 22 битов регистра r» (SETHI) 22-битовая константа записывается из инструкции в старшие биты регистра назначения. 10 младших бит сбрасываются, коды условий не изменяются. Данная инструкция, главным образом, используется при формировании 32-битовой константы в регистре.

Целочисленное умножение/деление

Целочисленные инструкции умножения осуществляют перемножение двух 32-битных операндов с получением 64-битного результата. Целочисленные инструкции деления выполняют деление 64-разрядного делимого на 32-разрядный делитель, с получением 32-разрядного результата. Есть версии инструкций с установкой кодов условий и без оной. При делении на ноль срабатывает системное прерывание division_by_zero.

Сложение/вычитание с меткой

При выполнении инструкций сложения/вычитания с меткой используется формат данных с меткой, в котором меткой являются два младших бита каждого операнда. «Переполнение метки» происходит, если один из операндов имеет ненулевую метку или при 32-битовом арифметическом переполнении. В случае переполнения метки TADDcc и TSUBcc устанавливают бит переполнения PSR, при этом TADDccTV и TSUBccTV вызывают системное прерывание tag_overflow (но не изменяют PSR.icc.V).

6.2.3 Инструкции передачи управления (CTI)

При выполнении инструкции передачи управления изменяется следующее значение счетчика команд (nPC). Существует пять основных типов инструкций передачи управления (CTI):

- условный переход (Bicc, FBfcc, CBccc);
- вызов подпрограммы и создание связи (CALL);
- переход и создание связи (JMPL);
- возврат из системного прерывания (RETT);
- системное прерывание (Ticc).

Категории CTI

Инструкции передачи управления можно разделить на категории (таблица 6.2.2) в соответствии с тем, как рассчитывается адрес назначения (исходя из PC или косвенно из регистров) и моментом времени передачи управления относительно CTI (без задержки, с задержкой, по условию и с задержкой).

Таблица 6.2.2 – Категории CTI

CTI	Расчет адреса назначения	Время передачи управления относительно CTI
Bicc, FBfcc, CBccc	относительно PC	по условию и с задержкой
CALL	относительно PC	с задержкой
JMPL, RETT	косвенно из регистров	с задержкой
Ticc	косвенно из регистров, векторно	без задержки

- CTI относительно PC. Адрес назначения рассчитывается из дополнения её непосредственного поля до 32-разрядной константы со знаком, сдвига влево на 2 разряда этого пословного смещения для создания побайтового смещения, и прибавления получившегося байтового смещения к содержимому PC.

- CTI косвенно из регистров. За адрес назначения принимается «r[rs1] + r[rs2]», если i = 0 или «r[rs1] + sign_ext(simm13)», если i = 1.

- CTI косвенно из регистров, векторно. Адрес назначения рассчитывается поэтапно. Сначала рассчитывается тип системного прерывания: «127 + r[rs1] + r[rs2]», если i = 0 или «127 + r[rs1] + sign_ext(simm13)», если i = 1. Данный тип системного прерывания сохраняется в поле tt регистра TBR. Итоговое содержимое TBR составляет адрес назначения CTI.

- CTI с задержкой (DCTI). Передача управления инструкцией по адресу назначения осуществляется после задержки в 1 инструкцию. Инструкция задержки, выполняется после инструкции CTI, но до того, как будет выполнен переход по адресу назначения CTI.

- CTI без задержки. Управление передается по адресу назначения непосредственно после выполнения CTI.

- CTI по условию и с задержкой (DCTI по условию). Данный вид CTI может вызывать передачу управления, как с задержкой, так и без нее, в зависимости от значения аннулирующего бита «a» инструкции, как для условной передачи, так и для безусловной.

Следует отметить, что как инструкции передачи управления с задержкой, так и по условию и с задержкой классифицируются, как CTI с задержкой или DCTI.

Инструкция задержки

Инструкция, на которую ссылается nPC, когда встретилась инструкция передачи управления с задержкой, называется инструкцией задержки. Обычно это следующая последовательная инструкция в пространстве инструкций (т.е. расположенная по адресу PC + 4). Однако если инструкция, непосредственно предшествующая DCTI, сама является DCTI, то адресом инструкции задержки фактически является адрес назначения предшествующей DCTI (поскольку на него будет указывать nPC при выполнении рассматриваемого DCTI). Более подробное объяснение приведено в подпункте «Пары передач управления с задержкой (DCTI)».

Передача управления с задержкой

В таблице 6.2.3 показан общий порядок выполнения передачи управления с задержкой. Последовательность адресов для выполнения: 8, 12, 16 и 40. Если инструкция передачи управления не выполняется, то последовательность адресов: 8, 12, 16 и 20.

Таблица 6.2.3 – Передача управления с задержкой

PC перед выполнением	nPC перед выполнением	Инструкция
8	12	не CTI
12	16	CTI к 40 с задержкой
16	40	не CTI
…	…	
40	44	…

Передача управления по условию и с задержкой

Передача управления инструкциями передачи управления по условию и с задержкой зависит от значения аннулирующего бита (a) инструкции и истинности заданного условия. Следует отметить, что бит a доступен только в инструкциях ветвления (Bicc, FBfcc и CBccc).

Если в инструкции передачи управления по условию и с задержкой аннулирующий бит сброшен, то инструкция задержки всегда выполняется. Если же в таких инструкциях аннулирующий бит установлен, то инструкция задержки не выполняется, за исключением случаев, когда инструкция передачи управления по условию выполняет переход по условию. В таблице 6.2.4 приводятся условия выполнения или невыполнения инструкции задержи.

Результат выполнения аннулированной инструкции задержки аналогичен результату выполнения инструкции NOP.

Таблица 6.2.4 – Условия выполнения инструкции задержки

Бит a	Тип ветвления	Выполнение инструкции задержки
a = 0	по условию, переход выполнен	Да
	по условию, переход не выполнен	Да
	безусловный, переход выполнен (BA и т. д.)	Да
	безусловный, переход не выполнен (BN и т. д.)	Да
a = 1	по условию, переход выполнен	Да
	по условию, переход не выполнен	Нет (аннулирована)
	безусловный, переход выполнен (BA и т. д.)	Нет (аннулирована)
	безусловный, переход не выполнен (BN и т. д.)	Нет (аннулирована)

- По условию при a = 1.

Если в инструкции ветвления по условию a = 1 (исключая «всегда переходить») и переход не выполнен, инструкция задержки аннулируется (не выполняется), что отображено в таблице 6.2.5. Если переход выполнен, инструкция задержки выполняется.

Таблица 6.2.5 – Невыполненный переход при a = 1

PC	nPC	Инструкция	Действие
8	12	не CTI	выполнено
12	16	ветвление к 40 по условию (a = 1)	переход не выполнен
16	20	не CTI	не выполнено (аннулировано)
20	24	…	выполнено

- По условию при a = 0.

Если в инструкции ветвления по условию a = 0 (включая «всегда переходить»), инструкция задержки выполняется независимо от того, выполнен переход или нет (см. таблицу 6.2.6).

Таблица 6.2.6 – Невыполненный переход при a = 0

PC	nPC	Инструкция	Действие
8	12	не CTI	выполнено
12	16	ветвление к 40 по условию (a = 0)	переход не выполнен
16	20	не CTI	выполнено
20	24	…	выполнено

- Безусловный переход с задержкой.

Если a = 0 в инструкции безусловного перехода («всегда переходить» или «никогда не переходить», т. е. BA, FBA, CBA, BN, FBN и CBN), инструкция задержки всегда выполняется. Если a = 1 в инструкции безусловного перехода, инструкция с задержкой не выполняется.

Замечания по программированию

«Всегда переходить» с a = 1 может быть использована супервизором во время выполнения программы для динамической замены нереализованной команды операцией ветвления в связанную эмулирующую процедуру, которую он записывает в адресное пространство пользователя. При повторном выполнении накладные расходы на эмуляцию нереализованной команды значительно сокращаются. Первые несколько инструкций эмулирующей процедуры можно специально приспособить под предоставление операндов замененной команды во время выполнения.

Замечания по программированию

Аннулирующий бит повышает вероятность того, что компилятор сможет найти полезную инструкцию для заполнения слота задержки после операции ветвления, тем самым сокращая количество выполняемых программой инструкций. Ниже приведены два примера.

Аннулирующий бит может использоваться для перемещения инструкции из тела цикла в слот задержки операции ветвления, которая находится в конце цикла. Если в Bicc (см. таблицу 6.2.7) бит a = 0, то компилятор может переместить инструкцию, которая не является инструкцией передачи управления, из тела цикла в позицию D. Если в Bicc a = 1, то компилятор может скопировать инструкцию, которая не является инструкцией передачи управления, с позиции L в позицию D и изменить точку назначения операции ветвления Bicc на L'.

Таблица 6.2.7 – Пример кода цикла

Адрес	Инструкция
L: L': D:	не CTI не CTI Bicc к L не CTI

Аннулирующий бит может использоваться для перемещения инструкции из ветвей «else» или «then» программного блока «if-then-else» в слот задержки операции ветвления, которая выбирает в которую из них зайти. Поскольку инструкции ветвления по условию поддерживают проверку, как истинности, так и ложности для всех доступных условий, компилятор может компоновать код (возможно с инвертированием значения условий проверки ветвления) таким образом, чтобы инструкция, которая не является инструкцией передачи управления, из ветви «else» или «then» была перемещена в слот задержки инструкции ветвления «if». Такая оптимизация показана в таблице 6.2.8.

Таблица 6.2.8 – Оптимизация «if-then-else»

Адрес	Инструкция	Адрес	Инструкция
 DELAY: ELSE: THEN:	Bicc(по условию, a = 1) THEN инструкция 1 ветви then инструкция 1 ветви else goto ... инструкция 2 ветви then	 DELAY: THEN: ELSE:	Bicc(по условию, a = 1) ELSE инструкция 1 ветви else инструкция 1 ветви then goto ... инструкция 2 ветви else

Инструкции CALL и JMPL

С помощью инструкции CALL в r[15] (выходной регистр 7 – %o7) записывается содержимое PC (который указывает на саму инструкцию CALL). Инструкция CALL вызывает передачу управления с задержкой в произвольный адрес назначения, вычисляемый относительно PC.

С помощью инструкции JMPL в регистр r, определяемый полем rd, записывается содержимое PC (который указывает на саму инструкцию JMPL). Инструкция JMPL вызывает передачу управления с задержкой в произвольный адрес назначения.

Инструкция SAVE

Инструкция SAVE аналогична инструкции ADD, за исключением того, что первая ещё и уменьшает CWP на 1. При этом окно (CWP – 1) становится новым текущим окном, таким образом «сохраняя» окно вызывающей подпрограммы. Притом, исходные значения

регистров для операции суммирования берутся из окна CWP, в то время как результат записывается в регистр окна (CWP – 1).

Инструкция RESTORE

Инструкция RESTORE также аналогична инструкции ADD, за исключением того, что первая увеличивает CWP на 1. При этом окно (CWP + 1) становится новым текущим окном, таким образом, окно вызывающей программы «восстанавливается». Притом, исходные значения регистров для операции суммирования берутся из окна CWP, в то время как результат записывается в регистр окна (CWP + 1).

Инструкции SAVE и RESTORE сравнивают новое значение CWP с маской недопустимости окна (WIM) на предмет переполнения или обратного переполнения окна.

Замечания по программированию

Вызов процедуры осуществляется через выполнение инструкции CALL (или JMPL). Если процедуре требуется регистровое окно, то она выполняет инструкцию SAVE. Подпрограмма, которая не выделяет регистрового окна под собственные нужды (вероятно «листовая» процедура, т.е. не имеющая в своем составе вложенных вызовов), не должна перезаписывать никакие регистры окна, за исключением выходных регистров 0...6 (следует учитывать, что в %o6 хранится текущее значение стека).

Возврат из процедуры, использующей регистровое окно, выполняется с помощью инструкций RESTORE и JMPL. Возврат из процедуры, не использующей выделения нового регистрового окна, выполняется с помощью одной только инструкции JMPL. Обычно инструкция JMPL осуществляет возврат к инструкции, следующей за инструкцией задержки инструкций CALL или JMPL; другими словами, стандартный адрес возврата: 8 плюс адрес, сохраненный с помощью CALL или JMPL.

Инструкции SAVE и RESTORE могут использоваться для атомарности установки нового указателя стека памяти в регистр r и обновления CWP.

Инструкции системных прерываний (Ticc)

С помощью инструкции Ticc проводится оценка кодов условий, обозначенных в поле cond и, в случае истинности результата, генерация системного прерывания. Другими словами, они изменяют поле tt базового регистра системных прерываний (TBR) и вызывают передачу управления без задержки по адресу TBR. Если выбранные коды условий оцениваются как ложные, результат выполнения инструкции аналогичен NOP.

Инструкция Ticc может задавать один из 128 типов программного системного прерывания, который может быть помещен в поле tt регистра TBR. После того как Ticc сработала, процессор запрещает системные прерывания, входит в режим супервизора, декрементирует CWP, сохраняет PC и nPC в r[17] и r[18] (локальные регистры 1 и 2) нового окна, помещает значение 128 плюс исходный операнд в поле tt регистра TBR и предает управление по адресу, указанному в TBR. Для получения дополнительной информации – см. подраздел «5.4 Системные прерывания».

Замечания по программированию

Ticc может использоваться для реализации точек прерывания, отслеживания и передачи управления программному обеспечению супервизора. Ticc может также использоваться для выполнения проверки во время рабочего цикла, например, выхода индекса за пределы массива, целочисленного переполнения и т. д.

Пары передач управления с задержкой (DCTI)

Если инструкция задержки сама является инструкцией передачи управления, такую пару инструкций называют парой инструкции передачи управления с задержкой (парой

DCTI). Порядок выполнения пары DCTI представлен в таблице 6.2.10, он базируется на последовательности кода, приведенного в таблице 6.2.9.

Таблица 6.2.9 – Пример кода пары DCTI

адрес	инструкция	адресат
8:	не CTI	
12:	CTI	40
16:	CTI	60
20:	не CTI	
24:	…	
40:	не CTI	
44:	…	
60:	не CTI	
64:	…	

В первых пяти случаях, рассмотренных в таблице 6.2.10, первая инструкция пары DCTI – CTI, которая вызывает передачу управления. Данные блоки характерны для элементов связи DCTI, которые возникают при возврате кода пользователя после обработки сортировщиком системных прерываний с помощью элементов связи инструкции «JMPL, RETT». «JMPL, RETT» и «RETT, CTI пользователя» возникают при возврате после обработки сортировщиком системных прерываний.

В шестом случае из таблицы 6.2.10 первая инструкция пары DCTI (возможно, невыполняемая) – ветвление по условию, при этом адрес назначения пары DCTI принадлежит адресному пространству, в котором располагается пара DCTI, но в остальном он является неопределенным.

Таблица 6.2.10 – Порядок выполнения элементов связи DCTI по адресу

Случай	12: CTI к 40	16: CTI к 60	Порядок выполнения по адресу
1	безусловная DCTI	выполняемая DCTI	12, 16, 40, 60, 64, ...
2	безусловная DCTI	невыполняемая B*cc (a = 0)	12, 16, 40, 44, ...
3	безусловная DCTI	невыполняемая B*cc (a = 1)	12, 16, 44, 48, ... (40 аннулирован)
4	безусловная DCTI	B*A (a = 1)	12, 16, 60, 64, ... (40 аннулирован)
5	B*A (a = 1)	любая CTI	12, 40, 44, ... (16 аннулирован)
6	B*cc	DCTI	12, не определяется
Примечание – Если бит «a» не обозначен, он может быть сброшен или установлен. Аббревиатуры, используемые в данной таблице, сведены в таблицу 6.2.11.			

Таблица 6.2.11 – Аббревиатуры, используемые в предыдущей таблице

Аббревиатура	Наименование инструкций
B*A	BA, FBA или CBA
B*cc	Bicc, FBfcc или CBccc (включая BN, FBN, CBN, но исключая B*A)
безусловная DCTI	CALL, JMPL, RETT или B*A (при a = 0)
выполняемая DCTI	CALL, JMPL, RETT, B*A (при a = 0) или выполняемый B*cc

6.2.4 Чтение/запись регистров состояния

Инструкции чтения / записи регистра состояния осуществляют доступ к видимым для программы регистрам состояния и статуса. С помощью данных инструкций выполняется считывание / запись регистров состояния в / из регистры r. Инструкция чтения / записи вспомогательного регистра состояния является привилегированной, если регистр, к которому выполняется доступ, является привилегированным.

6.2.5 Инструкции операций с плавающей запятой (FPop)

Инструкции операций с плавающей запятой (FPop) обычно используют триаду адресов регистров. Они вычисляют результат, являющийся функцией двух исходных операндов, и записывают его в регистр назначения f. Исключение составляют операции преобразования с плавающей запятой (использующие один исходный операнд и один операнд назначения) и операции сравнения с плавающей запятой (которые не выполняют записи в регистр f, но обновляют поле fcc регистра FSR). Если блок с плавающей запятой не присутствует или если PSR.EF = 0, то инструкция FPop генерирует системное прерывание fp_disabled.

Термин «FPop» относится к инструкциям с кодами операций FPop1 и FPop2, он не включает в себя инструкций ветвления, основанные на кодах условий операций с плавающей запятой (FBfcc) или инструкций загрузки / сохранения с плавающей запятой.

Все инструкций FPop сбрасывают поле ftt и устанавливают поле cexc, если их выполнение не было прервано системным прерыванием. Некоторые инструкций FPop также осуществляют запись в поле fcc. Все инструкций FPop, которые могут генерировать исключения IEEE, устанавливают поля cexc и aexc, если их выполнение не было прервано системным прерыванием. Инструкций FABS, FMOV и FNEG не могут генерировать исключения IEEE, поэтому они сбрасывают поле cexc, а поле aexc оставляют в неизменном виде.

6.2.6 Инструкции сопроцессора (CPop)

Инструкции сопроцессора выполняются присоединенным сопроцессором. Если сопроцессора нет или PSR.EC = 0, то CPop вызывают системное прерывание cp_disabled.

Поля инструкций CPop, за исключением op и op3 интерпретируются исключительно самим сопроцессором.

Термин «CPop» относится к инструкциям с кодами операций CPop1 и CPop2, он не включает в себя инструкций ветвления, основанные на кодах условий сопроцессора (CBccc), или инструкций загрузки / сохранения сопроцессора.

7 Программно-аппаратный комплекс средств разработки и отладки

7.1 Интегрированная среда разработки и отладки приложений

Интегрированная среда разработки (далее ИСР) предназначена для поддержки процесса разработки программного обеспечения (ПО) для 32-разрядного процессора архитектуры SPARC V8 (далее LEON4) и предоставляет пользователю следующую функциональность:

- управление проектами и конфигурациями разрабатываемого ПО;
- синтаксически ориентированное редактирование исходных текстов на языках C++, C и ассемблер;
- автоматическая проверка синтаксической корректности исходных текстов в процессе редактирования;
- отображение подсказок, автоматическое дополнение и рефакторинг на основе структурного анализа исходного текста;
- поддержка процесса сборки проекта с использованием утилит командной строки (компилятор C++, ассемблер, линкер и др.);
- структурированное отображение диагностических сообщений по результатам сборки проекта (ошибок, предупреждений) с привязкой к позициям в исходном тексте;
- обеспечение процесса интерактивной (высокоуровневой и низкоуровневой) отладки ПО;
- поддержка отладки с использованием аппаратного отладчика;
- отображение и редактирование данных, содержащихся в памяти и регистрах общего и специального назначения.

ИСР выполнена на основе Eclipse CDT и плагина LEON C/C++ IDE for Eclipse и предоставляет пользователю графический интерфейс к следующим компонентам:

- компилятору C/C++;
- ассемблеру;
- линкеру и утилитам;
- аппаратному отладчику.

7.1.1 Состав ИСР

В состав ИСР входят:

- менеджер проектов;
- редактор исходных текстов;
- подсистема структурного анализа исходного текста;
- подсистема сборки проекта;
- подсистема отладки.

7.1.2 Интерфейс пользователя

По умолчанию в ИСР поддерживаются две перспективы – «разработка проекта» и «отладка». Во втором случае пользователь имеет возможность создавать дополнительные перспективы, а также модифицировать набор, размер и расположение отображаемых интерфейсных элементов в рамках любой из перспектив.

Первоначальное расположение окон пользовательского интерфейса в рамках перспективы «разработка проекта» представлено на рисунке 7.1.1.

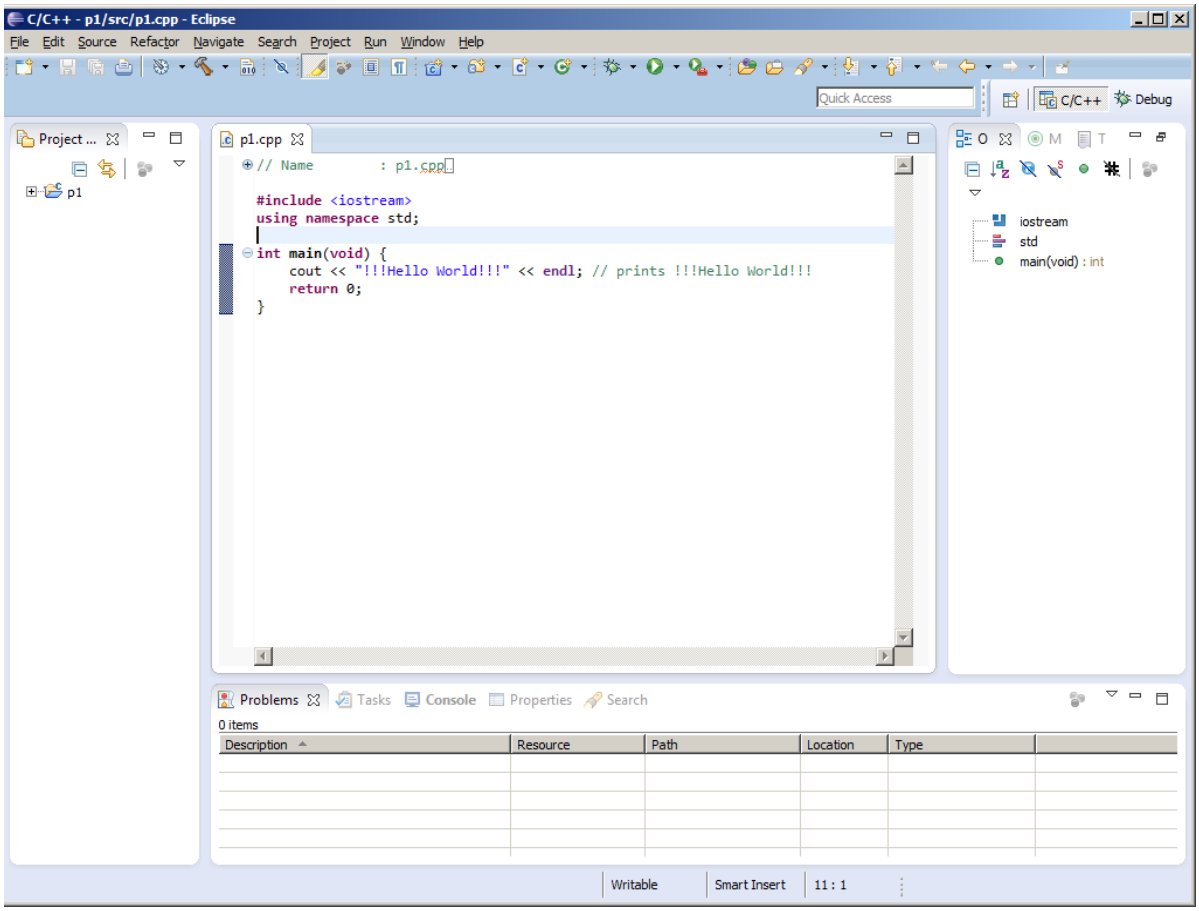


Рисунок 7.1.1 – Пользовательский интерфейс (перспектива «разработка проекта»)

Содержимое окон пользовательского интерфейса:

1 Область отображения проектов содержит список проектов рабочей области. Каждый проект представлен деревом, содержащим подключённые к проекту каталоги и файлы.

2 Область редактирования содержит окна с содержимым редактируемых файлов. Переключение между окнами осуществляется с помощью интерфейсного механизма закладок.

3 Область отображения структурных элементов содержит список структурных элементов исходного текста программы, содержащегося в текущем редактируемом файле.

4 Область отображения дополнительной информации содержит следующие окна:

- окно отображения списка синтаксических и семантических ошибок, обнаруженных подсистемой структурного анализа исходного текста;
- окно отображения результата мультифайлового поиска;
- окно отображения диагностических сообщений компилятора, ассемблера и линкера.

Переключение между окнами осуществляется с помощью интерфейсного механизма закладок.

Первоначальное расположение окон пользовательского интерфейса в рамках перспективы «отладка» представлено на рисунке 7.1.2.

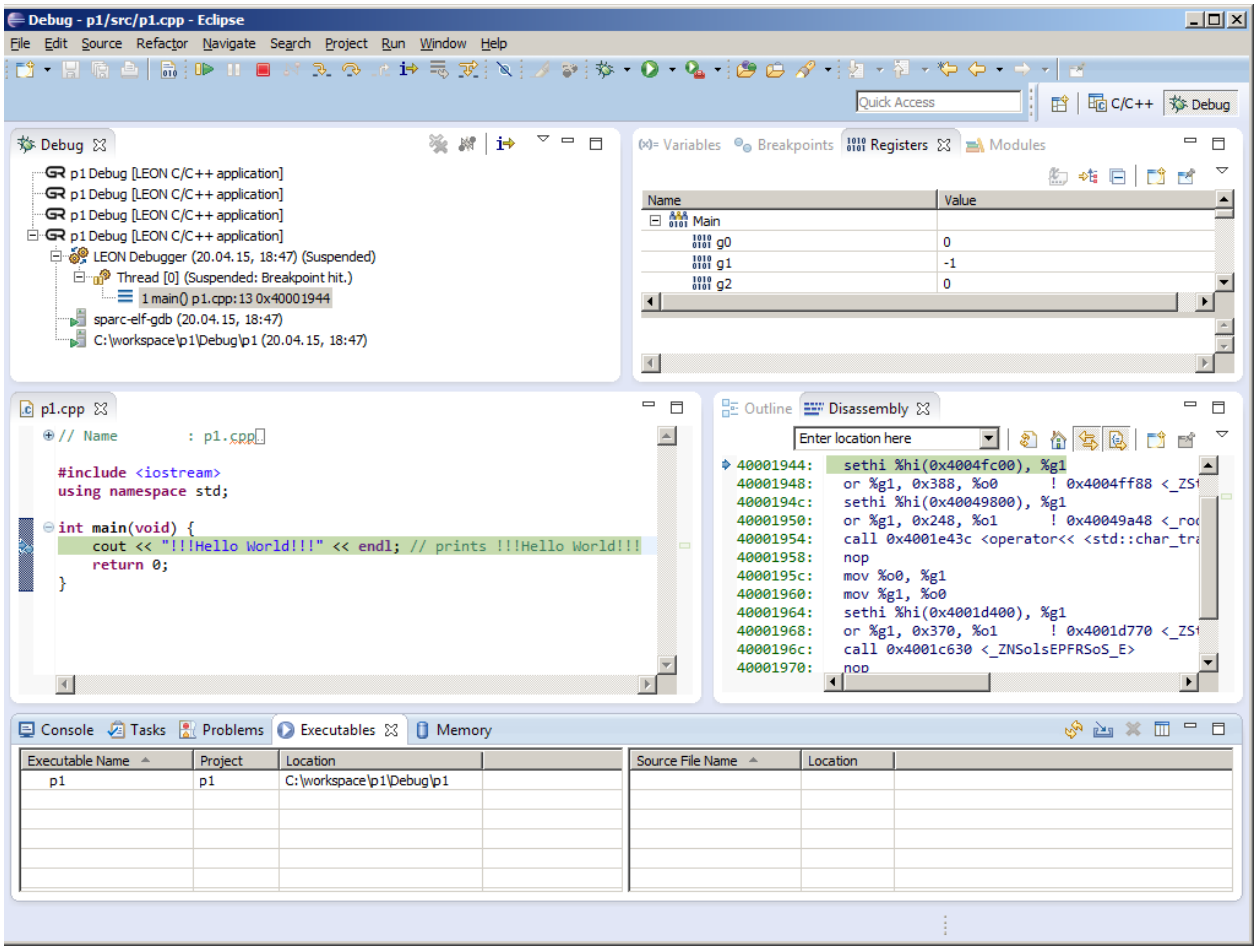


Рисунок 7.1.2 – Пользовательский интерфейс (перспектива «отладка»)

Содержимое окон пользовательского интерфейса:

1 Область подсистемы отладки содержит структурированную в виде дерева информацию о состоянии подсистемы отладки.

2 Область отображения переменных и точек останова содержит следующие окна:

- окно отображения и редактирования указанных пользователем переменных;
- окно отображения и редактирования регистров общего назначения;
- окно отображения и редактирования регистров специального назначения;
- окно с описанием заданных пользователем точек останова.

3 Область отображения исходного текста содержит исходный текст текущего отлаживаемого модуля программы.

4 Область отображения дизассемблера содержит дизассемблированный исполняемый код отлаживаемой программы.

5 Область отображения дополнительной информации содержит следующие окна:

- окно консоли;
- окна отображения и редактирования памяти;
- окно с динамическим списком исполняемых файлов и связанных с ними файлов исходного текста.

Переключение между окнами осуществляется с помощью интерфейсного механизма закладок.

Диалоги настроек предоставляют доступ к различным параметрам ИСР, которые могут быть заданы или изменены пользователем. Настройки делятся на следующие категории:

- настройки ИСР в целом;

- настройки, связанные с конкретным проектом;
- настройки, связанные с конкретным исходным файлом проекта.

Для каждой категории настройки делятся на смысловые группы. Диалог настроек разделен на две части. Слева расположено дерево, отображающее иерархию групп настроек данной категории, с возможностью фильтрации по названию группы. Справа – панель для доступа к настройкам выбранной группы.

Основные типы настроек ИСР:

1 Общие настройки графического интерфейса, связанные с отображением его элементов, а также горячие клавиши.

2 Управление перспективами.

3 Настройки менеджера проектов (порядок сборки проектов, действия при запуске/завершении).

4 Общие настройки текстового редактора.

5 Языково-ориентированные настройки текстового редактора:

- стиль отображения синтаксиса;
- настройки и шаблоны автоматического форматирования и автодополнения;
- настройки, задающие параметры стиля программирования;
- настройки, связанные со структурным анализом исходного текста.

6 Общие настройки подсистемы сборки (переменные окружения, журналирование и др.).

7 Общие настройки подсистемы отладки.

Часть типов настроек проекта является полным аналогом общих настроек ИСР (в части настроек текстового редактора, подсистем сборки и отладки). Значения этих настроек действуют при работе с указанным проектом, либо с файлами, входящими в проект (например, настройки редактора). По умолчанию, значения этих настроек наследуются из общих настроек ИСР, однако, они могут быть и заданы явно.

Кроме того, настройки проекта содержат дополнительную специфическую часть, а именно:

- параметры сборки (группируются в рамках конфигураций). Используются подсистемой сборки проекта для задания опций утилит командной строки;
- зависимости между проектом и другими проектами;
- управление сессиями отладки.

Настройки конкретного файла служат для перекрытия части настроек проекта, в основном, это касается параметров сборки. Данный механизм позволяет, в частности, производить компиляцию отдельных исходных файлов с значениями опций, отличных от значений для всех остальных файлов проекта, либо полностью исключить конкретный файл из процесса сборки в рамках заданной конфигурации.

7.1.3 Функциональные блоки

Менеджер проектов

Менеджер проектов отвечает за работу с проектами. Он обеспечивает загрузку проектов в ИСР, управление составом проектов, а также набором конфигураций проекта – параметрами, связанными как с проектом в целом, так и с отдельными файлами, входящими в проект.

Менеджер проектов поддерживает следующую функциональность:

а) работа с файлами проекта (содержащими информацию о проекте) – загрузка в ИСР и сохранение изменений;

б) поддержка работы со списком существующих проектов в области отображения проектов – загрузка/выгрузка, создание, удаление и переименование проектов, управление зависимостями между проектами;

в) поддержка работы с составом проекта – подключение, создание новых, переименование и удаление исходных файлов и каталогов;

г) анализ и отображение зависимостей исходных файлов и включаемых (include) файлов;

д) управление редактированием исходных файлов, входящих в проект;

е) поддержка доступа к настройкам проекта и отдельных исходных файлов в рамках конфигураций, управление списком конфигураций;

ж) управление подсистемой сборки проекта (передача необходимых параметров).

Редактор исходных текстов

Данная компонента предоставляет возможность пользователю создавать и модифицировать контент текстовых файлов как входящих в проект, так и загружаемых отдельно. Хотя редактор может быть успешно использован для работы с текстовыми файлами общего вида (например, исходниками документации, файлами описания релиза и прочее), основное его назначение – редактирование исходных (в том числе включаемых) файлов на языках C/C++ и ассемблер.

Исходя из этого, редактор содержит специфическую функциональность (в дополнение к стандартной функциональности текстовых редакторов офисного уровня), поддерживающую синтаксически-ориентированное редактирование и позволяющую в несколько раз ускорить написание программ, а именно:

1 Автоматическое выделение цветом:

- ключевых слов языков C++, C и ассемблера;
- имен регистров общего и специального назначения;
- разрешенных подсистемой структурного анализа исходного текста имен (переменных, методов, классов и пр.).

2 Предварительный синтаксический анализ:

- нотификация пользователя об ошибочных (синтаксически некорректных) элементах редактируемого текста;
- нотификация пользователя об ошибочных (не разрешенных подсистемой структурного анализа исходного текста) именах.

3 Поддержка работы со структурой текста:

- автоматическая расстановка отступов и переносов строк;
- свертка/развертка описания структурных элементов (классов, методов, функций, типов и пр.);
- свертка/развертка многострочных комментариев.

4 Поддержка работы с семантически значимыми элементами входного языка:

- автоматическое дописывание текста (например, добавление скобок после оператора «if»);
- автоматическое дополнение (подсказка с возможностью выбора) вложенных структурных элементов (например, предъявление списка полей при вводе имени переменной структурного типа);
- автоматическое переименование переменных, функций, методов, классов и пр. во всех файлах проекта (рефакторинг);
- автоматическое отображение строк объявления имени при наведении курсора на разрешенное имя;
- переход (например, по нажатию «Ctrl + правая кнопка мыши») в строку объявления разрешенного имени в исходном тексте.

7.1.4 Подсистема структурного анализа исходных текстов

Подсистема структурного анализа исходного текста занимается составлением и поддержкой в актуальном состоянии списка всех объявленных в проекте идентификаторов. Для этого подсистема структурного анализа исходного текста производит парсинг и интерпретацию всех исходных и включаемых (include) файлов проекта. Интерпретация производится как при подключении нового файла к проекту, так и при сохранении отредактированного файла. Далее эта информация используется редактором исходных текстов и менеджером проектов.

В редакторе эта информация используется для поддержки функций синтаксически-ориентированного редактирования.

В менеджере проектов эта информация используется для индикации состояния файла (наличие синтаксических ошибок) в дереве проекта.

7.1.5 Подсистема отладки

Подсистема отладки обеспечивает загрузку, запуск на исполнение и отладку исполняемого кода на реальном микроконтроллере LEON4, с использованием аппаратного отладчика, работающего по JTAG-интерфейсу.

На уровне интерфейса пользователя, подсистема отладки предоставляет доступ к:

- исходному и объектному коду программы, с указанием текущей позиции счетчика команд;
- текущим значениям переменных программы пользователя;
- стеку вызовов функций;
- различным ресурсам микроконтроллера (регистры, память и т.п.);
- списку пользовательских точек останова.

С точки зрения взаимодействия с целевой архитектурой, подсистема отладки обеспечивает:

- загрузку исполняемого кода в память целевого процессора;
- запуск на исполнение программы на целевом процессоре;
- останов (паузу) исполнения по команде пользователя;
- установку точек останова по коду;
- установку точек останова по данным (события чтения/записи);
- чтение / запись данных из памяти и регистров.

7.2 Программное обеспечение «GRAIP»

Программное обеспечение «GRAIP» РОФ.КФДЛ.00398 представляет собой консольное приложение, позволяющее выполнять программирование и отладку программ, разработанных для ИС 1906ВМ016 и ИС 1906ВМ01А6.

7.3 Аппаратная часть

На рисунке показана плата отладочного устройства КФДЛ.424939.013 для ИС 1906ВМ016 и ИС 1906ВМ01А6.

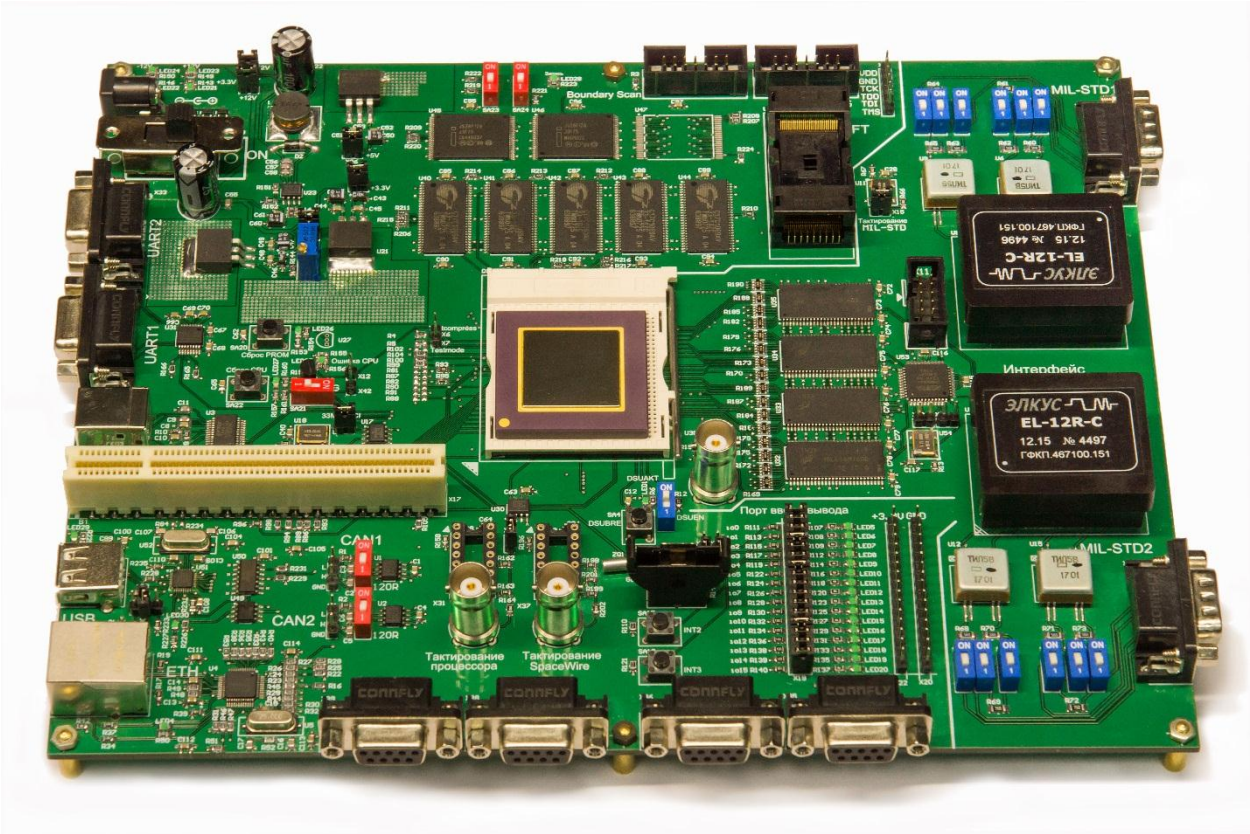


Рисунок 7.3.1 – Внешний вид платы отладочного устройства КФДЛ.424939.013

Для обеспечения взаимодействия ПО ИСР и отладочной платы служит адаптер JEM – LEON, подключаемый к USB–разъему персонального компьютера и JTAG–разъему на базовой отладочной плате (см. рисунок 7.3.2).

Рисунок 7.3.2 – Адаптер JEM – LEON

Адаптер выполнен в пластиковом корпусе и имеет разъем USB для подключения к персональному компьютеру с одной стороны и разъем для подключения отлаживаемого устройства с другой стороны. Электропитание адаптера осуществляется от порта USB компьютера.

Контакты отладочного разъема (рисунок 7.3.3) должны подключаться к плате пользователя в соответствии с таблицей 7.3.1.

Таблица 7.3.1– Контакты отладочного разъема

Название контакта	Номер контакта		Название контакта
VCC	1	2	VCC
NC	3	4	TRST
TDI	5	6	GND
TMS	7	8	GND
TCK	9	10	GND
TDO	11	12	GND
RSTN	13	14	GND

Описание назначения контактов, приведенных в таблице 7.3.1:

- VCC – напряжение питания +3,3 В на плате пользователя, соединяется с выводами VCC2 микропроцессора;
- NC – не подключается;
- TDI – соединяется с выводом TDI микропроцессора;
- TMS – соединяется с выводом TMS микропроцессора;
- TCK – соединяется с выводом TCK микропроцессора;
- TDO – соединяется с выводом TDO микропроцессора;
- RSTN – сброс, соединяется с выводом RESETN# микропроцессора;
- TRST – сброс JTAG порта (выход отладчика, открытый сток с подтяжкой к VCC), может оставаться неподключенным;
- GND – общий, соединяется с выводами GND микропроцессора.

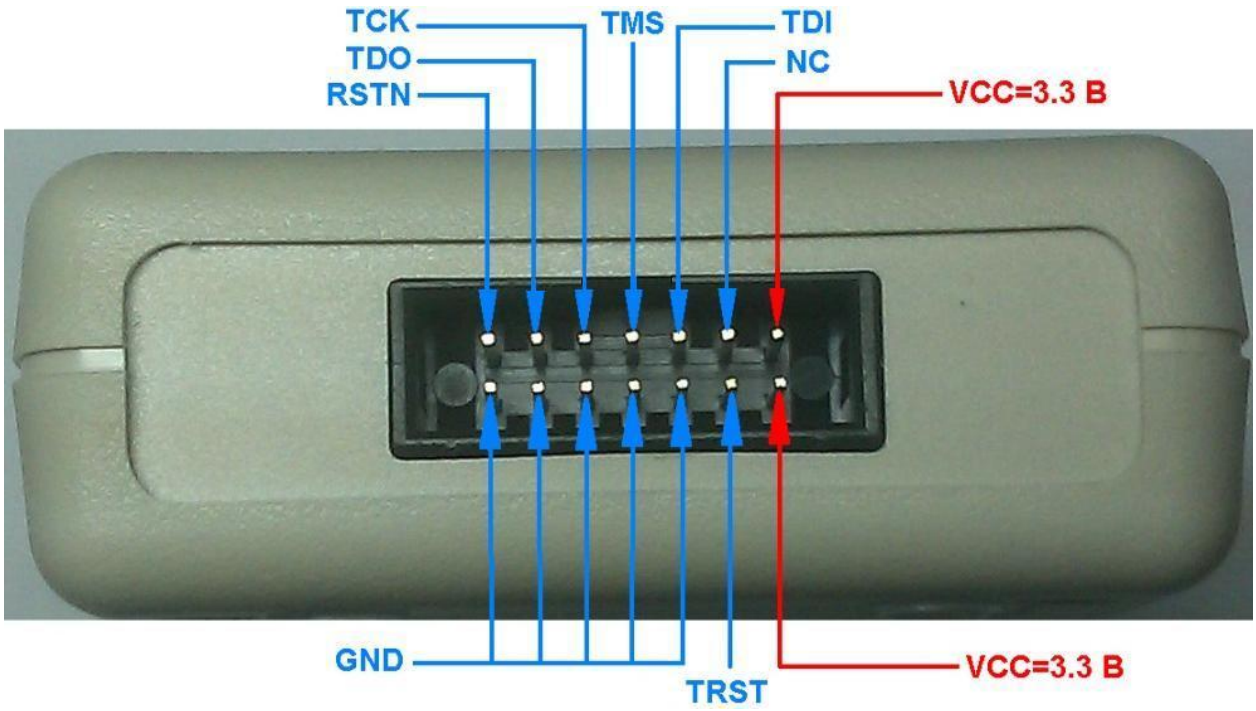


Рисунок 7.3.3– Разъем аппаратного блока адаптера

8 Заключение

В настоящем руководстве пользователя приведено подробное описание архитектуры, функционального построения, системы команд и особенностей применения ИС 1906ВМ016 и 1906ВМ01А6, которые представляют собой 32-разрядный процессор на базе ядра LEON4 архитектуры SPARC V8, с частичной поддержкой расширения V8e, с контроллерами интерфейсов PCI 2.2, Ethernet 10/100, SpaceWire, CAN 2.0B, USB 2.0, MIL-STD-1553, JTAG, DSU, UART, таймерами/счетчиками общего назначения, сторожевым таймером, портом ввода/вывода общего назначения.

Микропроцессоры предназначены для применения в перспективных системах управления аппаратами авиационных и космических направлений.

Все значения электрических параметров микросхем приведены в технических условиях АЕНВ.431280.039ТУ. Значения параметров, приведенные в настоящем РП, являются справочными.

Руководство пользователя может служить практическим руководством по применению микропроцессора для разработчиков систем на основе ИС 1906ВМ016, 1906ВМ01А6 и программистов.

Приложение А

(обязательное)

Описание инструкций

В настоящем приложении приводятся обозначения, используемые в описаниях синтаксических структур языка ассемблера, а также список некоторых искусственных инструкций, предусмотренных ассемблером архитектуры SPARC для удобства программистов, работающих с языком ассемблера.

Используемые обозначения

Описание синтаксических структур в данном приложении построено на использовании различных типов шрифтов. Элементы, выделенные полужирным шрифтом, – литеральные константы, которые должны записываться в том виде, в котором приведены. Элементы, выделенные курсивом, – метасимволы, которые должны заменяться числовыми или символьными значениями при записи фактического кода на языке ассемблера SPARC. Например, «*asi*» должно быть заменено числом в диапазоне от 0 до 255 (значение битов asi в двоичной инструкции) или символом, соответствующим такому числу.

Нижний индекс символов языка обозначает размещение компонента операции в сгенерированной двоичной инструкции. Например, reg_{rs2} – это reg (имя регистра), чье двоичное значение размещается в поле rs2 результирующей инструкции.

Имена регистров

reg – имя целочисленного регистра. Оно может принимать одно из следующих значений:

%r0 ... **%r31**
%g0 ... **%g7** (глобальные регистры; эквивалент **%r0** ... **%r7**)
%o0 ... **%o7** (выходные регистры; эквивалент **%r8** ... **%r15**)
%l0 ... **%l7** (локальные регистры; эквивалент **%r16**...**%r23**)
%i0 ... **%i7** (входные регистры; эквивалент **%r24** ...**%r31**)
%fp (указатель фрейма; условное обозначение **%i6**)
%sp (указатель стека; условное обозначение **%o6**)

Фактически, предпочтение отдается формам **%sp**, **%fp**, **%g***n*, **%o***n*, **%l***n* и **%i***n*, а не **%r***n*.

Нижний индекс обозначает размещение компонента операции в бинарной инструкции:

reg_{rs1} (поле rs1)
reg_{rs2} (поле rs2)
reg_{rd} (поле rd)

freg – имя регистра с плавающей запятой. Оно может принимать одно из следующих значений:

%f0 ... **%f31**

Нижний индекс обозначает размещение компонента операции в бинарной инструкции:

$freg_{rs1}$ (поле rs1)

$freg_{rs2}$ (поле rs2)
$freg_{rd}$ (поле rd)

creg – имя регистра сопроцессора. Оно может иметь одно из следующих значений:
%c0 ... %c31

Нижний индекс обозначает размещение компонента операции в бинарной инструкции:
$creg_{rs1}$ (поле rs1)
$creg_{rs2}$ (поле rs2)
$creg_{rd}$ (поле rd)

asr_reg – имя вспомогательного регистра состояния. Оно может принимать одно из следующих значений:
%asr1 ... %asr31

Нижний индекс обозначает размещение компонента операции в бинарной инструкции:
asr_reg_{rs1} (поле rs1)
asr_reg_{rd} (поле rd)

Имена специальных символов

Специальные символы в таблице синтаксических структур обозначены полужирным шрифтом. Они должны записываться в таком же виде, включая знак вначале (%). Этот знак является частью имени символа и должен записываться вначале.

Имена символов и регистры или операторы, на которых они ссылаются:

%psr	регистр состояния процессора
%wim	регистр маски недопустимости окна
%tbr	базовый регистр системных прерываний
%y	регистр Y
%fsr	регистр состояния операций с плавающей запятой
%csr	регистр состояния сопроцессора
%fq	очередь операций с плавающей запятой
%cq	очередь сопроцессора
%hi	унарный оператор, извлекающий старшие 22 бита своего операнда
%lo	унарный оператор, извлекающий младшие 10 битов своего операнда

Непосредственные значения:

imm7	Непосредственная константа в диапазоне от –64 до 127 (представляет собой 7 бит, со знаком или без знака)
uimm7	Непосредственная константа в диапазоне от 0 до 127 (представляет собой 7 бит, без знака)
simm13	Непосредственная константа в диапазоне от –4 096 до 4 095 (представляет собой 13 бит, со знаком)
const22	Константа, которая может быть представлена с помощью 22-х битов
asi	Идентификатор адресного пространства. Непосредственная константа в диапазоне от 0 до 255 (представляет собой 8 бит, без знака).

Метки

Метка – это последовательность символов из букв алфавита «a – z, A – Z» (верхнего и нижнего регистра), подчеркивания «_», знака «$», точек «.» и десятичных цифр «0 – 9». Метка может содержать десятичные цифры, но не может начинаться с них.

Другие синтаксические структуры компонентов операции

В некоторых инструкциях используются разные синтаксические структуры компонента операции:

address:
- reg_{rs1} (эквивалентно: reg_{rs1} + **%g0**)
- reg_{rs1} + reg_{rs2}
- reg_{rs1} + *simm13*
- reg_{rs1} – *simm13*
- *simm13* (эквивалентно: **%g0** + *simm13*)
- *simm13* + reg_{rs1} (эквивалентно: reg_{rs1} + *simm13*)

regaddr («адрес только из регистра»):
- reg_{rs1} (эквивалентно: reg_{rs1} + **%g0**)
- reg_{rs1} + reg_{rs2}

reg_or_imm («значение из регистра или непосредственное значение»):
- reg_{rs2}
- *simm13*

software_trap#:
- reg_{rs1} (эквивалентно: reg_{rs1} + **%g0**)
- reg_{rs1} + reg_{rs2}
- reg_{rs1} + *imm7*
- reg_{rs1} – *imm7*
- *uimm7* (эквивалентно: **%g0** + *uimm7*)
- *imm7* + reg_{rs1} (эквивалентно: reg_{rs1} + *imm7*)

Итоговое значение операнда (номер программного системного прерывания) должно находиться в диапазоне от 0 до 127 включительно.

Комментарии

Большинство ассемблеров SPARC поддерживают два типа комментариев: комментарии в стиле языка C «/*...*/» (которые могут занимать несколько строк) и комментарии «! ...», которые начинаются с символа «!» и длятся до конца строки.

Структура синтаксиса

Предлагаемая синтаксическая структура языка ассемблера SPARC обеспечивает следующее:

- операнд назначения (если имеется) всегда определяется, как последний (крайний справа) операнд в выражении языка ассемблера;
- ссылка на содержимое ячейки памяти (в инструкциях загрузки, сохранения или SWAP) всегда обозначается квадратными скобками «[ ]». Ссылка на адрес в памяти (такая, как в JMPL, CALL или SETHI) указывается без квадратных скобок.

Искусственные инструкции

В таблице А.1 приводится порядок преобразования набора искусственных инструкций (или «псевдо-инструкций») в фактические инструкции SPARC. Ассемблер SPARC поддерживает данные инструкции для удобства программистов, работающих с языком ассемблера.

Следует отметить, что искусственные инструкции не следует путать с «псевдо-операциями», которые обычно предоставляют информацию компонующей программе, но не генерируют инструкции. Искусственные инструкции всегда генерируют инструкции; они обеспечивают больше мнемонических синтаксических структур для стандартных инструкций SPARC.

Таблица А.1 – Преобразование искусственных инструкций в инструкции SPARC

Искусственная инструкция		Инструкция(-и) SPARC		Комментарий
1		2		3
cmp	*reg_{rs1}*, *reg_or_imm*	**subcc**	*reg_{rs1}*, *reg_or_imm*, **%g0**	сравнение
jmp	*адрес*	**jmpl**	*адрес*, **%g0**	
call	*адрес*	**jmpl**	*адрес*, **%o7**	
tst	*reg_{rs2}*	**orcc**	**%g0**, *reg_{rs2}*, **%g0**	проверка
ret		**jmpl**	**%i7 + 8, %g0**	возврат из подпрограммы
retl		**jmpl**	**%o7 + 8, %g0**	возврат из листовой подпрограммы
restore		**restore**	**%g0, %g0,%g0**	стандартное восстановление
save[1)]		**save**	**%g0, %g0, %g0**	стандартное сохранение
set[2)]	*значение*, *reg_{rd}*	**sethi**	**%hi**(*значение*), *reg_{rd}*	(значение & 0x1fff) равно 0
			или	
		or	**%g0**, *значение*, *reg_{rd}*	(–4096 ≤ значение ≤ 4095)
			или	
		sethi	**%hi**(*значение*), *reg_{rd}*;	(в противном случае)
		or	*reg_{rd}*, **%lo**(*значение*), *reg_{rd}*	
not	*reg_{rs1}*, *reg_{rd}*	**xnor**	*reg_{rs1}*, **%g0**, *reg_{rd}*	поразрядное дополнение до единицы (обратный код)
not	*reg_{rd}*	**xnor**	*reg_{rd}*, **%g0**, *reg_{rd}*	поразрядное дополнение до единицы (обратный код)
neg	*reg_{rs2}*, *reg_{rd}*	**sub**	**%g0**, *reg_{rs2}*, *reg_{rd}*	поразрядное дополнение до двух (дополнительный код)
neg	*reg_{rd}*	**sub**	**%g0**, *reg_{rd}*, *reg_{rd}*	поразрядное дополнение до двух (дополнительный код)
inc	*reg_{rd}*	**add**	*reg_{rd}*, 1, *reg_{rd}*	увеличение на 1
inc	*const13*, *reg_{rd}*	**add**	*reg_{rd}*, *const13*, *reg_{rd}*	увеличение на const13
inccc	*reg_{rd}*	**addcc**	*reg_{rd}*, 1, *reg_{rd}*	увеличение на 1 и установка icc
inccc	*const13*, *reg_{rd}*	**addcc**	*reg_{rd}*, *const13*, *reg_{rd}*	увеличение на const13 и установка icc
dec	*reg_{rd}*	**sub**	*reg_{rd}*, 1, *reg_{rd}*	уменьшение на 1
dec	*const13*, *reg_{rd}*	**sub**	*reg_{rd}*, *const13*, *reg_{rd}*	уменьшение на const13
deccc	*reg_{rd}*	**subcc**	*reg_{rd}*, *1*, *reg_{rd}*	уменьшение на 1 и установка icc
deccc	*const13*, *reg_{rd}*	**subcc**	*reg_{rd}*, *const13*, *reg_{rd}*	уменьшение на const13 и установка icc

Окончание таблицы А.1

1	2	3
btst *reg_or_imm*, reg_{rs1}	**andcc** reg_{rs1}, *reg_or_imm*, **%g0**	проверка бита
bset *reg_or_imm*, reg_{rd}	**or** reg_{rd}, *reg_or_imm*, reg_{rd}	установка бита
bclr *reg_or_imm*, reg_{rd}	**andn** reg_{rd}, *reg_or_imm*, reg_{rd}	сброс бита
btog *reg_or_imm*, reg_{rd}	**xor** reg_{rd}, *reg_or_imm*, reg_{rd}	переключение бита
clr reg_{rd}	**or** **%g0**, **%g0**, reg_{rd}	сбросить (обнулить) регистр
clrb [*адрес*]	**stb** **%g0**, [*адрес*]	сбросить байт
clrh [*адрес*]	**sth** **%g0**, [*адрес*]	сбросить полуслово
clr [*адрес*]	**st** **%g0**, [*адрес*]	сбросить слово
mov *reg_or_imm*, reg_{rd}	**or** **%g0**, reg_or_imm, reg_{rd}	
mov **%y**, reg_{rd}	**rd** **%y**, reg_{rd}	
mov **%asr***n*, reg_{rd}	**rd** **%asr***n*, reg_{rd}	
mov **%psr**, reg_{rd}	**rd** **%psr**, reg_{rd}	
mov **%wim**, reg_{rd}	**rd** **%wim**, reg_{rd}	
mov **%tbr**, reg_{rd}	**rd** **%tbr**, reg_{rd}	
mov *reg_or_imm*, **%y**	**wr** **%g0**, *reg_or_imm*, **%y**	
mov *reg_or_imm*, **%asr***n*	**wr** **%g0**, *reg_or_imm*, **%asr***n*	
mov *reg_or_imm*, **%psr**	**wr** **%g0**, *reg_or_imm*, **%psr**	
mov *reg_or_imm*, **%wim**	**wr** **%g0**, *reg_or_imm*, **%wim**	
mov *reg_or_imm*, **%tbr**	**wr** **%g0**, *reg_or_imm*, **%tbr**	

_____

[1)] Стандартное сохранение должно использоваться только в коде ядра.

[2)] Не используйте инструкцию set в слоте задержки DCTI.

Определение инструкций

Далее по тексту приведено описание каждой инструкции SPARC. Взаимосвязанные инструкции сгруппированы в подразделы, каждый из которых состоит из пяти частей:

1. Таблица кодов операций, определяемых в подразделе, со значениями поля(-ей), однозначно идентифицирующие инструкцию(-и).
2. Иллюстрация применяемого формата(-ов) инструкции. Описание самих форматов приведено в пункте «6.1.1 Форматы инструкций».
3. Список предлагаемых синтаксических структур на языке ассемблера.
4. Описание существенных характеристик, ограничений и условий возникновения системных прерываний. Следует отметить, что в данном описании символ «&» обозначает соединение битовых векторов. Запятая «,» в левой части выражения разделяет части, которые объединяются для осуществления присвоения. Например, если X, Y и Z – однобитовые векторы и двухбитовый вектор T равен 11_2, то: (X, Y, Z) ← 0&T – обозначает, что X = 0, Y = 1 и Z = 1.
5. Список системных прерываний, которые могут произойти при попытке выполнения инструкции(-ий). Системные прерывания instruction_access_error, instruction_access_exception или r_register_access_error, а также внешние запросы на прерывание не указаны в списке, поскольку они могут произойти при выполнении любой инструкции. Также, любая инструкция может вызвать системное прерывание illegal_instruction, если она не была реализована аппаратно.

В настоящем приложении не приводится какая-либо информация о временных характеристиках (ни в периодах, ни во временных отсчетах), т.к. они зависят от реализации.

В таблице А.2 приводится набор инструкций, определение каждой из этих инструкций приводится в таблицах А.3 – А.48.

Таблица А.2 – Набор инструкций

Код операции	Имя
LDSB (LDSBA*)	Загрузка байта со знаком (из альтернативного адресного пространства)
LDSH (LDSHA*)	Загрузка полуслова со знаком (из альтернативного адресного пространства)
LDUB (LDUBA*)	Загрузка байта без знака (из альтернативного адресного пространства)
LDUH (LDUHA*)	Загрузка полуслова без знака (из альтернативного адресного пространства)
LD (LDA*)	Загрузка слова (из альтернативного адресного пространства)
LDD (LDDA*)	Загрузка двойного слова (из альтернативного адресного пространства)
LDF	Загрузка слова данных с плавающей запятой
LDDF	Загрузка двойного слова данных с плавающей запятой
LDFSR	Загрузка в регистр состояния операций с плавающей запятой
LDC	Загрузка слова данных сопроцессора
LDDC	Загрузка двойного слова данных сопроцессора
LDCSR	Загрузка в регистр состояния сопроцессора
STB (STBA*)	Сохранение байта (в альтернативное адресное пространство)
STH (STHA*)	Сохранение полуслова (в альтернативное адресное пространство)
ST (STA*)	Сохранение слова (в альтернативное адресное пространство)
STD (STDA*)	Сохранение двойного слова (в альтернативное адресное пространство)
STF	Сохранение слова данных с плавающей запятой
STDF	Сохранение двойного слова данных с плавающей запятой
STFSR	Сохранение регистра состояния операций с плавающей запятой
STDFQ*	Сохранение двойного слова очереди отложенных системных прерываний операций с плавающей запятой
STC	Сохранение слова данных сопроцессора
STDC	Сохранение двойного слова данных сопроцессора
STCSR	Сохранение регистра состояния сопроцессора
STDCQ*	Сохранение двойного слова очереди отложенных системных прерываний сопроцессора
LDSTUB (LDSTUBA*)	Атомарная загрузка-сохранение байта без знака (в альтернативном адресном пространстве)
SWAP (SWAPA*)	Обмен содержимого регистра r и памяти (в альтернативном адресном пространстве)
SETHI	Установка старших 22 битов регистра r
NOP	Нет операции
AND (ANDcc)	И (с модификацией icc)
ANDN (ANDNcc)	НЕ-И (с модификацией icc)
OR (ORcc)	Включающее ИЛИ (с модификацией icc)
ORN (ORNcc)	Включающее НЕ-ИЛИ (с модификацией icc)
XOR (XORcc)	Исключающее ИЛИ (с модификацией icc)
XNOR (XNORcc)	Исключающее НЕ-ИЛИ (с модификацией icc)
SLL	Логический сдвиг влево
SRL	Логический сдвиг вправо
SRA	Арифметический сдвиг вправо
ADD (ADDcc)	Сложение (с модификацией icc)
ADDX (ADDXcc)	Сложение с учетом переноса (с модификацией icc)
TADDcc (TADDccTV)	Сложение с меткой и с модификацией icc (системное прерывание при переполнении)
SUB (SUBcc)	Вычитание (с модификацией icc)
SUBX (SUBXcc)	Вычитание с учетом переноса (с модификацией icc)

Окончание таблицы А.2

Код операции	Имя
TSUBcc (TSUBccTV)	Вычитание с меткой и с модификацией icc (системное прерывание при переполнении)
MULScc	Шаг умножения (с модификацией icc)
UMUL (UMULcc) SMUL (SMULcc)	Целочисленное умножение без знака (с модификацией icc) Целочисленное умножение со знаком (с модификацией icc)
UDIV (UDIVcc) SDIV (SDIVcc)	Целочисленное деление без знака (с модификацией icc) Целочисленное деление со знаком (с модификацией icc)
SAVE RESTORE	Сохранение окна вызывающей подпрограммы Восстановление окна вызывавшей подпрограммы
Bicc FBfcc CBccc	Ветвление по целочисленным кодам условий Ветвление по кодам условий операций с плавающей запятой Ветвление по кодам условий сопроцессора
CALL JMPL RETT*	Вызов подпрограммы и создание связи Переход и создание связи Возврат из системного прерывания
Ticc	Системное прерывание по целочисленным кодам условий
RDASR** RDY RDPSR* RDWIM* RDTBR*	Считывание вспомогательного регистра состояния Считывание регистра Y Считывание регистра состояния процессора Считывание регистра маски недопустимости окна Считывание базового регистра системных прерываний
WRASR**	Запись вспомогательного регистра состояния
WRY	Запись регистра Y
WRPSR*	Запись регистра состояния процессора
WRWIM*	Запись регистра маски недопустимости окна
WRTBR*	Запись базового регистра системных прерываний
STBAR	Барьер сохранений
UNIMP	Не реализовано
FLUSH	Сброс памяти инструкций
FPop	Операции с плавающей запятой: FiTO(s,d,q), F(s,d,q)TOi, FsTOd, FsTOq, FdTOs, FdTOq, FqTOs, FqTOd, FMOVs, FNEGs, FABSs, FSQRT(s,d,q), FADD(s,d,q), FSUB(s,d,q), FMUL(s,d,q), FDIV(s,d,q), FsMULd, FdMULq, FCMP(s,d,q), FCMPE(s,d,q)
CPop	Операции сопроцессора (в зависимости от реализации)

_____

* Привилегированная инструкция.

** Инструкция привилегирована, если ссылается на привилегированный регистр ASR.

Инструкции целочисленной загрузки

Таблица А.3 – Инструкции целочисленной загрузки

Код операции	op3	Операция
LDSB	001001	Загрузка байта со знаком
LDSH	001010	Загрузка полуслова со знаком
LDUB	000001	Загрузка байта без знака
LDUH	000010	Загрузка полуслова без знака
LD	000000	Загрузка слова
LDD	000011	Загрузка двойного слова

Окончание таблицы А.3

Код операции	op3	Операция
LDSBA*	011001	Загрузка байта со знаком из альтернативного адресного пространства
LDSHA*	011010	Загрузка полуслова со знаком из альтернативного адресного пространства
LDUBA*	010001	Загрузка байта без знака из альтернативного адресного пространства
LDUHA*	010010	Загрузка полуслова без знака из альтернативного адресного пространства
LDA*	010000	Загрузка слова из альтернативного адресного пространства
LDDA*	010011	Загрузка двойного слова из альтернативного адресного пространства

_____

* Привилегированная инструкция.

Формат 3:

11	rd	op3	rs1	i = 0	asi	rs2
31 30	29 25	24 19	18 14	13	12 5	4 0

11	rd	op3	rs1	i = 1	simm13
31 30	29 25	24 19	18 14	13	12 0

Предлагаемый синтаксис на языке ассемблера:

ldsb [*адрес*], reg_{rd}
ldsh [*адрес*], reg_{rd}
ldub [*адрес*], reg_{rd}
lduh [*адрес*], reg_{rd}
ld [*адрес*], reg_{rd}
ldd [*адрес*], reg_{rd}
ldsba [*regaddr*] *asi*, reg_{rd}
ldsha [*regaddr*] *asi*, reg_{rd}
lduba [*regaddr*] *asi*, reg_{rd}
lduha [*regaddr*] *asi*, reg_{rd}
lda [*regaddr*] *asi*, reg_{rd}
ldda [*regaddr*] *asi*, reg_{rd}

Описание

Инструкция целочисленной загрузки копирует байт, полуслово или слово из памяти в r[*rd*]. Извлеченный байт или полуслово будут размещены в регистре назначения r[*rd*] с выравниванием по правому краю; левая часть заполняется либо разрядом знака, либо нулями, в зависимости от того, была ли задана через код операции операция со знаком или без знака соответственно.

Инструкция целочисленной загрузки двойного слова (LDD, LDDA) перемещает двойное слово из памяти в пару регистров r. Старшее значащее слово из эффективного адреса памяти перемещается в четный регистр r. Младшее значащее слово (из эффективного адреса памяти + 4) перемещается в следующий нечетный регистр r. Следует отметить, что при загрузке двойного слова при rd = 0 изменяется только r[1]. Младший значащий бит поля rd не используется и должен быть сброшен программным обеспечением. Попытка выполнения инструкции загрузки двойного слова, которая ссылается на неверно выравненный номер регистра назначения (нечетный), может вызвать системное прерывание illegal_instruction.

Эффективный адрес инструкции загрузки – «r[*rs1*] + r[*rs2*]», если бит i сброшен, или «r[*rs1*] + sign_ext(*simm13*)», если бит i установлен. Инструкции, осуществляющие загрузку из альтернативного адресного пространства, в поле asi содержат идентификатор адресного пространства, используемого при загрузке, при этом бит i должен быть

сброшен, в противном случае срабатывает системное прерывание illegal_instruction. Инструкции загрузки, которые выполняются без задания альтернативного адресного пространства, реализуют доступ либо к пространству данных пользователя, либо к пространству данных системы, в соответствии с состоянием бита S в PSR.

Успешное выполнение инструкции загрузки (особенно загрузки двойного слова) осуществляется атомарно.

LD и LDA вызывают системное прерывание mem_address_not_aligned, если эффективный адрес не выровнен по границе слова; LDUH, LDSH, LDUHA и LDSHA, если эффективный адрес не выровнен по границе полуслова; LDD и LDDA, если эффективный адрес не выровнен по границе двойного слова.

Замечание по реализации

В ходе выполнения инструкции загрузки двойного слова, если в цикле памяти генерируется ошибка в момент загрузки второго слова, регистр(-ы) назначения может(-гут) быть изменен(-ы) до обработки системного прерывания.

Системные прерывания:

- illegal_instruction (загрузка в альтернативное адресное пространство при i = 1; LDD, LDDA при нечетном rd);
- privileged_instruction (только для загрузки в альтернативное адресное пространство);
- mem_address_not_aligned (исключая LDSB, LDSBA, LDUB, LDUBA);
- data_access_exception;
- data_access_error.

Инструкции загрузки данных с плавающей запятой

Таблица А.4 – Инструкции загрузки данных с плавающей запятой

Код операции	op3	Операция
LDF	100000	Загрузка слова данных с плавающей запятой
LDDF	100011	Загрузка двойного слова данных с плавающей запятой
LDFSR	100001	Загрузка в регистр состояния операций с плавающей запятой

Формат 3:

11	rd	op3	rs1	i = 0	unused (zero)	rs2
31 30	29 25	24 19	18 14	13	12 5	4 0

11	rd	op3	rs1	i = 1	simm13
31 30	29 25	24 19	18 14	13	12 0

Предлагаемый синтаксис на языке ассемблера:

ld [*адрес*], $freg_{rd}$
ldd [*адрес*], $freg_{rd}$
ld [*адрес*], **%fsr**

Описание

Инструкция загрузки слова данных с плавающей запятой (LDF) осуществляет перемещение слова данных из памяти в f[*rd*].

Инструкция загрузки двойного слова данных с плавающей запятой (LDDF) перемещает двойное слово из памяти в пару регистров f. Старшее значащее слово из эффективного адреса памяти перемещается в четный регистр f. Младшее значащее слово (из эффективного адреса памяти + 4) перемещается в следующий нечетный регистр f. Младший значащий бит поля rd не используется и должен быть сброшен программным

обеспечением. Если данный бит установлен, рекомендуется чтобы LDDF вызвала системное прерывание fp_exception с FSR.ftt = invalid_fp_register.

Инструкция загрузки в регистр состояния операций с плавающей запятой (LDFSR) ждет, пока не будут выполнены все инструкции FPop, после чего слово из памяти загружается в FSR. Если какая-либо из трех инструкций, выполняемых после (по времени) LDFSR является FBfcc, значение поля fcc в FSR, видимое для FBfcc, будет не определено.

Эффективный адрес инструкции загрузки – «r[*rs1*] + r[*rs2*]», если бит i сброшен, или «r[*rs1*] + sign_ext(*simm13*)», если бит i установлен.

LDF и LDFSR вызывают системное прерывание mem_address_not_aligned, если эффективный адрес не выровнен по границе слова; LDDF, если эффективный адрес не выровнен по границе двойного слова. Если бит EF в PSR сброшен, или если FPU отсутствует, инструкция загрузки данных с плавающей запятой вызовет системное прерывание fp_disabled.

Замечание по реализации

Если выполнение инструкции загрузки данных с плавающей запятой было прервано системным прерыванием data_access_exception, регистр(-ы) назначения *f* либо остается(-ются) неизменным(-и), либо в него(них) попадет, зависящая от реализации, предопределенная константа.

Системные прерывания:
- fp_disabled;
- fp_exception (sequence_error, invalid_fp_register(LDDF));
- data_access_exception;
- data_access_error;
- mem_address_not_aligned.

Инструкции загрузки сопроцессора

Таблица А.5– Инструкции загрузки сопроцессора

Код операции	op3	Операция
LDC	110000	Загрузка слова данных сопроцессора
LDDC	110011	Загрузка двойного слова данных сопроцессора
LDCSR	110001	Загрузка в регистр состояния сопроцессора

Формат 3:

11	rd	op3	rs1	i = 0	unused (zero)	rs2
31 30	29 25	24 19	18 14	13	12 5	4 0

11	rd	op3	rs1	i = 1	simm13
31 30	29 25	24 19	18 14	13	12 0

Предлагаемый синтаксис на языке ассемблера:

ld [*адрес*], $creg_{rd}$
ldd [*адрес*], $creg_{rd}$
ld [*адрес*], **%csr**

Описание

Инструкция загрузки слова данных сопроцессора (LDC) перемещает слово данных из памяти в регистр сопроцессора. Инструкция загрузки двойного слова данных сопроцессора (LDDC) перемещает двойное слово из памяти в пару регистров сопроцессора. Инструкция загрузки данных в регистр статуса сопроцессора (LDCSR) перемещает слово данных из памяти в регистр состояния сопроцессора. Семантика данных инструкций зависит от реализации присоединенного сопроцессора.

Эффективный адрес инструкции загрузки – «r[*rs1*] + r[*rs2*]», если бит i сброшен, или «r[*rs1*] + sign_ext(*simm13*)», если бит i установлен.

LDC и LDCSR вызывают системное прерывание mem_address_not_aligned, если эффективный адрес не выровнен по границе слова; LDDC, если эффективный адрес не выровнен по границе двойного слова. Если бит EC в PSR сброшен, или если сопроцессор отсутствует, инструкция загрузки сопроцессора вызывает системное прерывание cp_disabled.

Замечание по реализации

Конкретная реализация может вызывать системное прерывание data_access_exception из-за ошибки, «после которой невозможно восстановление в автоматическом режиме», во время выполнения доступа к памяти «по эффективному адресу + 4», хотя в соответствующий доступ «по эффективному адресу» не вызвал ошибки. Таким образом, в данном случае четный регистр назначения сопроцессора может измениться. Следует отметить, что подобное не может произойти на границе страницы, ввиду ограничений по выравниванию по границе двойного слова.

Системные прерывания:
- cp_disabled;
- cp_exception;
- mem_address_not_aligned;
- data_access_exception;
- data_access_error.

Инструкции целочисленного сохранения

Таблица А.6 – Инструкции целочисленного сохранения

Код операции	op3	Операция
STB	000101	Сохранение байта
STH	000110	Сохранение полуслова
ST	000100	Сохранение слова
STD	000111	Сохранение двойного слова
STBA*	010101	Сохранение байта в альтернативное адресное пространство
STHA*	010110	Сохранение полуслова в альтернативное адресное пространство
STA*	010100	Сохранение слова в альтернативное адресное пространство
STDA*	010111	Сохранение двойного слова в альтернативное адресное пространство
_____ * Привилегированная инструкция.		

Формат 3:

11	rd	op3	rs1	i = 0	asi	rs2
31 30	29 25	24 19	18 14	13	12 5	4 0

11	rd	op3	rs1	i = 1	simm13
31 30	29 25	24 19	18 14	13	12 0

Предлагаемый синтаксис на языке ассемблера:

stb reg_{rd}, [*адрес*] (эквивалентно **stub**, **stsb**)
sth reg_{rd}, [*адрес*] (эквивалентно **stuh**, **stsh**)
st reg_{rd}, [*адрес*]
std reg_{rd}, [*адрес*]
stba reg_{rd}, [*regaddr*] *asi* (эквивалентно **stuba**, **stsba**)
stha reg_{rd}, [*regaddr*] *asi* (эквивалентно **stuha**, **stsha**)

sta reg_{rd}, [*regaddr*] *asi*
stda reg_{rd}, [*regaddr*] *asi*

Описание

Инструкция целочисленного сохранения копирует слово, младшее значащее полуслово или младший значащий байт из r[*rd*] в память.

Инструкция целочисленного сохранения двойного слова (STD, STDA) копирует двойное слово из пары регистров r в память. Старшее значащее слово (в регистре r с четным номером) записывается в память по эффективному адресу. Младшее значащее слово (в следующем регистре r с нечетным номером) записывается в память по «эффективному адресу + 4». Младший значащий бит поля rd в инструкции сохранения двойного слова не используется и должен быть сброшен программным обеспечением. Попытка выполнения инструкции сохранения двойного слова, которая ссылается на не выровненный rd (нечетный), вызовет системное прерывание illegal_instruction.

Эффективный адрес инструкции сохранения – «r[*rs1*] + r[*rs2*]», если бит i сброшен, или «r[*rs1*] + sign_ext(*simm13*)», если бит i установлен. Инструкции, осуществляющие сохранение в альтернативное адресное пространство, в поле asi содержат идентификатор адресного пространства, используемого при сохранении, при этом бит i должен быть сброшен, в противном случае срабатывает системное прерывание illegal_instruction. Инструкции сохранения, которые выполняются без задания альтернативного адресного пространства, реализуют доступ либо к пространству данных пользователя, либо к пространству данных системы, в соответствии с состоянием бита S в PSR.

Успешное выполнение инструкции сохранения (особенно сохранения двойного слова) осуществляется атомарно.

ST и STA вызывают системное прерывание mem_address_not_aligned, если эффективный адрес не выровнен по границе слова; STH и STHA, если эффективный адрес не выровнен по границе полуслова; STD и STDA, если эффективный адрес не выровнен по границе двойного слова.

Замечание по реализации

Конкретная реализация может вызывать системное прерывание data_access_exception из-за ошибки, «после которой невозможно восстановление в автоматическом режиме», во время выполнения доступа к памяти «по эффективному адресу + 4», хотя в соответствующий доступ «по эффективному адресу» не вызвал ошибки. Таким образом, в данном случае данные в памяти, которые располагаются по эффективному адресу, могут измениться. Следует отметить, что подобное не может произойти на границе страницы, ввиду ограничений по выравниванию по границе двойного слова.

Системные прерывания:

- illegal_instruction (сохранение в альтернативное адресное пространство при i = 1; STD, STDA при нечетном rd);
- privileged_instruction (только для сохранения в альтернативное адресное пространство);
- mem_address_not_aligned (исключая STB и STBA);
- data_access_exception;
- data_access_error;
- data_store_error.

Инструкции сохранения данных с плавающей запятой

Таблица А.7 – Инструкции сохранения данных с плавающей запятой

Код операции	op3	Операция
STF	100100	Сохранение слова данных с плавающей запятой
STDF	100111	Сохранение двойного слова данных с плавающей запятой
STFSR	100101	Сохранение регистра состояния операций с плавающей запятой
STDFQ*	100110	Сохранение двойного слова очереди отложенных системных прерываний операций с плавающей запятой

_____

* Привилегированная инструкция.

Формат 3:

11	rd	op3	rs1	i = 0	unused (zero)	rs2
31 30	29 25	24 19	18 14	13	12 5	4 0

11	rd	op3	rs1	i = 1	simm13
31 30	29 25	24 19	18 14	13	12 0

Предлагаемый синтаксис на языке ассемблера:

st $freg_{rd}$, [*адрес*]
std $freg_{rd}$, [*адрес*]
st **%fsr**, [*адрес*]
std **%fq**, [*адрес*]

Описание

Инструкции сохранения слова данных с плавающей запятой (STF) копирует f[*rd*] в память.

Инструкция сохранения двойного слова данных с плавающей запятой (STDF) копирует двойное слово из пары регистров f в память. Старшее значащее слово (в регистре f с четным номером) записывается в память по эффективному адресу; младшее значащее слово (в регистре f с нечетным номером) записывается в память по «эффективному адресу + 4». Младший значащий бит поля rd не используется и должен быть сброшен программным обеспечением. Если данный бит установлен, рекомендуется чтобы STDF вызвала системное прерывание fp_exception с FSR.ftt = invalid_fp_register.

Инструкция сохранения данных очереди отложенных прерываний операций с плавающей запятой (STDFQ) сохраняет первое двойное слово очереди операций с плавающей запятой (FQ) в память. Попытка выполнения инструкции STDFQ, если в реализации отсутствует очередь операций с плавающей запятой, вызовет системное прерывание fp_exception с FSR.ftt равным 4 (sequence_error). Если в реализации присутствует очередь операций с плавающей запятой, то при попытке выполнения инструкции STDFQ при пустой FQ (FSR.qne = 0) будет сгенерировано системное прерывание fp_exception с FSR.ftt равным 4 (sequence_error). Дополнительная семантика данной инструкции зависит от реализации.

Инструкция сохранения данных в регистр состояния операций с плавающей запятой (STFSR) ждет, пока не будут выполнены все инструкции FPop, после чего FSR записывается в память. STFSR может сбрасывать FSR.ftt после записи FSR в память.

Эффективный адрес инструкции сохранения – «r[*rs1*] + r[*rs2*]», если бит i сброшен, или «r[*rs1*] + sign_ext(*simm13*)», если бит i установлен.

STF и STFSR вызывают системное прерывание mem_address_not_aligned, если эффективный адрес не выровнен по границе слова; STDF и STDFQ, если эффективный адрес не выровнен по границе двойного слова. Если бит EF в PSR сброшен, либо если FPU отсутствует, инструкция сохранения данных с плавающей запятой вызовет системное прерывание fp_disabled.

Замечание по реализации

Конкретная реализация может вызывать системное прерывание data_access_exception из-за ошибки, «после которой невозможно восстановление в автоматическом режиме», во время выполнении доступа к памяти «по эффективному адресу + 4», хотя в соответствующий доступ «по эффективному адресу» не вызвал ошибки. Таким образом, в данном случае данные в памяти, которые располагаются по эффективному адресу, могут измениться. Следует отметить, что подобное не может произойти на границе страницы, ввиду ограничений по выравниванию по границе двойного слова.

Системные прерывания:
- fp_disabled;
- fp_exception (sequence_error(STDFQ), invalid_fp_register(STDF, STDFQ));
- privileged_instruction (только при STDFQ);
- mem_address_not_aligned;
- data_access_exception;
- data_access_error;
- data_store_error.

Инструкции сохранения сопроцессора

Таблица А.8 – Инструкции сохранения сопроцессора

Код операции	op3	Операция
STC	110100	Сохранение слова данных сопроцессора
STDC	110111	Сохранение двойного слова данных сопроцессора
STCSR	110101	Сохранение регистра состояния сопроцессора
STDCQ*	110110	Сохранение двойного слова очереди отложенных системных прерываний сопроцессора

_____

* Привилегированная инструкция.

Формат 3:

11	rd	op3	rs1	i = 0	unused (zero)	rs2
31 30	29 25	24 19	18 14	13	12 5	4 0

11	rd	op3	rs1	i = 1	simm13
31 30	29 25	24 19	18 14	13	12 0

Предлагаемый синтаксис на языке ассемблера:

st $creg_{rd}$, [*адрес*]
std $creg_{rd}$, [*адрес*]
st **%csr**, [*адрес*]
std **%cq**, [*адрес*]

Описание

Инструкция сохранения слова данных сопроцессора (STC) копирует содержимое регистра сопроцессора в память.

Инструкция сохранения двойного слова данных сопроцессора (STDC) копирует содержимое пары регистров сопроцессора в память.

Инструкция сохранения регистра состояния сопроцессора (STCSR) копирует содержимое регистра состояния сопроцессора в память. Инструкция сохранения двойного слова очереди сопроцессора (STDCQ) перемещает первую запись очереди сопроцессора в память. Если в реализации отсутствует очередь сопроцессора, STDCQ может вызвать

системное прерывание cp_exception. Семантика данных инструкций зависит от реализации присоединенного сопроцессора, если таковой имеется.

Эффективный адрес инструкции сохранения – «r[*rs1*] + r[*rs2*]», если бит i сброшен, или «r[*rs1*] + sign_ext(*simm13*)», если бит i установлен.

STC и STCSR вызывают системное прерывание mem_address_not_aligned, если эффективный адрес не выровнен по границе слова. STDC и STDCQ, если эффективный адрес не выровнен по границе двойного слова. Если бит EC в PSR сброшен, либо если сопроцессор отсутствует, инструкция сохранения сопроцессора вызывает системное прерывание cp_disabled.

Системные прерывания:
- cp_disabled;
- cp_exception;
- privileged_instruction (только для STDCQ);
- mem_address_not_aligned;
- data_access_exception;
- illegal_instruction (только для STDCQ; зависит от реализации);
- data_access_error;
- data_store_error.

Инструкции атомарной загрузки-сохранения байта без знака

Таблица А.9 – Инструкции атомарной загрузки-сохранения байта без знака

Код операции	op3	Операция
LDSTUB	001101	Атомарная загрузка-сохранение байта без знака
LDSTUBA	011101	Атомарная загрузка-сохранение байта без знака в альтернативном адресном пространстве

Формат 3:

11	rd	op3	rs1	i = 0	asi	rs2
31 30	29 25	24 19	18 14	13	12 5	4 0

11	rd	op3	rs1	i = 1	simm13
31 30	29 25	24 19	18 14	13	12 0

Предлагаемый синтаксис на языке ассемблера:
ldstub [*адрес*], reg_{rd}
ldstuba [*regaddr*] *asi*, reg_{rd}

Описание

При помощи инструкций атомарной загрузки-сохранения байта без знака байт из памяти копируется в r[*rd*], после чего адресуемый байт в памяти перезаписывается со всеми установленными битами. Операция выполняется атомарно, т.е. без допуска промежуточных прерывающих или отложенных системных прерываний. В мультипроцессорной системе два или более процессора, выполняющие инструкции атомарной загрузки-сохранения байта без знака, инструкции SWAP или SWAPA, одновременно адресующие один и тот же байт или слово, гарантированно будут выполнять инструкции последовательно, но порядок их следования будет не определен.

Эффективный адрес инструкции атомарной загрузки-сохранения – «r[*rs1*] + r[*rs2*]», если бит i сброшен, или «r[*rs1*] + sign_ext(*simm13*)», если бит i установлен. Бит i в LDSTUBA должен быть сброшен, в противном случае произойдет системное прерывание illegal_instruction. Идентификатор адресного пространства, используемый для доступа к памяти, берется из поля asi. Для LDSTUB доступно или адресное пространство данных

пользователя или пространство данных системы, в соответствии с состоянием бита S в PSR.

Замечание по реализации

Конкретная реализация может вызывать системное прерывание data_access_exception из-за ошибки, «после которой невозможно восстановление в автоматическом режиме», во время выполнения сохранения в память, хотя соответствующий доступ загрузки не вызвал ошибки. Таким образом, в данном случае регистр назначения может измениться.

Системные прерывания:
- illegal_instruction (только для LDSTUBA при i = 1);
- privileged_instruction (только для LDSTUBA);
- data_access_exception;
- data_access_error;
- data_store_error.

Инструкции обмена содержимого регистра и памяти

Таблица А.10 – Инструкции обмена содержимого регистра и памяти

Код операции	op3	Операция
SWAP	001111	Обмен содержимого регистра и памяти
SWAPA*	011111	Обмен содержимого регистра и памяти в альтернативном адресном пространстве
_____ * Привилегированная инструкция.		

Формат 3:

11	rd	op3	rs1	i = 0	asi	rs2
31 30	29 25	24 19	18 14	13	12 5	4 0

11	rd	op3	rs1	i = 1	simm13
31 30	29 25	24 19	18 14	13	12 0

Предлагаемый синтаксис на языке ассемблера:
swap [*адрес*], reg_{rd}
swapa [*regaddr*] *asi*, reg_{rd}

Описание

Инструкции SWAP и SWAPA выполняют обмен данных r[*rd*] и содержимого слова в адресуемой ячейке памяти. Операция выполняется атомарно, т.е. без допуска прерывающих или отложенных системных прерываний. В мультипроцессорной системе два или более процессора, выполняющие инструкции SWAP, SWAPA или инструкции атомарной загрузки-сохранения байта без знака, одновременно адресующие одно и то же слово или байт, гарантированно будут выполнять инструкции последовательно, но порядок их следования будет не определен.

Эффективный адрес инструкции SWAP – «r[*rs1*] + r[*rs2*]», если бит i сброшен, или «r[*rs1*] + sign_ext(*simm13*)», если бит i установлен. Бит i в SWAPA должен быть сброшен, в противном случае срабатывает системное прерывание illegal_instruction. Идентификатор адресного пространства, используемый для доступа к памяти, берется из поля asi. Для SWAP доступно или адресное пространство данных пользователя или пространство данных системы, в соответствии с состоянием бита S в PSR.

Данные инструкции вызывают системное прерывание mem_address_not_aligned, если эффективный адрес не выровнен по границе слова.

Замечание по реализации

Конкретная реализация может вызывать системное прерывание data_access_exception из-за ошибки, «после которой невозможно восстановление в автоматическом режиме», во время выполнения сохранения в память, хотя соответствующий доступ загрузки не вызвал ошибки. Таким образом, в данном случае регистр назначения может измениться.

Системные прерывания:
- illegal instruction (только для SWAPA при i = 1);
- privileged_instruction (только для SWAPA);
- mem_address_not_aligned;
- data_access_exception;
- data_access_error;
- data_store_error.

Инструкция SETHI

Таблица А.11 – Инструкция SETHI

Код операции	op	op2	Операция
SETHI	00	100	Установка 22 битов старшего разряда

Формат 2:

00	rd	100	imm22
31 30	29 25	24 22	21 0

Предлагаемый синтаксис на языке ассемблера:
sethi *const22*, reg_{rd}
sethi **%hi**(*значение*), reg_{rd}

Описание

Инструкция SETHI сбрасывает 10 младших значащих битов в r[*rd*] и заменяет старшие 22 разряда значением поля imm22.

SETHI не изменяет коды условий.

Инструкция SETHI при rd = 0 и imm22 = 0 определяется, как инструкция NOP.

Системные прерывания: отсутствуют.

Инструкция NOP

Таблица А.12 – Инструкция NOP

Код операции	op	op2	Операция
NOP	00	100	Операция отсутствует

Формат 2:

00	00000	100	0000000000000000000000
31 30	29 25	24 22	21 0

Предлагаемый синтаксис на языке ассемблера:
nop

Описание

Инструкция NOP не изменяет видимое программным путем состояние (за исключением PC и nPC).

Следует отметить, что NOP – это частный случай инструкции SETHI с imm22 = 0 и rd = 0.

Системные прерывания: отсутствуют.

Инструкции логических операций

Таблица А.13 – Инструкции логических операций

Код операции	op3	Операция
AND	000001	И
ANDcc	010001	И с модификацией icc
ANDN	000101	НЕ-И
ANDNcc	010101	НЕ-И с модификацией icc
OR	000010	Включающее ИЛИ
ORcc	010010	Включающее ИЛИ с модификацией icc
ORN	000110	Включающее НЕ-ИЛИ
ORNcc	010110	Включающее НЕ-ИЛИ с модификацией icc
XOR	000011	Исключающее ИЛИ
XORcc	010011	Исключающее ИЛИ с модификацией icc
XNOR	000111	Исключающее НЕ-ИЛИ
XNORcc	010111	Исключающее НЕ-ИЛИ с модификацией icc

Формат 3:

10	rd	op3	rs1	i = 0	unused (zero)	rs2
31 30	29 25	24 19	18 14	13	12 5	4 0

10	rd	op3	rs1	i = 1	simm13
31 30	29 25	24 19	18 14	13	12 0

Предлагаемый синтаксис на языке ассемблера:

and reg_{rs1}, *reg_or_imm*, reg_{rd}
andcc reg_{rs1}, *reg_or_imm*, reg_{rd}
andn reg_{rs1}, *reg_or_imm*, reg_{rd}
andncc reg_{rs1}, *reg_or_imm*, reg_{rd}
or reg_{rs1}, *reg_or_imm*, reg_{rd}
orcc reg_{rs1}, *reg_or_imm*, reg_{rd}
orn reg_{rs1}, *reg_or_imm*, reg_{rd}
orncc reg_{rs1}, *reg_or_imm*, reg_{rd}
xor reg_{rs1}, *reg_or_imm*, reg_{rd}
xorcc reg_{rs1}, *reg_or_imm*, reg_{rd}
xnor reg_{rs1}, *reg_or_imm*, reg_{rd}
xnorcc reg_{rs1}, *reg_or_imm*, reg_{rd}

Описание

Данные инструкции осуществляют поразрядные логические операции. Они выполняют расчет «r[*rs1*] *операция* r[*rs2*]», если бит i сброшен, или «r[*rs1*] *операция* sign_ext(*simm13*)», если бит i установлен, с записью результата в r[*rd*].

ANDcc, ANDNcc, ORcc, ORNcc, XORcc и XNORcc модифицируют целочисленные коды условий (icc).

В инструкциях ANDN, ANDNcc, ORN и ORNcc перед осуществлением основной операции (И или ИЛИ) над вторым операндом выполняется логическая операция НЕ.

Замечания по программированию

XNOR и XNORcc логически реализуют XOR-Not и XOR-Not-cc, соответственно.

Системные прерывания: отсутствуют.

Инструкции сдвига

Таблица А.14 – Инструкции сдвига

Код операции	op3	Операция
SLL	100101	Логический сдвиг влево
SRL	100110	Логический сдвиг вправо
SRA	100111	Арифметический сдвиг вправо

Формат 3:

10	rd	op3	rs1	i = 0	unused (zero)	rs2
31 30	29 25	24 19	18 14	13	12 5	4 0
10	rd	op3	rs1	i = 1	unused (zero)	shcnt
31 30	29 25	24 19	18 14	31	12 5	4 0

Предлагаемый синтаксис на языке ассемблера:

sll reg_{rs1}, *reg_or_imm*, reg_{rd}

srl reg_{rs1}, *reg_or_imm*, reg_{rd}

sra reg_{rs1}, *reg_or_imm*, reg_{rd}

Описание

Величина сдвига для данных инструкций определяется пятью младшими значащими битами r[*rs2*], если бит i сброшен, или значением shcnt, если бит i установлен.

Если бит i сброшен, 27 старших значащих разрядов значения в r[*rs2*] не учитываются.

Если бит i установлен, биты с пятого по двенадцатый инструкции сдвига зарезервированы и должны быть сброшены программным обеспечением.

При выполнении SLL значение r[*rs1*] смещается влево на количество битов, задаваемых величиной сдвига.

При выполнении SRL и SRA значение r[*rs1*] смещается вправо на количество битов, задаваемых величиной сдвига.

При выполнении SLL и SRL освобожденные позиции заменяются нулями, при выполнении SRA освобожденные позиции заполняются старшим значащим битом r[*rs1*]. Если величина сдвига равна 0, то сдвиг не выполняется.

При выполнении данных инструкций результат сдвига записывается в r[*rd*].

При выполнении данных инструкций коды условий не модифицируются.

Замечания по программированию

«Арифметического сдвига влево на 1 (и расчета переполнения)» можно добиться при помощи инструкции ADDcc.

Замечание по реализации

В инструкциях сдвига shcnt соответствует 5 младшим значащим битам simm13 в других инструкциях формата 3. Однако биты с пятого по двенадцатый в инструкциях смещения должны быть сброшены.

Системные прерывания: отсутствуют.

Инструкции сложения

Таблица А.15 – Инструкции сложения

Код операции	op3	Операция
ADD	000000	Сложение
ADDcc	010000	Сложение с модификацией icc
ADDX	001000	Сложение с учетом переноса
ADDXcc	011000	Сложение с учетом переноса и с модификацией icc

Формат 3:

10	rd	op3	rs1	i = 0	unused (zero)	rs2
31 30	29 25	24 19	18 14	13	12 5	4 0

10	rd	op3	rs1	i = 1	simm13
31 30	29 25	24 19	18 14	13	12 0

Предлагаемый синтаксис на языке ассемблера:

add reg_{rs1}, *reg_or_imm*, reg_{rd}
addcc reg_{rs1}, *reg_or_imm*, reg_{rd}
addx reg_{rs1}, *reg_or_imm*, reg_{rd}
addxcc reg_{rs1}, *reg_or_imm*, reg_{rd}

Описание

При выполнении ADD и ADDcc рассчитывается «r[*rs1*] + r[*rs2*]», если бит i сброшен, или «r[*rs1*] + sign_ext(*simm13*)», если бит i установлен, с записью суммы в r[*rd*].

При выполнении ADDX и ADDXcc («ADD eXtended») к итоговой сумме также добавляется бит переноса (c) регистра PSR; т.е., рассчитывается «r[*rs1*] + r[*rs2*] + *c*» или «r[*rs1*] + sign_ext(*simm13*) + *c*», с записью результата в r[*rd*].

При выполнении ADDcc и ADDXcc модифицируются целочисленные коды условий (icc). Переполнение при сложении происходит, если оба операнда имеют один знак, а знак суммы – другой.

Системные прерывания: отсутствуют.

Инструкции сложения с меткой

Таблица А.16 – Инструкции сложения с меткой

Код операции	op3	Операция
TADDcc	100000	Сложение с меткой и с модификацией icc
TADDccTV	100010	Сложение с меткой, с модификацией icc и системным прерыванием при переполнении

Формат 3:

10	rd	op3	rs1	i = 0	unused (zero)	rs2
31 30	29 25	24 19	18 14	13	12 5	4 0

10	rd	op3	rs1	i = 1	simm13
31 30	29 25	24 19	18 14	13	12 0

Предлагаемый синтаксис на языке ассемблера:

taddcc reg_{rs1}, *reg_or_imm*, reg_{rd}
taddcctv reg_{rs1}, *reg_or_imm*, reg_{rd}

Описание

При выполнении данных инструкций рассчитывается сумма «r[*rs1*] + r[*rs2*]», если бит i сброшен, или «r[*rs1*] + sign_ext(*simm13*)», если бит i установлен.

При выполнении TADDcc модифицируются целочисленные коды условий (icc); при выполнении TADDccTV также модифицируются целочисленные коды условий, если не происходит системного прерывания.

Переполнение tag_overflow происходит, если бит 1 или бит 0 любого операнда установлен, или если при сложении генерируется арифметическое переполнение (оба компонента операции имеют один знак, а знак суммы – другой).

Если TADDccTV вызывает tag_overflow, генерируется системное прерывание tag_overflow, при этом r[*rd*] и коды условий не меняются. Если TADDccTV не вызывает tag_overflow, выполняется обновление целочисленных кодов условий (в частности, бит переполнения (v) сбрасывается) с записью суммы в r[*rd*].

Если TADDcc вызывает tag_overflow, бит переполнения (v) в PSR устанавливается; если TADDcc не вызывает tag_overflow, бит переполнения сбрасывается. В обоих случаях выполняется обновление остальных целочисленных кодов условий с записью суммы в r[*rd*].

Системные прерывания:
- tag_overflow (только TADDccTV).

Инструкции вычитания

Таблица А.17 – Инструкции вычитания

Код операции	op3	Операция
SUB	000100	Вычитание
SUBcc	010100	Вычитание с модификацией icc
SUBX	001100	Вычитание с учетом переноса
SUBXcc	011100	Вычитание с учетом переноса и с модификацией icc

Формат 3:

10	rd	op3	rs1	i = 0	unused (zero)	rs2
31 30	29 25	24 19	18 14	13	12 5	4 0

10	rd	op3	rs1	i = 1	simm13
31 30	29 25	24 19	18 14	13	12 0

Предлагаемый синтаксис на языке ассемблера:

sub reg_{rs1}, *reg_or_imm*, reg_{rd}
subcc reg_{rs1}, *reg_or_imm*, reg_{rd}
subx reg_{rs1}, *reg_or_imm*, reg_{rd}
subxcc reg_{rs1}, *reg_or_imm*, reg_{rd}

Описание

При выполнении данных инструкций рассчитывается «r[*rs1*] – r[*rs2*]», если бит i сброшен, или «r[*rs1*] – sign_ext(*simm13*)», если бит i установлен, с записью разности в r[*rd*].

При выполнении SUBX и SUBXcc («SUBtract eXtended») из итоговой разности также вычитается бит переноса (c) PSR; т.е. рассчитывается «r[*rs1*] – r[*rs2*] – *c*» или «r[*rs1*] – sign_ext(*simm13*) – *c*» с записью результата в r[*rd*].

При выполнении SUBcc и SUBXcc модифицируются целочисленные коды условий (icc). Переполнение при вычитании происходит, если оба операнда имеют разный знак, а знак разности отличается от знака r[*rs1*].

Замечания по программированию

SUBcc с rd = 0 используется для выполнения целочисленного сравнения со знаком или без знака.

Системные прерывания: отсутствуют.

Инструкции вычитания с меткой

Таблица А.18 – Инструкции вычитания с меткой

Код операции	op3	Операция
TSUBcc	100001	Вычитание с меткой, с модификацией icc
TSUBccTV	100011	Вычитание с меткой, модификацией icc и системным прерыванием при переполнении

Формат 3:

10	rd	op3	rs1	i = 0	unused (zero)	rs2
31 30	29 25	24 19	18 14	13	12 5	4 0

10	rd	op3	rs1	i = 1	simm13
31 30	29 25	24 19	18 14	13	12 0

Предлагаемый синтаксис на языке ассемблера:
tsubcc reg_{rs1}, *reg_or_imm*, reg_{rd}
tsubcctv reg_{rs1}, *reg_or_imm*, reg_{rd}

Описание

При выполнении данных инструкций рассчитывается «r[*rs1*] – r[*rs2*]», если бит i сброшен, или «r[*rs1*] – sign_ext(*simm13*)», если бит i установлен.

При выполнении TSUBcc модифицируются целочисленные коды условий (icc); при выполнении TSUBccTV также модифицируются целочисленные коды условий, если не происходит системного прерывания.

Переполнение tag_overflow происходит, если бит 1 или бит 0 любого операнда установлен, или если при вычитании генерируется арифметическое переполнение (оба компонента операции имеют разный знак, а знак разности отличается от знака r[*rs1*]).

Если TSUBccTV вызывает tag_overflow, генерируется системное прерывание tag_overflow, при этом регистр назначения и коды условий не меняются. Если TSUBccTV не вызывает tag_overflow, выполняется обновление целочисленных кодов условий (в частности, бит переполнения (v) сбрасывается) с записью разности в r[*rd*].

Если TSUBcc вызывает tag_overflow, бит переполнения (v) в PSR устанавливается; если TSUBcc не вызывает tag_overflow, бит переполнения сбрасывается. В обоих случаях выполняется обновление остальных целочисленных кодов условий с записью разности в r[*rd*].

Системные прерывания:
- tag_overflow (только TSUBccTV).

Инструкция шага умножения

Таблица А.19 – Инструкция шага умножения

Код операции	op3	Операция
MULScc	100100	Шаг умножения с модификацией icc

Формат 3:

10	rd	op3	rs1	i = 0	reserved	rs2
31 30	29 25	24 19	18 14	13	12 5	4 0

10	rd	op3	rs1	i = 1	simm13
31 30	29 25	24 19	18 14	13	12 0

Предлагаемый синтаксис на языке ассемблера:

mulscc reg_{rs1}, *reg_or_imm*, reg_{rd}

Описание

MULScc рассматривает r[*rs1*] и регистр Y как один 64-битовый регистр двойного слова с возможностью сдвига вправо. Младший значащий бит r[*rs1*] трактуется как ближайший к старшему значащему биту регистра Y. MULScc осуществляет сложение по условию, основанному на значении младшего значащего бита регистра Y.

Умножение предполагает, что регистр Y первоначально содержит множитель, r[*rs1*] – старшие значащие биты произведения, а r [*rs2*] – сомножитель. После выполнения умножения регистр Y содержит младшие значащие биты произведения.

Следует отметить, что обычно при выполнении инструкции MULScc rs1 = rd. Пример умножения со знаком 32×32 → 64 на базе MULScc.

Выполнение инструкции MULScc:

1) В качестве множителя принимается r[*rs2*], если бит i сброшен, или sign_ext(*simm13*), если бит i установлен.

2) 32-битовое значение вычисляется смещением r[*rs1*] вправо на один бит, старший разряд заменяется результатом «N ИСКЛЮЧАЮЩЕЕ ИЛИ V» из PSR (это корректный знак предыдущего промежуточного произведения).

3) Если младший значащий бит регистра Y равен 1, к смещаемому значению из этапа 2 прибавляется множитель. Если младший значащий бит регистра Y равен 0, к смещаемому значению из этапа 2 прибавляется 0.

4) Сумма из этапа 3 записывается в r[*rd*].

5) Целочисленные коды условий (icc), обновляются в соответствии с операцией сложения, выполненной на этапе 3.

6) Регистр Y смещается вправо на один бит, при этом младший значащий бит несмещенного регистра r[*rs1*] заменяет старший значащий бит регистра Y.

Системные прерывания: отсутствуют.

Инструкции умножения

Таблица А.20 – Инструкции умножения

Код операции	op3	Операция
UMUL	001010	Целочисленное умножение без знака
SMUL	001011	Целочисленное умножение со знаком
UMULcc	011010	Целочисленное умножение без знака и с модификацией icc
SMULcc	011011	Целочисленное умножение со знаком и с модификацией icc

Формат инструкции:

10	rd	op3	rs1	i = 0	unused (zero)	rs2
31 30	29 25	24 19	18 14	13	12 5	4 0

10	rd	op3	rs1	i = 1	simm13
31 30	29 25	24 19	18 14	13	12 0

Предлагаемый синтаксис на языке ассемблера:

umul reg_{rs1}, *reg_or_imm*, reg_{rd}

smul reg_{rs1}, *reg_or_imm*, reg_{rd}

umulcc reg_{rs1}, *reg_or_imm*, reg_{rd}

smulcc reg_{rs1}, *reg_or_imm*, reg_{rd}

Описание

При помощи инструкций умножения выполняется перемножение 32 бит на 32 бита с получением 64-битового результата. В ходе выполнения данных инструкций

рассчитывается «r[*rs1*] x r[*rs2*]», если бит i сброшен, или «r[*rs1*] x sign_ext(*simm13*)», если бит i установлен. В регистр Y записываются 32 старших значащих бита произведения, в регистр r[*rd*] – 32 младших значащих бита.

При выполнении умножения без знака (UMUL, UMULcc) операндами являются целочисленные слова без знака, при этом рассчитывается целочисленное произведение двойного слова без знака. При выполнении умножения со знаком (SMUL, SMULcc) компонентами операции являются целочисленные слова со знаком, при этом рассчитывается целочисленное произведение двойного слова со знаком.

При выполнении UMUL и SMUL биты кодов условий не изменяются. При выполнении UMULcc и SMULcc биты целочисленных кодов условий (icc), записываются согласно таблице, приведенной ниже. Следует отметить, что отрицательное значение (N) и ноль (Z) устанавливаются в соответствии с младшим значащим словом произведения.

Таблица А.21 – Определение флагов при выполнении операции умножения

Бит icc	UMULcc	SMULcc
N	Устанавливается, если результат[31] = 1	Устанавливается, если результат[31] = 1
Z	Устанавливается, если результат[31:0] = 0	Устанавливается, если результат[31:0] = 0
V	Ноль*	Ноль*
C	Ноль*	Ноль*

_____

* Спецификация данного кода условия в последующих версиях архитектуры может измениться. Программное обеспечение не должно проверять данный код условия.

Замечания по программированию

32-битовое переполнение после UMUL / UMULcc обозначается Y ≠ 0. 32-битовое переполнение после SMUL / SMULcc обозначается Y ≠ (r[*rd*] >> 31).

Замечание по реализации

В конкретной реализации может предполагаться, что меньший операнд обычно будет размещен в r[*rs2*] или simm13.

Системные прерывания: отсутствуют.

Инструкции деления

Таблица А.22 – Инструкции деления

Код операции	op3	Операция
UDIV	001110	Целочисленное деление без знака
SDIV	001111	Целочисленное деление со знаком
UDIVcc	011110	Целочисленное деление без знака и с модификацией icc
SDIVcc	011111	Целочисленное деление со знаком и с модификацией icc

Формат 3:

10	rd	op3	rs1	i = 0	unused (zero)	rs2
31 30	29 25	24 19	18 14	13	12 5	4 0

10	rd	op3	rs1	i = 1	simm13
31 30	29 25	24 19	18 14	13	12 0

Предлагаемый синтаксис на языке ассемблера:

udiv reg_{rs1}, *reg_or_imm*, reg_{rd}

sdiv reg_{rs1}, *reg_or_imm*, reg_{rd}

udivcc reg_{rs1}, *reg_or_imm*, reg_{rd}

sdivcc *reg*$_{rs1}$, *reg_or_imm*, *reg*$_{rd}$

Описание

При помощи инструкций деления выполняется деление 64 бит на 32 бита с получением 32-битового результата. Если бит i сброшен, при помощи данных инструкций рассчитывается «(Y & r[*rs1*]) ÷ r[*rs2*]». В противном случае (бит i установлен) при выполнении инструкций деления рассчитывается «(Y & r[*rs1*]) ÷ sign_ext(*simm13*)». В обоих случаях 32 бита целочисленного частного записываются в r[*rd*]. Остаток (если он есть) не учитывается.

При выполнении деления без знака (UDIV, UDIVcc) делимым является целочисленное двойное слово без знака (Y & r[*rs1*]), делителем является целочисленное слово (r[rs2]), в результате получается целочисленное частное слово без знака (r[*rd*]). При выполнении деления со знаком (SDIV, SDIVcc) делимым является целочисленное двойное слово со знаком (Y & r[*rs1*]), делителем является целочисленное слово со знаком (r[*rs2*] либо sign_ext(*simm13*)), в результате получается целочисленное частное слово со знаком (r[*rd*]).

При делении со знаком неточное частное округляется в направлении нуля, если получен ненулевой остаток. Например, при делении минус 3 на 2 результат равен минус 1 с остатком минус 1 (не минус 2 с остатком 1). Реализация может строго придерживаться этого округления, и в этом случае при отрицательном результате для обнаружения переполнения должен использоваться метод [А], приведенный ниже. Или в ней может быть сделано исключение округления с получением максимального отрицательного эквивалента, и в этом случае при отрицательном результате для обнаружения переполнения должен использоваться метод [Б].

Результат инструкции деления при определенных условиях может вызывать переполнение 32-битового регистра назначения r[*rd*]. В случае переполнения (вне зависимости от того, устанавливаются ли коды условий в icc или нет) максимальное соответствующее целое число возвращается, как частное в r[*rd*]. Условия переполнения и возвращаемое в r[*rd*] значение при данных условиях обозначены в таблице А.23.

Таблица А.23 – Условия детектирования переполнения и возвращаемое в r[rd] значение

Инструкция	Условия переполнения («результат» имеет частное и остаток)	Значение, возвращаемое в r[rd]
UDIV, UDIVcc	результат > (2^{32} – 1 с остатком (делитель – 1))	2^{32} – 1 (0xFFFFFFFF)
SDIV, SDIVcc (положительный результат)	результат > (2^{31} – 1 с остатком (|делитель| – 1))	2^{31} – 1 (0x7FFFFFFF)
SDIV, SDIVcc (отрицательный результат)	или метод [А]*: результат < (– 2^{31} с остатком – (|делитель| – 1)) или метод [Б]*: результат < (– 2^{31} с остатком 0)	– 2^{31} (0x80000000)
_____ * Использование одного из двух условий детектирования переполнения зависит от реализации и постоянно в рамках данной реализации.		

При выполнении UDIV и SDIV биты кодов условий не изменяются. При выполнении UDIVcc и SDIVcc биты кодов условий записываются согласно таблице А.24. Следует отметить, что отрицательное значение (N) и ноль (Z) устанавливаются в соответствии со значением частного (после того как было обозначено переполнение, если оно возникает), а также что при выполнении UDIVcc и SDIVcc переполнение (V) устанавливается по-разному.

Таблица А.24 – Определение флагов при выполнении операции деления

Бит icc	UDIVcc	SDIVcc
N	Устанавливается, если частное[31] = 1	Устанавливается, если частное[31] = 1
Z	Устанавливается, если частное[31:0] = 0	Устанавливается, если частное[31:0] = 0
V	Устанавливается при переполнении (согласно таблице А.23)	Устанавливается при переполнении (согласно таблице А.23)
C	Ноль	Ноль

Для обеспечения совместимости с последующими версиями программное обеспечение должно предполагать, что содержимое регистра Y не сохраняется инструкциями деления.

Замечание по реализации

При выполнении целочисленных инструкций деления может генерироваться остаток. В таком случае рекомендуется сохранять остаток в регистр Y.

Системные прерывания:
- division_by_zero.

Инструкции SAVE и RESTORE

Таблица А.25 – Инструкции SAVE и RESTORE

Код операции	op3	Операция
SAVE	111100	Сохранение окна вызывающей подпрограммы
RESTORE	111101	Восстановление окна вызывавшей подпрограммы

Формат 3:

10	rd	op3	rs1	i = 0	unused (zero)	rs2
31 30	29 25	24 19	18 14	13	12 5	4 0

10	rd	op3	rs1	i = 1	simm13
31 30	29 25	24 19	18 14	13	12 0

Предлагаемый синтаксис на языке ассемблера:
save *reg$_{rs1}$, reg_or_imm, reg$_{rd}$*
restore *reg$_{rs1}$, reg_or_imm, reg$_{rd}$*

Описание

Инструкция SAVE вычитает из CWP (по модулю NWINDOWS) единицу, полученное значение (new_CWP) сравнивается с содержимым регистра маски недействительности окна (WIM). Если бит WIM, соответствующий new_CWP, установлен, т.е. (WIM and 2^{new_CWP}) = 1, генерируется системное прерывание window_overflow. Если бит WIM, соответствующий new_CWP сброшен, системное прерывание window_overflow не генерируется, и new_CWP записывается в CWP. При этом текущим окном становится окно (CWP – 1), сохраняя окно вызывающей подпрограммы.

Инструкция RESTORE прибавляет к CWP (по модулю NWINDOWS) единицу, полученное значение (new_CWP) сравнивается с содержимым регистра маски недействительности окна (WIM). Если бит WIM, соответствующий new_CWP установлен, т.е. (WIM and 2^{new_CWP}) = 1, то генерируется системное прерывание window_underflow. Если бит WIM, соответствующий new_CWP сброшен, системное прерывание window_underflow не генерируется, и new_CWP записывается в CWP. При этом окно (CWP + 1) становится текущим окном, восстанавливая окно вызывавшей подпрограммы.

Более того, только в том случае, если системное прерывание переполнения или обратного переполнения не генерируется, инструкции SAVE и RESTORE работают аналогично обычным инструкциям ADD. Единственным отличием является то, что исходные компоненты операции r[*rs1*] и / или r[*rs2*] считываются из старого окна (т.е. окна, на которое указывает первоначальный CWP), а сумма записывается в r[*rd*] нового окна (т.е. окна, на которое указывает new_CWP).

Следует отметить, что арифметика CWP выполняется по модулю количества реализованных окон NWINDOWS.

Замечания по программированию

Инструкция SAVE может быть использована для автоматического выделения нового окна в регистровом файле и нового стекового кадра программного обеспечения в основной памяти.

Как правило, если при выполнении инструкции SAVE (RESTORE) срабатывает системное прерывание, обработчик системного прерывания при переполнении (обратном переполнении) возвращается к инструкции, в ходе выполнения которой сработало системное прерывание, для ее повторного выполнения. Поэтому, хотя операция ADD не выполняется в первый раз (когда срабатывает системное прерывание), она выполняется во второй раз.

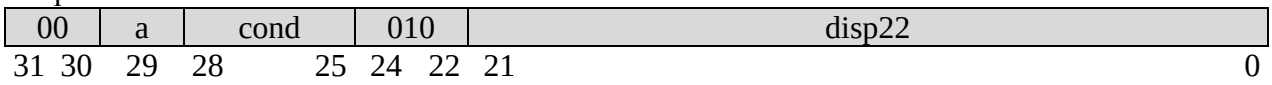
Системные прерывания:
- window_overflow (только SAVE);
- window_underflow (только RESTORE).

Инструкции ветвления по целочисленным кодам условий

Таблица А.26 – Инструкции ветвления по целочисленным кодам условий

Код операции	cond	Операция	Проверка icc
BA	1000	Всегда выполнять переход	1
BN	0000	Никогда не выполнять переход	0
BNE	1001	Переход, если не равно	not Z
BE	0001	Переход, если равно	Z
BG	1010	Переход, если больше	not (Z or (N xor V))
BLE	0010	Переход, если меньше или равно	Z or (N xor V)
BGE	1011	Переход, если больше или равно	not (N xor V)
BL	0011	Переход, если меньше	N xor V
BGU	1100	Для значений без знака, переход, если больше	not (C or Z)
BLEU	0100	Для значений без знака, переход, если меньше или равно	(C or Z)
BCC	1101	Переход при сбросе переноса (больше или равно, для значений без знака)	not C
BCS	0101	Переход при установке переноса (меньше, для значений без знака)	C
BPOS	1110	Переход, если значение положительное	not N
BNEG	0110	Переход, если значение отрицательное	N
BVC	1111	Переход, если переполнение не обнаружено	not V
BVS	0111	Переход, если обнаружено переполнение	V

Формат 2:



Предлагаемый синтаксис на языке ассемблера:

ba{**,a**}	*метка*	
bn{**,a**}	*метка*	
bne{**,a**}	*метка*	(эквивалентно: **bnz**)
be{**,a**}	*метка*	(эквивалентно: **bz**)
bg{**,a**}	*метка*	
ble{**,a**}	*метка*	
bge{**,a**}	*метка*	
bl{**,a**}	*метка*	
bgu{**,a**}	*метка*	
bleu{**,a**}	*метка*	
bcc{**,a**}	*метка*	(эквивалентно: **bgeu**)
bcs{**,a**}	*метка*	(эквивалентно: **blu**)
bpos{**,a**}	*метка*	
bneg{**,a**}	*метка*	
bvc{**,a**}	*метка*	
bvs{**,a**}	*метка*	

Примечание – Для установки «аннулирующего» бита инструкций Bicc, необходимо добавить «**,a**» к мнемоническому коду операции. Например, «**bgu,a** *метка*». Для того чтобы указать, что «**,a**» является опциональной возможностью, данный символ взят в фигурные скобки ({}).

Описание

Безусловные переходы (BA, BN)

Если аннулирующий бит сброшен, то инструкция BN (никогда не выполнять переход) выполняется как NOP. Если аннулирующий бит установлен, то следующая инструкция (инструкция задержки) аннулируется (не выполняется). В обоих случаях передача управления не выполняется.

BA (всегда выполнять переход) вызывает передачу управления относительно PC с задержкой по адресу «PC + (4 x sign_ext(*disp22*))», независимо от значений битов целочисленных кодов условий. Если аннулирующий бит инструкции ветвления установлен, то инструкция задержки аннулируется (не выполняется). Если аннулирующий бит сброшен, инструкция задержки выполняется.

Icc-условные переходы

Инструкции условных Bicc (все остальные, за исключением BA и BN) оценивают целочисленные коды условий (icc) в соответствии со значением поля cond инструкции. Такая оценка порождает «истинный» или «ложный» результат. Если результат «истинный», переход выполняется, т.е. инструкция вызывает передачу управления относительно PC с задержкой по адресу «PC + (4 x sign_ext(*disp22*))». Если результат «ложный», переход не выполняется.

Если выполняется условный переход, инструкция задержки всегда выполняется, независимо от значения аннулирующего бита. Если условный переход не выполняется и бит a (аннулирование) установлен, инструкция задержки аннулируется (не выполняется). Следует отметить, что аннулирующий бит воздействует на условные переходы иначе, чем на безусловные переходы.

Аннулирование, инструкции задержки и передача управления с задержкой подробнее описаны в пункте «6.2.3 Инструкции передачи управления (CTI)». Также следует отметить, что инструкции Bicc не следует помещать в слот задержки инструкции условного перехода.

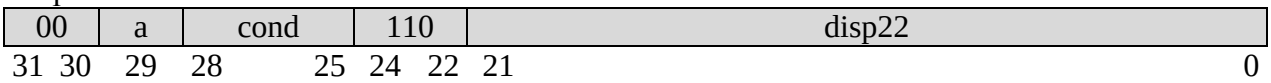
Системные прерывания: отсутствуют.

Инструкции ветвления по кодам условий операций с плавающей запятой

Таблица А.27 – Инструкции ветвления по кодам условий операций с плавающей запятой

Код операции	cond	Операция	Проверка fcc
FBA	1000	Всегда выполнять переход	1
FBN	0000	Никогда не выполнять переход	0
FBU	0111	Переход, если значения не упорядочены	U
FBG	0110	Переход, если больше	G
FBUG	0101	Переход, если больше или значения не упорядочены	G or U
FBL	0100	Переход, если меньше	L
FBUL	0011	Переход, если меньше или значения не упорядочены	L or U
FBLG	0010	Переход, если меньше или больше	L or G
FBNE	0001	Переход, если не равно	L or G, or U
FBE	1001	Переход, если равно	E
FBUE	1010	Переход, если равно или значения не упорядочены	E or U
FBGE	1011	Переход, если больше или равно	E or G
FBUGE	1100	Переход, если больше, меньше или значения не упорядочены	E or G or U
FBLE	1101	Переход, если меньше или равно	E or L
FBULE	1110	Переход, если меньше, равно или значения не упорядочены	E or L or U
FBO	1111	Переход, если значения упорядочены	E or L or G

Формат 2:



Предлагаемый синтаксис на языке ассемблера:

fba{**,a**} *метка*
fbn{**,a**} *метка*
fbu{**,a**} *метка*
fbg{**,a**} *метка*
fbug{**,a**} *метка*
fbl{**,a**} *метка*
fbul{**,a**} *метка*
fblg{**,a**} *метка*
fbne{**,a**} *метка* (эквивалентно: **fbnz**)
fbe{**,a**} *метка* (эквивалентно: **fbz**)
fbue{**,a**} *метка*
fbge{**,a**} *метка*
fbuge{**,a**} *метка*
fble{**,a**} *метка*
fbule{**,a**} *метка*
fbo{**,a**} *метка*

Примечание – Для установки «аннулирующего» бита инструкций FBfcc, необходимо добавить «**,a**» к мнемоническому коду операции. Например, «**fbl,a** *метка*». Для того чтобы указать, что «**,a**» является опциональной возможностью, данный символ взят в фигурные скобки ({}).

Описание

Безусловные переходы (FBA, FBN)

Если аннулирующий бит сброшен, инструкция FBN (никогда не выполнять переход) выполняется как NOP. Если аннулирующий бит установлен, то следующая инструкция (инструкция задержки) аннулируется (не выполняется). В обоих случаях передача управления не выполняется.

FBA (всегда осуществлять переход) вызывает передачу управления относительно PC с задержкой по адресу «PC + (4 x sign_ext(*disp22*))», независимо от значения битов кодов условий операций с плавающей запятой. Если аннулирующий бит инструкции ветвления установлен, то инструкция задержки аннулируется (не выполняется). Если аннулирующий бит сброшен, то инструкция задержки выполняется.

Fcc-условные переходы

Инструкции условных FBfcc (все остальные, за исключением FBA и FBN) оценивают коды условий операций с плавающей запятой (fcc) в соответствии со значением поля cond инструкции. Такая оценка порождает «истинный» или «ложный» результат. Если результат «истинный», переход выполняется, т.е. инструкция вызывает передачу управления относительно PC с задержкой по адресу «PC + (4 x sign_ext(*disp22*))». Если результат «ложный», переход не выполняется.

Если выполняется условный переход, инструкция задержки всегда выполняется, независимо от значения аннулирующего поля. Если условный переход не выполняется и бит a (аннулирование) установлен, инструкция задержки аннулируется (не выполняется). Следует отметить, что аннулирующий бит воздействует на условные переходы иначе, чем на безусловные переходы.

Аннулирование, инструкции задержки и передача управления с задержкой подробнее описаны в пункте «6.2.3 Инструкции передачи управления (CTI)». Также следует отметить, что инструкции FBfcc не следует помещать в слот задержки инструкции условного перехода.

Если в PSR бит EF сброшен, либо если FPU отсутствует, инструкция FBfcc не вызывает перехода, следующая инструкция не аннулируется, и срабатывает системное прерывание fp_disabled.

Если непосредственно перед FBfcc выполняется инструкция FPop2, то результат FBfcc будет неопределенным. Поэтому между FPop2 и последующей инструкцией FBfcc должна выполняться хотя бы одна не-FPop2 инструкция.

Если одна из этих трех инструкций, выполняемых после (по времени) LDFSR, является инструкцией FBfcc, значение поля fcc в FSR, видимое для FBfcc, не определено.

Системные прерывания:

- fp_disabled;
- fp_exception.

Инструкции ветвления по кодам условий сопроцессора

Таблица А.28 – Инструкции ветвления по кодам условий сопроцессора

Код операции	cond	Проверка bp_CP_cc[1:0]
CBA	1000	Всегда выполнять переход
CBN	0000	Никогда не выполнять переход
CB3	0111	3
CB2	0110	2
CB23	0101	2 или 3
CB1	0100	1
CB13	0011	1 или 3
CB12	0010	1 или 2
CB123	0001	1 или 2, или 3

Окончание таблицы А.28

Код операции	cond	Проверка bp_CP_cc[1:0]
CB0	1001	0
CB03	1010	0 или 3
CB02	1011	0 или 2
CB023	1100	0 или 2 или 3
CB01	1101	0 или 1
CB013	1110	0 или 1 или 3
CB012	1111	0 или 1 или 2

Формат 2:

00	a	cond	111	disp22
31 30	29	28 25	24 22	21 0

Предлагаемый синтаксис на языке ассемблера:

cba{**,a**} *метка*
cbn{**,a**} *метка*
cb3{**,a**} *метка*
cb2{**,a**} *метка*
cb23{**,a**} *метка*
cb1{**,a**} *метка*
cb13{**,a**} *метка*
cb12{**,a**} *метка*
cb123{**,a**} *метка*
cb0{**,a**} *метка*
cb03{**,a**} *метка*
cb02{**,a**} *метка*
cb023{**,a**} *метка*
cb01{**,a**} *метка*
cb013{**,a**} *метка*
cb012{**,a**} *метка*

Примечание – Для установки «аннулирующего» бита инструкций CBccc, необходимо добавить «**,a**» к мнемоническому коду операции. Например, «**cbl2,a** *метка*». Для того чтобы указать, что «**,a**» является опциональной возможностью, данный символ взят в фигурные скобки ({}).

Описание

Безусловные переходы (CBA, CBN)

Если аннулирующий бит сброшен, инструкция CBN (никогда не выполнять переход) выполняется как NOP. Если аннулирующий бит установлен, следующая инструкция (инструкция задержки) аннулируется (не выполняется). В обоих случаях передача управления не выполняется.

CBA (всегда выполнять переход) вызывает передачу управления относительно PC с задержкой по адресу «PC + (4 x sign_ext(*disp22*))», независимо от значения битов кодов условий сопроцессора. Если аннулирующий бит инструкции ветвления установлен, то инструкция задержки аннулируется (не выполняется). Если аннулирующий бит сброшен, инструкция задержки выполняется.

Ccc-условные переходы

Инструкций условных CBccc (все остальные, за исключением CBA и CBN) оценивают коды условий сопроцессора (ccc) в соответствии со значением поля cond инструкции. Такая оценка порождает «истинный» или «ложный» результат. Если результат «истинный», переход выполняется, т.е. инструкция вызывает передачу

управления относительно PC с задержкой по адресу «PC + (4 x sign_ext(*disp22*))». Если результат «ложный», переход не выполняется.

Если выполняется условный переход, всегда выполняется инструкция задержки, независимо от значения аннулирующего поля. Если условный переход не выполняется и бит a (аннулирование) установлен, инструкция задержки аннулируется (не выполняется). Следует отметить, что аннулирующий бит воздействует на условные переходы иначе, чем на безусловные переходы.

Аннулирование, инструкции задержки и передача управления с задержкой подробнее описаны в пункте «6.2.3 Инструкции передачи управления (CTI)». Также следует отметить, что инструкции CBccc не следует помещать в слот задержки инструкции условного перехода.

Если в PSR бит EC сброшен, либо если сопроцессор отсутствует, инструкция CBccc не вызывает перехода, следующая инструкция не аннулируется, и срабатывает системное прерывание cp_disabled.

Системные прерывания:
- cp_disabled;
- cp_exception.

Инструкция вызова подпрограммы и создания связи

Таблица А.29 – Инструкция вызова подпрограммы и создания связи

Код операции	op	Операция
CALL	01	Вызов подпрограммы и создание связи

Формат 1:

01	disp30
31 30	29 0

Предлагаемый синтаксис на языке ассемблера:
call *метка*

Описание

Инструкции CALL вызывает безусловную передачу управления относительно PC с задержкой по адресу «PC + (4 x *disp30*)». Поскольку поле слова замены (disp30) вмещает 30 бит, адрес назначения может находиться на произвольном удалении. Замена PC выполняется с помощью добавления двух нулевых младших значащих разрядов к 30-ти битовому полю слова замены в инструкции.

При выполнении инструкции CALL также в регистр r[15] (выходной регистр 7) записывается значение PC, содержащее адрес CALL.

Системные прерывания: (отсутствуют).

Инструкция перехода и создания связи

Таблица А.30 – Инструкция перехода и создания связи

Код операции	op3	Операция
JMPL	111000	Переход и создание связи

Формат 3:

10	rd	op3	rs1	i = 0	unused (zero)	rs2
31 30	29 25	24 19	18 14	13	12 5	4 0
10	rd	op3	rs1	i = 1	simm13	
31 30	29 25	24 19	18 14	13	12 0	

Предлагаемый синтаксис на языке ассемблера:
jmpl *адрес*, reg_{rd}

Описание

Инструкция JMPL выполняет передачу управления с задержкой по адресу, косвенно вычисляемому из значения(-й) регистра(-ов). Итоговый адрес равняется «r[*rs1*] + r[*rs2*]», если бит i сброшен, или «r[*rs1*] + sign_ext(*simm13*)», если бит i установлен.

Инструкция JMPL копирует PC, содержащий адрес самой инструкции JMPL, в r[*rd*].

Если хотя бы один из двух младших разрядов адреса перехода установлен, то срабатывает системное прерывание mem_address_not_aligned.

Замечания по программированию

Инструкция JMPL с rd = 15 выполняется как команда вызова с косвенным расчетом адреса, исходя из значения(-й) регистра(-ов), которая использует стандартный регистр связи.

Инструкция JMPL с rd = 0 может использоваться для возврата из подпрограммы. Стандартный адрес возврата – «r[31] + 8», если подпрограмма, вызываемая инструкцией CALL, является не-листовой (используется инструкция SAVE), или «r[15] + 8», если вызываемая инструкцией CALL подпрограмма является листовой (без использования SAVE).

Замечание по реализации

Когда инструкция RETT находится в слоте задержки JMPL, адрес назначения JMPL в обязательном порядке извлекается из адресного пространства, обусловленного новым значением (т.е. после RETT) бита S в PSR. В частности, это относится к возврату из системного прерывания в адресное пространство пользователя.

Системные прерывания:
- mem_address_not_aligned.

Инструкция возврата из системного прерывания

Таблица А.31 – Инструкция возврата из системного прерывания

Код операции	op3	Операция
RETT*	111001	Возврат из системного прерывания
_____ * Привилегированная инструкция.		

Формат 3:

10	unused (zero)	op3	rs1	i = 0	unused (zero)	rs2
31 30	29 25	24 19	18 14	13	12 5	4 0

10	unused (zero)	op3	rs1	i = 1	simm13
31 30	29 25	24 19	18 14	13	12 0

Предлагаемый синтаксис на языке ассемблера:
rett *адрес*

Описание

RETT используется для возврата из обработчика системных прерываний. При определенных условиях RETT может сама вызывать системное прерывание. Если

инструкция RETT не вызывает системное прерывание, то к CWP добавляется 1 (по модулю NWINDOWS), выполняется передача управления с задержкой по адресу назначения, из бита PS регистра PSR восстанавливается значения бита S, устанавливается бит ET в PSR. Адрес назначения – «r[*rs1*] + r[*rs2*]», если бит i сброшен, или «r[*rs1*] + sign_ext(*simm13*)», если бит i установлен.

При выполнении RETT может произойти одно из приведенных ниже системных прерываний. Описание приводится в порядке приоритетности (сначала системные прерывания высокого приоритета):

- если системные прерывания разрешены (ET = 1) и процессор находится в режиме пользователя (S=0), срабатывает системное прерывание privileged_instruction;
- если системные прерывания разрешены (ET = 1) и процессор находится в режиме супервизора (S=1), срабатывает системное прерывание illegal_instruction;
- если системные прерывания не разрешены (ET = 0) и процессор находится в режиме пользователя (S = 0), или обнаружено выполнение условия window_underflow (WIM and 2^{new_CWP}) = 1, или один из двух младших разрядов адреса назначения установлен, тогда процессор указывает о возникновении одного из следующих системных прерываний: privileged_instruction, или window_underflow, или mem_address_not_aligned (соответственно) в поле tt регистра TBR, и входит в режим error_mode.

Инструкция, выполняемая непосредственно перед RETT, должна быть инструкцией JMPL. В противном случае, один или более доступов инструкций, которые следуют за RETT, могут быть выполнены в некорректное адресное пространство.

Замечания по программированию

Для повторного выполнения инструкции, на которую пришлось системное прерывание, при возврате из обработчика системного прерывания необходимо использовать следующую последовательность:

```
jmpl   %r17, %r0   ! old PC
rett   %r18        ! old nPC
```

Для возврата к инструкции, которая выполняется после инструкции, на которую пришлось системное прерывание (например, после эмулирования инструкции), необходимо использовать следующую последовательность:

```
jmpl   %r18, %r0   ! old nPC
rett   %r18 + 4    ! old nPC + 4
```

Системные прерывания:
- illegal_instruction;
- privileged_instruction;
- privileged_instruction (может вызвать вход процессора в режим error_mode);
- mem_address_not_aligned (может вызвать вход процессора в режим error_mode);
- window_underflow (может вызвать вход процессора в режим error_mode).

Инструкции системного прерывания по целочисленным кодам условий

Таблица А.32 – Инструкции системного прерывания по целочисленным кодам условий

Код операции	cond	Операция	Проверка icc
TA	1000	Системное прерывание происходит всегда	1
TN	0000	Системное прерывание никогда не происходит	0
TNE	1001	Системное прерывание, если не равно	not Z
TE	0001	Системное прерывание, если равно	Z

Окончание таблицы А.32

Код операции	cond	Операция	Проверка icc
TG	1010	Системное прерывание, если больше	not (Z or (N xor V))
TLE	0010	Системное прерывание, если меньше или равно	Z or (N xor V)
TGE	1011	Системное прерывание, если больше или равно	not (N xor V)
TL	0011	Системное прерывание, если меньше	N xor V
TGU	1100	Системное прерывание, если больше, для значении без знака	not (C or Z)
TLEU	0100	Системное прерывание, если меньше или равно, для значении без знака	(C or Z)
TCC	1101	Системное прерывание, если бит переноса сброшен (больше или равно, для значений без знака)	not C
TCS	0101	Системное прерывание, если бит переноса установлен (меньше, для значений без знака)	C
TPOS	1110	Системное прерывание, если значение положительное	not N
TNEG	0110	Системное прерывание, если значение отрицательное	N
TVC	1111	Системное прерывание, если бит переполнения сброшен	not V
TVS	0111	Системное прерывание, если бит переполнения установлен	V

Формат 3:

10	reserved	cond	111010	rs1	i = 0	reserved	rs2
31 30	29	28 25	24 19	18 14	13	12 5	4 0

10	reserved	cond	111010	rs1	i = 1	reserved	imm7
31 30	29	28 25	24 19	18 14	13	12 7	6 0

Предлагаемый синтаксис на языке ассемблера:

ta *software_trap#*
tn *software_trap#*
tne *software_trap#* (эквивалентно: **tnz**)
te *software_trap#* (эквивалентно: **tz**)
tg *software_trap#*
tle *software_trap#*
tge *software_trap#*
tl *software_trap#*
tgu *software_trap#*
tleu *software_trap#*
tcc *software_trap#* (эквивалентно: **tgeu**)
tcs *software_trap#* (эквивалентно: **tlu**)
tpos *software_trap#*
tneg *software_trap#*
tvc *software_trap#*
tvs *software_trap#*

Описание

Инструкция Ticc проводит оценку целочисленных кодов условий (icc) в соответствии со значением поля cond инструкции, получая либо «истинный» либо «ложный» результат. Если результат «истинный» и более высокоприоритетные исключения или запросы на прерывания не ожидают обработки, то генерируется системное прерывание trap_instruction. Если результат «ложный», системное прерывание trap_instruction не происходит и инструкция выполняется аналогично NOP.

Если системное прерывание trap_instruction генерируется, то в поле tt базового регистра системных прерываний (TBR) записывается 128 плюс семь младших значащих битов «r[*rs1*] + r[*rs2*]», если бит i сброшен, или 128 плюс семь младших значащих битов «r[*rs1*] + sign_ext(*software_trap#*)», если бит i установлен.

При выполнении Ticc процессор входит в режим супервизора, системные прерывания запрещаются, CWP уменьшается (по модулю NWINDOWS), PC и nPC сохраняются в r[17] и r[18] (локальные регистры 1 и 2) нового окна.

Замечания по программированию

Ticc можно использовать для реализации точки останова, трассировки и вызовов программного обеспечения супервизора. Данные инструкции также могут использоваться для динамической проверки, например, выхода индекса за пределы массива, целочисленного переполнения и т.д.

Системные прерывания:
- trap_instruction.

Инструкции считывания регистров состояния

Таблица А.33 – Инструкции считывания регистров состояния

Код операции	op3	rs1	Операция
RDY	101000	0	Считывание регистра Y
RDASR**	101000	1–15	Считывание вспомогательного регистра состояния (зарезервировано)
RDASR**	101000	16–31	Считывание вспомогательного регистра состояния (в зависимости от реализации)
RDPSR*	101001	зарезервировано	Считывание регистра состояния процессора
RDWIM*	101010	зарезервировано	Считывание регистра маски недопустимости окна
RDTBR*	101011	зарезервировано	Считывание базового регистра системного прерывания

_____
* Привилегированная инструкция.
** Привилегированная инструкция, если исходный регистр привилегирован.

Формат 3:

10	rd	op3	rs1	unused (zero)	unused (zero)
31 30	29 25	24 19	18 14	13	12 0

Предлагаемый синтаксис на языке ассемблера:

rd **%y**, *reg*$_{rd}$
rd *asr_reg*$_{rs1}$, *reg*$_{rd}$
rd **%psr**, *reg*$_{rd}$
rd **%wim**, *reg*$_{rd}$
rd **%tbr**, *reg*$_{rd}$

Описание

Данные инструкций осуществляют считывание определенного регистра состояния IU в r[*rd*].

Следует отметить, что RDY отличается от RDASR только значением поля rs1. Для считывания регистра Y поле rs1 должно обнулено, а op3 = 0x28.

Если rs1 не равно 0 и op3 = 0x28, выполняется считывание зависящего от конкретной реализации вспомогательного регистра состояния. Значения rs1 в диапазоне

от 1 до 14 зарезервированы для последующих версий архитектуры; значения 16–31 доступны для использования в конкретных реализациях. Инструкция RDASR при rs1 = 15 и rd = 0 определена как инструкция STBAR. Инструкции RDASR при rs1 = 15 и значении поля rd не равном 0 зарезервированы для последующих версий архитектуры.

Если значение rs1 инструкции RDASR находится в диапазоне от 1 до 14, её итоговый результат не определен, при этом системное прерывание illegal_instruction не генерируется.

Если значение rs1 инструкции RDASR находится в диапазоне 16–31, интерпретация битов с 13-го по 0-й и с 29-го по 25-й инструкции зависит от реализации, независимо от того, является данная инструкция привилегированной или нет, и вызывает ли данная инструкция системное прерывание illegal_instruction или нет.

Замечание по реализации

Вспомогательные регистры состояния могут включать в себя (например) таймеры, счетчики, регистры диагностики, самодиагностики и управления прерываниями.

Системные прерывания:
- privileged_instruction (за исключением RDY);
- illegal_instruction (только RDASR; зависит от реализации).

Инструкции записи регистров состояния

Таблица А.34 – Инструкции записи регистра статуса

Код операции	op3	rd	Операция
WRY	110000	0	Запись регистра Y
WRASR**	110000	1–15	Запись вспомогательного регистра состояния (зарезервировано)
WRASR**	110000	16–31	Запись вспомогательного регистра состояния (в зависимости от реализации)
WRPSR*	110001	зарезервировано	Запись регистра состояния процессора
WRWIM*	110010	зарезервировано	Запись регистра маски недопустимости окна
WRTBR*	110011	зарезервировано	Запись базового регистра системных прерываний

_____

* Привилегированная инструкция.

** Привилегированная инструкция, если регистр назначения привилегирован.

Формат 3:

10	rd	op3	rs1	i = 0	unused (zero)	rs2
31 30	29 25	24 19	18 14	13	12 5	4 0

10	rd	op3	rs1	i = 1	simm13
31 30	29 25	24 19	18 14	13	12 0

Предлагаемый синтаксис на языке ассемблера:

wr *reg*$_{rs1}$, *reg_or_imm*, **%y**

wr *reg*$_{rs1}$, *reg_or_imm*, *asr_ reg*$_{rd}$

wr *reg*$_{rs1}$, *reg_or_imm*, **%psr**

wr *reg*$_{rs1}$, *reg_or_imm*, **%wim**

wr *reg*$_{rs1}$, *reg_or_imm*, **%tbr**

Описание

При помощи WRY, WRPSR, WRWIM и WRTBR выполняется запись «r[*rs1*] xor r[*rs2*]», если бит i сброшен, или «r[*rs1*] xor sign_ext(*simm13*)», если бит i установлен, в поля для записи определенного регистра состояния IU.

Следует отметить, что WRY отличается от WRASR только значением поля rd. Для записи в регистр Y поле rd должно быть обнулено, а op3 = 0x30.

Инструкция WRASR выполняет запись значения во вспомогательный регистр состояния (ASR), указанный в rd. Операция, при помощи которой генерируется записываемое значение, может зависеть от rd или от реализации (см. ниже). Инструкция WRASR выполняется при rd не равном 0 и op3 = 0x30.

Инструкции WRASR со значениями rd, лежащими в диапазоне от 1 до 15, зарезервированы для последующих версий архитектуры, результат их выполнения не определен.

Инструкции WRASR со значениями rd, лежащими в диапазоне от 16 до 31, доступны для использования в конкретных реализациях. Для таких инструкций следующие особенности являются зависимыми от реализации: интерпретация битов с 18-го по 0-й, выполнение операции(-ий) (например, операции исключающего ИЛИ) для генерации записываемого в ASR значения, является ли данная инструкция привилегированной или нет, вызывает ли данная инструкция системное прерывание illegal_instruction или нет. В некоторых существующих реализациях, при помощи инструкций WRASR можно осуществлять запись в регистр Y. В новых реализациях инструкция WRASR не должна использоваться для записи регистра Y.

Если результат выполнения инструкции WRPSR приводит к тому, что поле CWP в PSR начинает указывать на недопустимое окно, происходит системное прерывание illegal_instruction, запись в PSR не выполняется.

Инструкции записи регистра состояния являются инструкциями записи с задержкой. То есть, завершение операции записи может длиться до окончания выполнения третьей инструкции, идущей после инструкции записи. Количество инструкций задержки (от 0 до 3) зависит от реализации.

Инструкция WRPSR выполняет запись полей ET и PIL немедленно по отношению к прерываниям.

В описании ниже приведена взаимосвязь между записью поля регистра состояния и выполнением одновременного или последовательного доступа к данному полю:

1) Если одна из трех инструкций, следующих после инструкции записи регистра состояния, записывает какое-либо поле того же регистра состояния, то последующее содержимое данного поля не определено. Исключением из данного правила будет являться случай, когда экземпляр той же самой инструкции записи регистра состояния (например, инструкция WRPSR, располагающаяся в пределах трех инструкций от другой WRPSR) будет записывать поле, как предполагалось.

Замечание по программированию

Многие инструкции в неявной форме перезаписывают CWP или поле icc в PSR. Например, SAVE, RESTORE, системные прерывания и RETT перезаписывают CWP, многие инструкции осуществляют запись icc.

2) Если одна из трех инструкций, следующих после инструкции записи регистра состояния, считывает какое-либо поле того же регистра состояния, которое было изменено первоначальной инструкцией записи регистра, то значение поля, считываемое данной инструкцией, не определено.

Замечания по программированию

Многие инструкции в неявной форме считывают CWP или icc. Например, CALL выполняет неявное считывание CWP, инструкции, которые ссылаются на целочисленные не глобальные (принадлежащие какому-либо окну) регистры, также неявно считывают CWP; SAVE, RESTORE, системные прерывания (включая Ticc) и RETT считывают CWP, Bicc и Ticc считывают поле icc.

SAVE, RESTORE и RETT в неявной форме считывают WIM. Если любая из них выполняется в пределах трех инструкций после WRWIM, которая изменяет содержимое WIM, то предсказать, возникнет ли одно из системных прерываний window_overflow или window_underflow, не представляется возможным.

MULScc, RDY, SDIV, SDIVcc, UDIV и UDIVcc в неявной форме считывают регистр Y. Если какая-либо из данных инструкций выполняется в пределах трех инструкций после WRY, которая изменяет содержимое регистра Y, то результат выполнения инструкции не определен.

3) В некоторых реализациях, если инструкция WRPSR обновляет значение поля PIL регистра PSR и одновременно устанавливает бит ET, то может произойти прерывающее системное прерывание с уровнем, соответствующим старому значению PIL.

Замечания по программированию

При разрешении системных прерываний и изменении значения PIL следует использовать пару инструкций WRPSR. Первая WRPSR устанавливает ET = 0 и новое значение PIL; вторая WRPSR выставляет ET = 1 и новое значение PIL.

Если системные прерывания разрешены (ET = 1), необходимо соблюдать осторожность в случае, если программное обеспечение собирается их запретить (ET = 0). Поскольку последовательность «RDPSR, WRPSR» может выполняться с прерываниями – регистр PSR может быть изменен между выполнением данных двух инструкций – такая последовательность не является надежным механизмом запрещения системных прерываний. Существует две альтернативы:

а. генерация системного прерывания Ticc, обработчик которого запретит системные прерывания. Перед тем как возвратиться из прерывания в режим супервизора обработчик системного прерывания должен проверить, действительно ли он был «вызван» из режима супервизора (через контроль бита PS в PSR);

б. использовать последовательность «RDPSR, WRPSR», но тогда все обработчики системных прерываний и внешних запросов на прерывания должны быть составлены так, чтобы перед возвратом в режим супервизора они восстанавливали значение PSR на момент входа в обработчик прерывания.

4) Если выполнение одной из трех инструкций, которые следуют за WRPSR, привело к системному прерыванию, то значения полей S и CWP, считанные из PSR при обработке системного прерывания, могут быть как прежними, так и уже обновленными.

5) Если выполнение одной из трех инструкций, которые следуют за WRTBR, привело к системному прерыванию, то используемый базовый адрес системных прерываний (TBA) может иметь как прежнее, так и новое значение.

6) Если при выполнении одной из трех инструкций, следующих за любой инструкцией записи регистра состояния, происходит системное прерывание, при последующем считывании регистра состояния обработчик системных прерываний получит новое значение регистра состояния.

Замечания по реализации

Вспомогательные регистры состояния могут включать в себя (например) таймеры, счетчики, регистры диагностики, самодиагностики и управления прерываниями.

Два возможных способа реализации инструкцией WRPSR немедленной записи ET и PIL по отношению к прерываниям:

- немедленная запись ET и PIL (с распространением по конвейеру по мере необходимости);
- запрещение прерываний в течение последующих трех инструкций.

Системные прерывания:

- privileged_instruction (за исключением WRY);
- illegal_instruction (WRPSR, если CWP ≥ NWINDOWS);

- illegal_instruction (WRASR в зависимости от реализации).

Инструкция STBAR

Таблица А.35 – Инструкция STBAR

Код операции	op3	Операция
STBAR	101000	Барьер сохранений

Формат 3:

10	00000	op3	01111	0	unused (zero)
31 30	29 25	24 19	18 14	13	12 0

Предлагаемый синтаксис на языке ассемблера:
stbar

Описание

Инструкция барьера сохранений (STBAR) принуждает завершить все операции сохранения и атомарной загрузки-сохранения, выданные процессору перед STBAR, перед тем как совершить любую из операций сохранения и атомарной загрузки-сохранения, выданной процессору после STBAR, в памяти.

STBAR выполняется как NOP в системах, в которых реализуется только модель «Строгой консистентности памяти» или модель «Полного упорядочивания сохранения» (TSO), а также в системах, которые реализуют модель «Частичного упорядочивания сохранения» (PSO), но запущены с отключенным режимом PSO.

Следует отметить, что кодирование STBAR соответствует кодированию инструкции RDASR с rs1 = 15 и rd = 0.

Замечание по реализации

Для обеспечения корректной работы процессору достаточно прекратить подачу новых операций сохранения и атомарной загрузки-сохранения при обнаружении STBAR и продолжить их выдачу только после того, как все операции сохранения завершены и наблюдаются в памяти всеми процессорами. Более эффективные реализации используют тот факт, что процессор может выдавать операции сохранения и атомарной загрузки-сохранения после STBAR, при условии, что данные операции гарантированно не будут выполняться памятью до завершения предыдущих операций сохранения и атомарной загрузки-сохранения.

Системные прерывания: отсутствуют.

Нереализованная инструкция

Таблица А.36 – Нереализованная инструкция

Код операции	op	op2	Операция
UNIMP	00	000	Операция не реализована

Формат 2:

00	reserved	000	const22
31 30	29 25	24 22	21 0

Предлагаемый синтаксис на языке ассемблера:
unimp *const22*

Описание

Инструкция UNIMP вызывает системное прерывание illegal_instruction. Значение const22 не учитывается аппаратным обеспечением; в частности, значения const22 не зарезервированы архитектурой для использования в последующих версиях.

Замечания по программированию

Данная инструкция может использоваться как часть протокола вызова функции, которая должна возвращать составное значение, например, такое как **struct** или **union** языка С или как **record** языка Паскаль.

1) Инструкцию UNIMP располагают после слота задержки (не в самом слоте) инструкции CALL вызова функции.

2) Если вызванная функция возвращает структуру, она может найти размер структуры, который ожидается вызывающей программой в качестве результата, в поле const22 инструкции UNIMP. Вызываемая функция может проверить код операции, чтобы убедиться, что это действительно UNIMP.

3) Если в функции не планировалось возвращать структуру, то при возврате будет осуществлена попытка выполнения инструкции UNIMP, а не её пропуск, как это подразумевалось. Это вызывает завершение работы программы. Такое поведение добавляет некоторую проверку во время выполнения программы интерфейсам, которые нельзя полноценно проверить во время компиляции.

Системные прерывания:
- illegal_instruction.

Инструкция сброса памяти инструкций

Таблица А.37 – Инструкция сброса памяти инструкций

Код операции	op3	Операция
FLUSH	111011	Сброс памяти инструкций

Формат 3:

10	unused (zero)	op3	rs1	i = 0	unused (zero)	rs2
31 30	29 25	24 19	18 14	13	12 5	4 0

10	unused (zero)	op3	rs1	i = 1	simm13
31 30	29 25	24 19	18 14	13	12 0

Предлагаемый синтаксис на языке ассемблера:
flush *адрес*

Описание

Инструкция FLUSH гарантирует, что выборка последующих инструкций в процессор, осуществляющий FLUSH, будет произведена после всех загрузок, сохранений и атомарных загрузок-сохранений данным процессором до FLUSH.

В мультипроцессорных системах FLUSH также гарантирует, что операции сохранения и атомарной загрузки-сохранения адрес назначения инструкции FLUSH, выполненные до FLUSH данным процессором, станут видимыми для выборки инструкций всех остальных процессоров через некоторое время после выполнения FLUSH.

При выполнении процессором последовательности сохранений и атомарных загрузок-сохранений наряду с соответствующими инструкциями FLUSH и STBAR, (последняя из них требуется только для модели памяти PSO), изменения в выборке инструкций всех процессоров будут проявляться в порядке их выполнения.

FLUSH действует на двойное слово, которое содержится по заданному адресу.

FLUSH требуется выполнять исключительно между сохранением и последующим доступом к инструкции в измененной ячейке. Модель памяти сама по себе гарантирует, что результат операции загрузки данных будет соответствовать последнему произведенному сохранению, даже если промежуточные FLUSH не выполняются.

Операндом действительного виртуального адреса инструкции FLUSH является «r[*rs1*] + r[*rs2*]», если бит i сброшен, или «r[*rs1*] + sign_ext(*simm13*)», если бит i установлен. Два младших значащих адресных бита результата не используются и должны быть сброшены программным обеспечением. Бит 2 адреса не учитывается.

К моменту выполнения пяти инструкций, следующих после FLUSH, любая внутренняя копия адресуемой ячейки в процессоре, выдающем команды, будет содержать такое же значение, которое было бы при считывании из памяти. В качестве примера можно привести конвейер процессора или буфер инструкций, которые могут содержать внутреннюю копию адресуемой ячейки.

Замечания по программированию

1) FLUSH обычно используется в самомодифицирующемся коде.

2) Хотя FLUSH обеспечивает поддержку самомодифицирующегося кода, использование подобного кода не рекомендуется. В некоторых реализациях выполнение FLUSH является затратным по времени.

Замечания по реализации

1) FLUSH может работать не только с двойным словом, но и с большими блоками информации, обозначенными в действительном адресе. В частности, при помощи данной инструкции может сбрасываться одна или более строк или блоков, содержащих кэш.

2) В однопроцессорной системе с совмещенным кэшем инструкций и данных (или без кэша) и общей глубиной конвейера (буфер сохранения плюс буфер инструкций) не более пяти инструкций, FLUSH может не выполнять никаких операций.

В однопроцессорной системе с раздельными кэшами инструкций и данных FLUSH гарантирует, что если оба кэша содержат копию адресуемой позиции, то эти кэшированные копии в конечном итоге станут согласованными.

В мультипроцессорной системе с кэшами FLUSH гарантирует согласованность всех копий кэшируемого содержимого адресуемой ячейки.

Согласованность кэша может быть реализована через любую комбинацию операций признания недействительности, обратной записи кэшируемых данных или другими зависящими от реализации механизмами согласования.

3) Если инструкция FLUSH не реализована аппаратно, как описано выше, FLUSH вызывает системное прерывание unimplemented_FLUSH (или illegal_instruction), при этом функции FLUSH выполняются программным обеспечением системы. Будет ли происходить системное прерывание при выполнении FLUSH или нет, зависит от реализации. Если она вызывает системное прерывание, то при выполнении FLUSH срабатывает системное прерывание unimplemented_FLUSH или illegal_instruction. В реализациях, в которых unimplemented_FLUSH поддерживает более быстрое эмулирование FLUSH программным обеспечением по сравнению с illegal_instruction, рекомендуется использовать unimplemented_FLUSH. В реализации может быть организован выбор предпочтительного системного прерывания при выполнении инструкции FLUSH, который может осуществляться на базе отдельного вывода или бита в зависящем от конкретной реализации регистре управления.

4) В заданной реализации при помощи FLUSH может сбрасываться буфер инструкций и / или конвейер процессора при условии, что буфер инструкций и конвейер будут согласованы с кэшем за время выполнения пяти инструкций.

5) Количество инструкций, которые должны быть выполнены после FLUSH, перед тем как её эффект будет применен, зависит от реализации, но не может быть более пяти.

Системные прерывания:
- unimplemented_FLUSH (зависит от реализации);
- illegal_instruction (зависит от реализации).

Инструкции операций с плавающей запятой (FPop)

Таблица А.38 – Инструкции операций с плавающей запятой (FPop)

Код операции	op3	Операция
FPop1	110100	Операции с плавающей запятой
FPop2	110101	Операции с плавающей запятой

Формат 3:

10	rd	110100	rs1	opf	rs2
31 30	29 25	24 19	18 14	13 5	4 0

10	rd	110101	rs1	opf	rs2
31 30	29 25	24 19	18 14	13 5	4 0

Описание

Кодирование инструкций операций с плавающей запятой (FPop) выполняется при помощи двух форматов типа 3: FPop1 и FPop2. Конкретная операция с плавающей запятой указывается в поле opf. Следует отметить, что инструкции загрузки / сохранения данных с плавающей запятой не являются инструкциями FPop.

Инструкции FPop1 не меняют коды условий операций с плавающей запятой. Инструкции FPop2 меняют коды условий операций с плавающей запятой.

Инструкции FPop поддерживают операции с целочисленными словами и операндами с одинарной точностью, с удвоенной точностью и с учетверенной точностью в регистре(ax) f.

Все инструкции FPop производят операции в соответствии со стандартом ANSI/IEEE 754-1985 над форматами данных одинарной, удвоенной и учетверенной точности. Формат записи данных с плавающей запятой приведен в подразделе «5.2 Форматы данных».

При выполнении инструкций FPop удвоенной точности не используется младший значащий бит адреса регистра f; при выполнении инструкций FPop учетверенной точности не используются два младших значащих бита адреса регистра f. Неиспользуемые биты адреса регистра зарезервированы и, для совместимости с последующими версиями, должны быть сброшены программным обеспечением. Если данные биты не сброшены в FPop с компонентами удвоенной и учетверенной точности, рекомендуется, чтобы FPop вызвала системное прерывание fp_exception с FSR.ftt = invalid_fp_register.

Если при помощи инструкции FPop2 (например, FCMP, FCMPE) устанавливаются коды условий операций с плавающей запятой, между FPop2 и последующей инструкцией FBfcc, должна быть выполнена хотя бы одна инструкция не-FPop2 (операция, которая не принадлежит к типу FPop2). В противном случае результат FBfcc не определен.

Инструкция FPop вызывает системное прерывание fp_disabled, если бит EF в PSR сброшен или FPU отсутствует.

Исключения операций с плавающей запятой могут вызывать строгие или отложенные системные прерывания.

Инструкции конвертации целочисленных данных в данные с плавающей запятой

Таблица А.39 – Инструкции конвертации целочисленных данных в данные с плавающей запятой

Код операции	opf	Операция
FiTOs	011000100	Конвертация целочисленных данных в данные с плавающей запятой одинарной точности
FiTOd	011001000	Конвертация целочисленных данных в данные с плавающей запятой удвоенной точности
FiTOq	011001100	Конвертация целочисленных данных в данные с плавающей запятой учетверенной точности

Формат 3:

10	rd	110100	unused (zero)	opf	rs2
31 30	29 25	24 19	18 14	13 5	4 0

Предлагаемый синтаксис на языке ассемблера:
fitos $freg_{rs2}$, $freg_{rd}$
fitod $freg_{rs2}$, $freg_{rd}$
fitoq $freg_{rs2}$, $freg_{rd}$

Описание

При помощи данных инструкций выполняется конвертация 32-х битового целочисленного слова из f[*rs2*] в число с плавающей запятой в назначенном формате. Результат данных инструкций записывается в регистр(-ы) f, заданный(-ые) полем rd.

Округление FiTOs выполняется в соответствии со значением поля RD в FSR.

Системные прерывания:
- fp_disabled;
- fp_exception (NX (только FiTOs), invalid_fp_register (FiTOd, FiTOq)).

Инструкции конвертации данных с плавающей запятой в целочисленные данные

Таблица А.40 – Инструкции конвертации данных с плавающей запятой в целочисленные данные

Код операции	opf	Операция
FsTOi	011010001	Конвертация данных с плавающей запятой одинарной точности в целочисленные данные
FdTOi	011010010	Конвертация данных с плавающей запятой удвоенной точности в целочисленные данные
FqTOi	011010011	Конвертация данных с плавающей запятой учетверенной точности в целочисленные данные

Формат 3:

10	rd	110100	rs1	opf	rs2
31 30	29 25	24 19	18 14	13 5	4 0

Предлагаемый синтаксис на языке ассемблера:
fstoi $freg_{rs2}$, $freg_{rd}$
fdtoi $freg_{rs2}$, $freg_{rd}$
fqtoi $freg_{rs2}$, $freg_{rd}$

Описание

При помощи данных инструкций выполняется конвертация операнда с плавающей запятой из регистра(-ов) f, заданного(-ых) полем rs2, в 32-х битовое целочисленное слово в f[*rd*].

Результат всегда округляется в направлении нуля (поле RD регистра FSR игнорируется).

Системные прерывания:
- fp_disabled;
- fp_exception (NV, NX, invalid_fp_register (FdTOi, FqTOi)).

Инструкции конвертации форматов данных с плавающей запятой

Таблица А.41 – Инструкции конвертации форматов данных с плавающей запятой

Код операции	opf	Операция
FsTOd	011001001	Конвертация данных одинарной точности в данные удвоенной точности
FsTOq	011001101	Конвертация данных одинарной точности в данные учетверенной точности
FdTOs	011000110	Конвертация данных удвоенной точности в инструкции одинарной точности
FdTOq	011001110	Конвертация данных удвоенной точности в данные учетверенной точности
FqTOs	011000111	Конвертация данных учетверенной точности в данные одинарной точности
FqTOd	011001011	Конвертация данных учетверенной точности в данные в данные удвоенной точности

Предлагаемый синтаксис на языке ассемблера:
fstod $freg_{rs2}$, $freg_{rd}$
fstoq $freg_{rs2}$, $freg_{rd}$
fdtos $freg_{rs2}$, $freg_{rd}$
fdtoq $freg_{rs2}$, $freg_{rd}$
fqtos $freg_{rs2}$, $freg_{rd}$
fqtod $freg_{rs2}$, $freg_{rd}$

Формат 3:

10	rd	110100	unused (zero)	opf	rs2
31 30	29 25	24 19	18 14	13 5	4 0

Описание

При помощи данных инструкций выполняется конвертация операнда с плавающей запятой из регистра(-ов) f, заданного(-ых) полем rs2, в число с плавающей запятой в формате назначения. Результат данных инструкций записывается в регистр(-ы) f, заданный(-ые) полем rd.

Округление выполняется в соответствии со значением поля RD в FSR.

При выполнении FqTOd, FqTOs и FdTOs («сужающие» инструкции конвертации) могут возникнуть исключения OF, UF и NX. При выполнении FdTOq, FsTOq и FsTOd («расширяющие» инструкции конвертации) данные исключения не возникают.

При выполнении данных шести инструкций может возникнуть исключение NV, если исходный операнд является «сигнализирующим» NaN (SNaN).

Системные прерывания:
- fp_disabled;
- fp_exception (OF, UF, NV, NX, invalid_fp_register).

Инструкции пересылки с плавающей запятой

Таблица А.42 – Инструкции пересылки с плавающей запятой

Код операции	opf	Операция
FMOVs	000000001	Перемещение
FNEGs	000000101	Логическая операция НЕ
FABSs	000001009	Взятие абсолютного значения

Формат 3:

10	rd	110100	unused (zero)	opf	rs2
31 30	29 25	24 19	18 14	13 5	4 0

Предлагаемый синтаксис на языке ассемблера:
fmovs $freg_{rs2}$, $freg_{rd}$
fnegs $freg_{rs2}$, $freg_{rd}$
fabss $freg_{rs2}$, $freg_{rd}$

Описание
FMOVs копируется содержимое f[*rs2*] в f[*rd*].
FNEGs копирует содержимое f[*rs2*] в f[*rd*] с комплементарным битом знака.
FABSs копирует содержимое f[*rs2*] в f[*rd*] со сброшенным битом знака.
Данные инструкций не выполняют округление.

Замечания по программированию
Для передачи значения повышенной точности между регистрами f требуется по одной инструкции FMOV на каждое слово данных.

Если исходный регистр и регистр назначения ($freg_{rs2}$ и $freg_{rd}$) одинаковы, то при помощи инструкций FNEGs (FABSs) выполняется операция логического НЕ (получение абсолютного значения) операнда любой точности, включая удвоенную и учетверенную точность.

Если исходный регистр и регистр назначения отличаются, при помощи инструкций FNEGs (FABSs) и FMOVs выполняется операция логического НЕ (получение абсолютного значения) для данных удвоенной точности. Аналогично, при помощи инструкций FNEGs (FABSs) и трех инструкций FMOVs выполняется операция логического НЕ (получение абсолютного значения) для данных учетверенной точности.

Системные прерывания:
- fp_disabled.

Инструкции вычисления квадратного корня с плавающей запятой

Таблица А.43 – Инструкции вычисления квадратного корня с плавающей запятой

Код операции	opf	Операция
FSQRTs	000101001	Квадратный корень одинарной точности
FSQRTd	000101010	Квадратный корень удвоенной точности
FSQRTq	000101011	Квадратный корень учетверенной точности

Формат 3:

10	rd	110100	unused (zero)	opf	rs2
31 30	29 25	24 19	18 14	13 5	4 0

Предлагаемый синтаксис на языке ассемблера:
fsqrts *freg*$_{rs2}$, *freg*$_{rd}$
fsqrtd *freg*$_{rs2}$, *freg*$_{rd}$
fsqrtq *freg*$_{rs2}$, *freg*$_{rd}$

Описание

При помощи данных инструкций генерируется квадратный корень операнда с плавающей запятой в регистре(-ах) f, заданном(-ых) полем rs2. Результат данных инструкций записывается в регистр(-ы) f, заданный(-ые) полем rd.

Округление выполняется в соответствии со значением поля rd в FSR.

Системные прерывания:
- fp_disabled;
- fp_exception (NV, NX, invalid_fp_register (FSQRTd, FSQRTq)).

Инструкции сложения и вычитания с плавающей запятой

Таблица А.44 – Инструкции сложения и вычитания с плавающей запятой

Код операции	opf	Операция
FADDs	001000001	Сложение одинарной точности
FADDd	001000010	Сложение удвоенной точности
FADDq	001000011	Сложение учетверенной точности
FSUBs	001000101	Вычитание одинарной точности
FSUBd	001000110	Вычитание удвоенной точности
FSUBq	001000111	Вычитание учетверенной точности

Формат 3:

10	rd	110100	rs1	opf	rs2
31 30	29 25	24 19	18 14	13 5	4 0

Предлагаемый синтаксис на языке ассемблера:
fadds *freg*$_{rs1}$, *freg*$_{rs2}$, *freg*$_{rd}$
faddd *freg*$_{rs1}$, *freg*$_{rs2}$, *freg*$_{rd}$
faddq *freg*$_{rs1}$, *freg*$_{rs2}$, *freg*$_{rd}$
fsubs *freg*$_{rs1}$, *freg*$_{rs2}$, *freg*$_{rd}$
fsubd *freg*$_{rs1}$, *freg*$_{rs2}$, *freg*$_{rd}$
fsubq *freg*$_{rs1}$, *freg*$_{rs2}$, *freg*$_{rd}$

Описание

При помощи инструкций сложения с плавающей запятой выполняется сложение регистра(-ов) f, заданного(-ых) полем rs1, и регистра(-ов) f, заданного(-ых) полем rs2, с записью суммы в регистр(ы) f, заданный(-ые) полем rd.

При помощи инструкций вычитания с плавающей запятой выполняется вычитание регистра(-ов) f, заданного(-ых) полем rs2,из регистра(-ов) f, заданного(-ых) полем rs1, с записью разницы в регистр(-ы) f, заданный(-ые) полем rd.

Системные прерывания:
- fp_disabled;
- fp_exception (OF, UF, NX, NV ($\infty - \infty$));

- invalid_fp_register (все, за исключением инструкций FADDs и FSUBs).

Инструкции умножения и деления с плавающей запятой

Таблица А.45 – Инструкции умножения и деления с плавающей запятой

Код операции	opf	Операция
FMULs	001001001	Умножение одинарной точности
FMULd	001001010	Умножение удвоенной точности
FMULq	001001011	Умножение учетверенной точности
FsMULd	001101001	Умножение операнда одинарной точности на операнд удвоенной точности
FdMULq	001101110	Умножение операнда удвоенной точности на операнд учетверенной точности
FDIVs	001001101	Деление одинарной точности
FDIVd	001001110	Деление удвоенной точности
FDIVq	001001111	Деление учетверенной точности

Формат 3:

10	rd	110100	rs1	opf	rs2
31 30	29 25	24 19	18 14	13 5	4 0

Предлагаемый синтаксис на языке ассемблера:

fmuls $freg_{rs1}$, $freg_{rs2}$, $freg_{rd}$
fmuld $freg_{rs1}$, $freg_{rs2}$, $freg_{rd}$
fmulq $freg_{rs1}$, $freg_{rs2}$, $freg_{rd}$
fsmuld $freg_{rs1}$, $freg_{rs2}$, $freg_{rd}$
fdmulq $freg_{rs1}$, $freg_{rs2}$, $freg_{rd}$
fdivs $freg_{rs1}$, $freg_{rs2}$, $freg_{rd}$
fdivd $freg_{rs1}$, $freg_{rs2}$, $freg_{rd}$
fdivq $freg_{rs1}$, $freg_{rs2}$, $freg_{rd}$

Описание

При помощи инструкций умножения с плавающей запятой выполняется умножение регистра(-ов) f, заданного(-ых) полем rs1, на регистр(ы) f, заданный(-ые) полем rs2, с записью произведения в регистр(ы) f, заданный(-ые) полем rd.

Инструкция FsMULd реализует точное произведение удвоенной точности двух операндов одинарной точности, без обратного переполнения, переполнения или ошибки округления. Аналогично FdMULq обеспечивает точное произведение учетверенной точности двух операндов удвоенной точности.

При помощи инструкций деления с плавающей запятой выполняется деление регистра(-ов) f, заданного(-ых) полем rs1, на регистр(ы) f, заданный(-ые) полем rs2, с записью частного в регистр(-ы) f, заданный(-ые) полем rd.

Системные прерывания:
- fp_disabled;
- fp_exception (OF, UF, DZ (только FDIV), NV, NX;
- invalid_fp_register (все, за исключением инструкций FMULs и FDIVs).

Инструкции сравнения с плавающей запятой

Таблица А.46 – Инструкции сравнения с плавающей запятой

Код операции	opf	Операция
FCMPs	001010001	Сравнение операндов одинарной точности
FCMPd	001010010	Сравнение операндов удвоенной точности
FCMPq	001010011	Сравнение операндов учетверенной точности
FCMPEs	001010101	Сравнение операндов одинарной точности и вызова исключения, если не упорядочено
FCMPEd	001010110	Сравнение операндов удвоенной точности и вызова исключения, если не упорядочено
FCMPEq	001010111	Сравнение операндов учетверенной точности и вызова исключения, если отношение не упорядочено

Формат 3:

10	unused (zero)	110101	rs1	opf	rs2
31 30	29 25	24 19	18 14	13 5	4 0

Предлагаемый синтаксис на языке ассемблера:

fcmps $freg_{rs1}$, $freg_{rs2}$
fcmpd $freg_{rs1}$, $freg_{rs2}$
fcmpq $freg_{rs1}$, $freg_{rs2}$
fcmpes $freg_{rs1}$, $freg_{rs2}$
fcmped $freg_{rs1}$, $freg_{rs2}$
fcmpeq $freg_{rs1}$, $freg_{rs2}$

Описание

При помощи данных инструкций выполняется сравнение регистра(-ов) f, заданного(-ых) полем rs1, с регистром(-ами) f, заданным(-ыми) полем rs2, с установкой кодов условий операций с плавающей запятой в соответствии с таблицей А.47.

Таблица А.47 – Коды условий операций с плавающей запятой (fcc)

fcc	Отношение
0	$freg_{rs1} = freg_{rs2}$
1	$freg_{rs1} < freg_{rs2}$
2	$freg_{rs1} > freg_{rs2}$
3	$freg_{rs1}$? $freg_{rs2}$ (не упорядочено)

Инструкции «сравнения и вызова исключения, если не упорядочено» (FCMPEs, FCMPEd и FCMPEq) вызывают исключение недействительности (NV), если один из операндов является «сигнализирующим» NaN или «тихим» NaN. FCMP вызывает исключение недействительности (NV), если любой из операндов является «сигнализирующим» NaN.

Инструкция не-FPop2 (операция, которая не принадлежит к типу FPop2) должна быть выполнена между инструкцией FPop2 (FCMP или FCMPE) и последующей инструкцией FBfcc. В противном случае результат FBfcc не определен.

Системные прерывания:

- fp_disabled;
- fp_exception (NV, invalid_fp_register(все инструкций, за исключением FCMPs и FCMPEs)).

Инструкции операций сопроцессора

Таблица А.48 – Инструкции операций сопроцессора

Код операции	op3	Операция
CPopl	110110	Операции сопроцессора
CPop2	110111	Операции сопроцессора

Формат 3:

10	rd	110110	rs1	opc	rs2
31 30	29 25	24 19	18 14	13 5	4 0

10	rd	110111	rs1	opc	rs2
31 30	29 25	24 19	18 14	13 5	4 0

Предлагаемый синтаксис на языке ассемблера:
cpop1 *opc*, $creg_{rs1}$, $creg_{rs2}$, $creg_{rd}$
cpop2 *opc*, $creg_{rs1}$, $creg_{rs2}$, $creg_{rd}$

Стоит отметить, что выше приведен «универсальный» предлагаемый синтаксис на языке ассемблера для данных инструкций, который может использоваться вне зависимости от конкретной реализации ассемблера SPARC. Предполагается, что ассемблеры, поддерживающие конкретные реализации сопроцессора, (также) поддерживают синтаксис с большим количеством мнемонических имен инструкций и меньшим количеством операндов.

Описание

Кодирование инструкций управления сопроцессора (CPop) выполняется с помощью двух форматов типа 3: CPop1 и CPop2. Интерпретация полей rd, rs1, opc и rs2 зависит от сопроцессора. Следует отметить, что инструкции загрузки / сохранения сопроцессора не являются инструкциями «CPop».

Инструкции CPop1 не изменяют коды условий сопроцессора. Инструкции CPop2 изменяют коды условий сопроцессора.

Все инструкции CPop принимают все входные операнды и возвращают результат только в регистры сопроцессора. Поддерживаемые сопроцессором типы данных зависят от сопроцессора. Выравнивание операндов в сопроцессоре зависит от сопроцессора.

Если бит EC в PSR сброшен или сопроцессор отсутствует, инструкция CPop вызывает системное прерывание cp_disabled.

Условия, при которых инструкция CPop вызывает системное прерывание cp_exception, зависят от сопроцессора.

Системные прерывания:
- cp_disabled;
- cp_exception (в зависимости от сопроцессора).

Пример программы

```
/* a simple program computing the first n Fibonacci numbers */

#include <stdlib.h>
#include <stdio.h>

extern unsigned * fibonacci(int n);

#define MAX_FIB_REPRESENTABLE 49

/* compute the first n Fibonacci numbers */
unsigned * fibonacci(int n) {
  static unsigned fib_array[MAX_FIB_REPRESENTABLE] = {0,1};
  unsigned prev_number = 0;
  unsigned curr_number = 1;
  int i;
  if (n >= MAX_FIB_REPRESENTABLE) {
    printf("Fibonacci(%d) cannot be represented in a 32 bit word\n", n);
    exit(1);
  }

  for (i = 2; i < n; i++) {
    fib_array[i] = prev_number + curr_number;
    prev_number = curr_number;
    curr_number = fib_array[i];
  }

  return(fib_array);
 }

int main() {
  int n, i;
  unsigned * result;
  printf("Fibonacci(n):, please enter n:\n");
  scanf("%d", &n);
  result = fibonacci(n);
  for (i = 1; i <= n; i++) {
    printf("Fibonacci (%d) is %u\n", i, *result++);
  }
  return 0;
}

! A simple program computing the first n Fibonacci numbers, showing
! various pseudo-operations, sparc and synthetic instructions
!
! pseudo-operations:        .align, .ascii, .file, .global, .ident,
!                           .proc, .section, .size, .skip, .type, .word
! sparc instructions:       add, bg, bge, bl, ble, ld, or, restore,
!                           save, sethi, st
! synthetic instructions:   call, cmp, inc, mov, ret

.file "fibonacci.c"         ! the original source file name
```

```
        .section ".text"            ! text section (executable instructions)
        .proc 79                    ! subroutine fibonacci, it's return value will be in %i0
        .global fibonacci           ! fibonacci() can be referenced outside this file
        .align 4                    ! align the beginning of this section to word boundary

fibonacci:
        save %sp, -96, %sp          ! create new stack frame and register window for
                                    ! this subroutine

        /* if (n >= MAX_FIB_REPRESENTABLE) { */
                                    ! note, C style comment strings are also permitted
        cmp %i0, 49                 ! n >= MAX_FIB_REPRESENTABLE ?
                                    ! note, n, the 1st parameter to fibonacci(), is stored
                                    ! in %i0 upon entry
        bl .L77003
          mov 0, %i2                ! initialization of variable prev_number is executed
                                    ! in the delay slot

        /* printf("Fibonacci(%d) cannot be represented in a 32 bits word\n", n); */
        sethi %hi(.L20), %o0        ! if branch not taken, call printf(),set up 1st,
        or %o0, %lo(.L20), %o0      ! 2nd argument in %o0, %o1;
        call printf, 2              ! the ",2" means there are 2 out registers used
          mov %i0, %o1              ! as arguments

        /* exit(1); */
        call exit, 1
          mov 1, %o0

.L77003:                            ! initialize variables before the loop
        /* for (i = 2; i < n; i++) { */
        mov 1, %i4                  ! curr_number = 1
        mov 2, %i3                  ! i = 2
        cmp %i3, %i0                ! i <= n?
        bge .L77006                 ! if not, return
          sethi %hi(.L16 + 8), %o0  ! use %i5 to store &fib_array[i]
        add %o0, %lo(.L16 + 8), %i5

.LY1:                               ! loop body
        /* fib_array[i] = prev_number + curr_number; */
        add %i2, %i4, %i2           ! fib_array[i] = prev_number + curr_number
        st %i2, [%i5]
        /* prev_number = curr_number; */
        mov %i4, %i2                ! prev_number = curr_number
        /* curr_number = fib_array[i]; */
        ld [%i5], %i4               ! curr_number = fib_array[i]
        inc %i3                     ! i++
        cmp %i3, %i0                ! i <= n?
        bl .LY1                     ! if yes, repeat loop
          inc 4, %i5                ! increment ptr to fib_array[]

.L77006:
        /* return(fib_array); */
        sethi %hi(.L16), %o0        ! return fib_array in %i0
```

```
        add %o0, %lo(.L16), %i0
        ret
          restore                       ! destroy stack frame and register window

        .type fibonacci,#function       ! fibonacci() is of type function
        .size fibonacci,(.-fibonacci)   ! size of function:
                                        !  current location counter minus beginning definition
                                        !  of function
        .proc 18                        ! main program
        .global main
        .align 4
main:
        save %sp,-104,%sp               ! create stack frame for main()

        /* printf("Fibonacci(n):, please input n:\n"); */
        sethi %hi(.L31), %o0            ! call printf, with 1st arg in %o0
        call printf, 1
          or %o0, %lo(.L31), %o0

        /* scanf("%d", &n); */
        sethi %hi(.L33), %o0            ! call scanf, with 1st arg, in %o0
        or %o0, %lo(.L33), %o0          ! move 2nd arg. to %o1, in delay slot
        call scanf, 2
          add %fp, -4, %o1

        /* result = fibonacci(n); */
        call fibonacci, 1
          ld [%fp - 4], %o0
                                        ! some initializations before the for-loop,
                                        ! put the variables in registers
        /* for (i = 1; i <= n; i++) */
        mov 1, %i5                      ! %i5 <-- i
        mov %o0, %i4                    ! %i4 <-- result
        sethi %hi(.L38), %o0            ! %i2 <-- format string for printf
        add %o0, %lo(.L38), %i2
        ld [%fp - 4], %o0               ! test if (i <= n) ?
        cmp %i5, %o0                    ! note, n is stored in [%fp - 4]
        bg .LE27
          nop

.LY2:                                   ! loop body
        /* printf("Fibonacci (%d) is %u\n", i, *result++); */
        ld [%i4], %o2                   ! call printf, with (*result) in %o2,
        mov %i5, %o1                    ! i in %o1, format string in %o0
        call printf, 3
          mov %i2, %o0
        inc %i5                         ! i++
        ld [%fp - 4], %o0               ! i <= n?
        cmp %i5, %o0
        ble .LY2
          inc 4, %i4                    ! result++
        cmp %i5,%o0
```

```
.LE27:
        ret
          restore

        .type main,#function            ! type and size of main
        .size main,(.-main)
        .section ".data"                ! switch to data section (contains initialized data)
        .align 4
.L16:
        /* static unsigned fib_array[MAX_FIB_REPRESENTABLE] = {0,1}; */
        .align 4                        ! initialization of first 2 elements of fib_array[]
        .word 0
        .align 4
        .word 1
        .skip 188
        .type .L16,#object              ! storage allocation for the rest of fib_array[]
        .section ".data1"               ! the ascii string data are entered into the .data1
                                        ! section;
                                        ! #alloc: memory would be allocated for this section
                                        ! during run time
                                        ! #write: the section contains data that is writeable
                                        ! during process execution
        .align 4
.L20:                                   ! ascii strings used in the printf statements
        .ascii "Fibonacci(%d) cannot be represented in a 32 bit w"
        .ascii "ord\n\0"
        .align 4                        ! align the next ascii string to word boundary
.L31:
        .ascii "Fibonacci(n):, please enter n:\n\0"
        .align 4
.L33:
        .ascii "%d\0"
        .align 4
.L38:
        .ascii "Fibonacci (%d) is %u\n\0"
        .ident "acomp: (CDS) SPARCompilers 2.0 05 Jun 1991"
                                        ! an identification string produced by the compiler
                                        ! to be entered into the .comment section
```

Приложение Б

(обязательное)

Конфигурация plug & play

Общая информация

GRLIB-реализация AHB шины включает механизм, обеспечивающий поддержку plug & plug. Она состоит из трех частей: идентификации подключаемого модуля (ведущий или ведомый), отображения адресов ведомых и маршрутизации прерываний. Информация plug & plug для каждого AHB устройства состоит из конфигурационной записи, содержащей восемь 32-битных слов. Первое слово называется регистром идентификации и содержит информацию о типе устройства и маршрутизации прерывания. Последние четыре слова называются регистрами адресов банков и содержат информацию об отображении адресов ведомых устройств. Оставшиеся три слова не детерминированы и на этапе разработки в них может быть сохранена специфичная для модуля конфигурационная информация. На рисунке Б.1 приведен формат конфигурационной записи в области plug & plug.

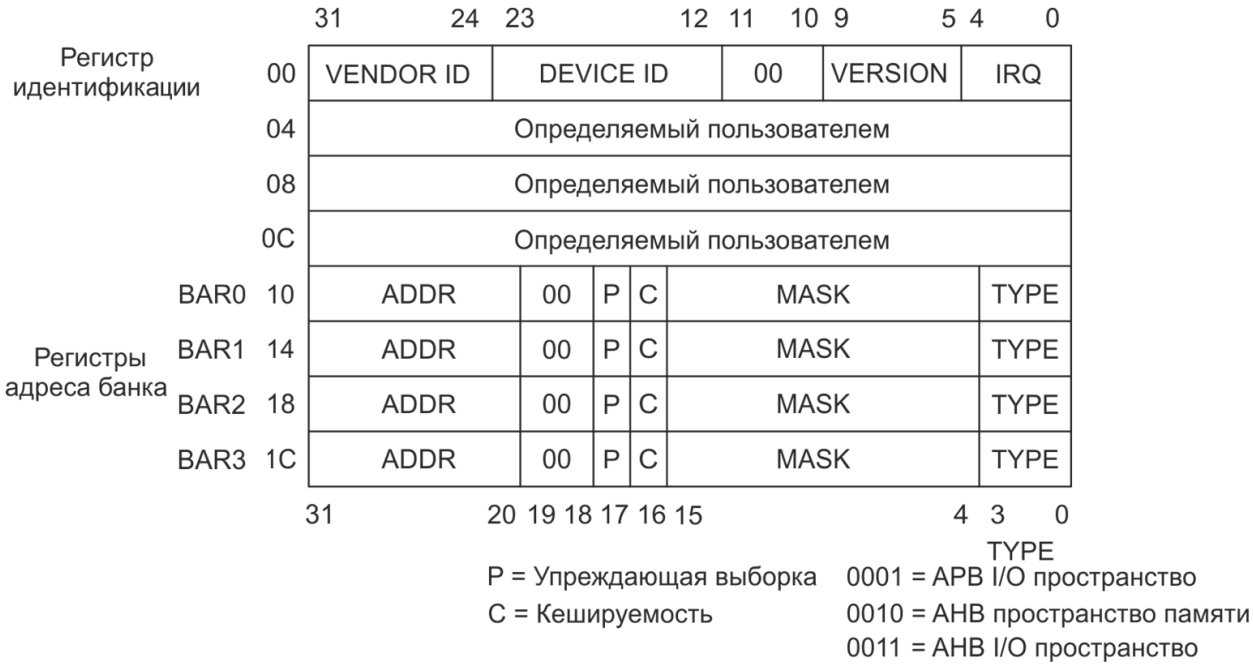


Рисунок Б.1 – Формат конфигурационной записи plug & plug

Информация plug & plug обо всех подключенных AHB устройствах приведена в виде доступной только для чтения таблицы, отображенной по фиксированному адресу на шине AHB, обычно 0xFFFFF000. Конфигурационные записи ведущих AHB размещены по адресам 0xFFFFF000 – 0xFFFFF800, в то время как записи ведомых располагаются по адресам 0xFFFFF800 – 0xFFFFFFFC. Поскольку каждая запись состоит из восьми слов, в таблице есть место для 64 ведущих и 64 ведомых устройств. Операционная система plug & plug (или любое другое приложение) может сканировать конфигурационную таблицу и автоматически определять, какие блоки присутствуют на шине AHB, как они сконфигурированы и где они размещены (для ведомых).

Поскольку конфигурационная информация является неизменяемой, её хранение может быть эффективно реализовано, через использование небольшого ПЗУ или относительно небольшого количества вентилей.

Идентификация устройства

Идентификационный регистр содержит три поля, позволяющие однозначно идентифицировать подключенное на шину AHB устройство: идентификатор поставщика (vendor ID), идентификатор устройства (device ID) и номер его версии. Идентификатор поставщика – это уникальный номер, закрепленный за поставщиком IP (Intellectual Property) или организацией. Идентификатор устройства – это уникальный номер, присвоенный поставщиком конкретному IP блоку. Номер версии может быть использован для идентификации различных версий модуля.

Идентификатор поставщика присваивается компанией Cobham Gaisler. В таблице Б.1 приведен список идентификаторов устройств различных поставщиков.

Таблица Б.1 – Идентификаторы устройств различных поставщиков

Поставщик	Идентификатор поставщика
Gaisler Research	0x01
Pender Electronic Design	0x02
European Space Agency	0x04
Astrium EADS	0x06
OpenChip.org	0x07
OpenCores.org	0x08
Various contributions	0x09
Eonic BV	0x0B
Telecom ParisTech	0x0C
DTU Space	0x0D
BSC	0x0E
Radionor	0x0F
Gleichmann Electronics	0x10
Menta	0x11
Sun Microsystems	0x13
Movidia	0x14
Orbita	0x17
Siemens AG	0x1A
Synopsys Inc.	0x21
NASA	0x22
NIIET	0x23
S3 Group	0x31
Actel Corporation	0xAC
AppleCore	0xAE
TU Braunschweig C3E	0xC3
CBK PAN	0xC8
Caltech	0xCA
Ceton Corporation	0xCB
Embeddit	0xEA

Идентификатор поставщика 0x00 зарезервирован, чтобы указывать, что на данной позиции нет модуля. Неиспользованные позиции в конфигурационной таблице будут иметь значение идентификационного регистра, равное 0.

Декодирование адреса

Контроллер шины AHB, в который встроен декодер адреса, использует конфигурационную информацию ведомых устройств для автоматической генерации сигналов выбора ведомого (HSEL).

Диапазон адресов AHB для каждого ведомого определяется регистрами адреса банка (BAR). Декодирование адреса выполняется сравнением 12-битового поля ADDR регистра BAR с частью AHB адреса (HADDR). Существует два типа банков, определенных для шины AHB: банк памяти AHB и банк области ввода-вывода AHB. Декодирование AHB адреса выполняется двумя разными способами.

Для банков памяти AHB декодирование адреса выполняется сравнением 12-битового поля ADDR в регистре BAR с 12 старшими разрядами адреса AHB (HADDR[31:20]). В случае равенства будет сгенерирован сигнал HSEL. Это означает, что минимальный диапазон адресов, занимаемый банком памяти AHB, составляет 1 Мбайт. Для реализации более широкого диапазона адресов при сравнении используются только те биты, которые установлены в поле MASK регистра BAR. Таким образом, сигнал HSEL будет сгенерирован, если следующее равенство будет истинным:

((BAR.ADDR xor HADDR [31:20]) and BAR.MASK) = 0.

Для примера, при декодировании 16-Мбайтного банка памяти AHB, расположенного по адресу 0x24000000, в поле ADDR должно находиться значение 0x240, а в поле MASK – 0xFF0. Стоит отметить, что если поле MASK = 0, то BAR будет заблокирован, вместо того чтобы занимать весь диапазон адресов области AHB.

Для банков области ввода-вывода AHB декодирование адреса выполняется сравнением 12-битового поля ADDR в регистре BAR с 12 разрядами AHB адреса (HADDR[19:8]). В случае равенства будет сгенерирован сигнал HSEL. Это означает, что минимальный диапазон адреса, занимаемый банком области ввода-вывода AHB, составляет 256 байт. Для реализации более широкого диапазона адресов при сравнении используются только те биты, которые установлены в поле MASK регистра BAR. Таким образом, сигнал HSEL будет сгенерирован, если следующее равенство будет истинным:

((BAR.ADDR xor HADDR [19:8]) and BAR.MASK) = 0.

Старшие 12 бит адреса AHB (HADDR[31:20]) всегда будут содержать фиксированное значение 0xFFF, фактически размещая все банки области ввода-вывода AHB в адресном пространстве 0xFFF00000 – 0xFFFFFFFF. Для примера, при декодировании 4-Кбайтного банка области ввода-вывода AHB по адресу 0xFFF24000, в поле ADDR должно находиться значение 0x240, а в поле MASK – 0xFF0. Стоит отметить, что если поле MASK = 0, то BAR будет заблокирован, вместо того чтобы занимать весь диапазон адресов области ввода-вывода AHB.

Ведомый AHB может иметь до четырех регистров отображения адреса, посредством этого декодируется до четыре независимых областей адресного пространства AHB. Сигнал HSEL устанавливается при выборе любой из этих областей.

Кэшируемость

В процессорных системах без MMU кэшируемые области обычно статически определены в контроллерах кэша. В процессорах LEON4 таблица кэшируемости строится на этапе синтеза на основе информации о кэшируемости, приведенной в конфигурационных записях AHB. В этом случае кэшируемость областей всегда имеет отображение в текущей конфигурации.

В системах с MMU информация о кэшируемости может быть прочитана программным способом из конфигурационных записей. Это позволяет операционной системе строить таблицу страниц MMU с корректным набором битов кэшируемости, установленным в записях таблицы страниц.

Управление прерываниями

GRLIB обеспечивает унифицированную схему обработки прерываний через добавление 32 сигналов прерывания (HIRQ) к AHB шине, как в качестве входов, так и в качестве выходов. И ведущие, и ведомые устройства шины AHB могут, как устанавливать, так и считывать любое из прерываний.

Вывод каждого из ведущих включает в себя все 32 сигнала прерывания, объединенные в один вектор. Ведущий шины AHB может управлять одной (определяется значением поля IRQ) или несколькими линиями (идут по порядку, начиная с IRQ).

То же самое относится к выходу каждого из ведомых устройств, которое включает в себя все 32 сигнала прерывания. Управляемые конкретным модулем линии также определены через значение поля IRQ.

Контроллер шины AHB в GRLIB обеспечивает объединение прерываний. Каждый элемент HIRQ формируется через операцию логического ИЛИ между всеми выходными сигналами прерываний ведущих и ведомых шины AHB. Объединенный результат возвращается и на входы ведущих и на входы ведомых шины AHB. Следовательно, ведущие и ведомые AHB шины совместно используют те же самые 32 сигнала прерывания.

Контроллер прерываний, расположенный на шине AHB, осуществляет мониторинг объединенного вектора прерываний и генерирует соответствующее системное прерывание процессора.

Приложение В

(справочное)

Версии РП

В таблице В.1 отображены изменения, вносимые в РП.

Таблица В.1 – Список внесенных изменений

Дата, № изм.	Описание внесенных изменений	Всего листов
25.12.15	Стартовая версия РП	361
28.12.15 изм. 1	Корректировка РП	361
26.03.18, изм. 2	Корректировка структуры и оформления РП	372
09.06.20 изм. 3	Введен типономинал ИС 1906ВМ01А6	396
	Изменена структура раздела «2 Краткое техническое описание» (пункты вынесены в качестве отдельных подразделов), переименованы подразделы «2.1» и «2.7»	
	В подразделе «2.3 Функциональное назначение выводов» скорректировано описание вывода CFG_8B_PROM, изменен тип выводов JTAG_TDO, PCI_RST# и PCI_REQ#, вывод PCI_HOST# указан как сигнал с активным низким логическим уровнем, увеличен шрифт и изменен порядок следования части групп выводов. Выводы A23, AF26, N25, AE3, F22, D24 имеют схему «Pull-up»	
	Подраздел «2.5 Карта памяти микропроцессор» дополнен, внесены коррективы в размер областей на рисунке 2.5.1	
	Подраздел «2.6 Механизмы сбоеустойчивости» дополнен и уточнен, скорректирована разрядность кода Хэмминга в таблице 2.6.2	
	Подраздел «2.7» дополнен рекомендацией по конфигурированию доступов во внешнюю память, расширен диапазон рабочих температур микропроцессора, скорректировано значение его тактовой частоты и некоторых электрических параметров	
	В подразделах «3.1» и «4.1» изменено описание функционирования вывода RESETN#	
	В подраздел «3.2» добавлен перечень выводов неиспользуемых интерфейсов, обязательных к подключению	
	В подраздел «3.3 Внешнее ОЗУ» добавлена ссылка на вариант коррекции положения фронта SDCLK относительно CLK средствами процессора, добавлены пояснения по активации контроллера SDRAM	
	Подраздел «3.5 Запуск программы из ПЗУ» исправлен и дополнен, скорректирован пример файла «bdinit.S»	
	Добавлен подраздел «3.8 Требования по инициализации областей памяти при использовании помехоустойчивого кодирования»	
	В подраздел «4.3 Контроллер конфигурации» добавлено описание регистров SDCLK_CTRL, BYPASS_CTRL и BYPASS_CHECKBITS, в таблице 4.3.8 скорректировано описание битов управления PROM_8B, LIP, LOP, LIS, LOS, в таблицу 4.3.6 добавлен бит управления WBSD, в таблицу 4.3.11 добавлено поле PM и изменено описание полей	

	В подразделе «4.4 Процессорное ядро LEON4» видоизменен рисунок «4.4.1». В подпункт «Режим пониженного потребления энергии» добавлена оценка ожидаемого снижения тока потребления. Подпункт «Исключения» дополнен, приведен пример размещения в памяти векторов системных прерываний. Подпункты «Адресация кэша инструкций» и «Адресация кэша данных» объединены в подпункт «Отображение адреса в кэш инструкций и кэш данных», уточнено описание и назначение полей. Уточнен подпункт «Отслеживание (Snooping)». В пункте «4.4.8» дополнено описание регистров %wim и %tbr, скорректировано описание подпункта «Аппаратные точки останова/контрольные точки данных», добавлен подпункт «Регистр управления защитой», а в подраздел «2.6» – рекомендация по его конфигурации	
	В подразделах «4.5 Контроллер PROM / IO» и «4.6 Контроллер SRAM» изменены требования к использованию сигналов WREN[0:3] и WRITEN, скорректированы временные диаграммы и информация о конфигурировании временных диаграмм в пункте «4.5.4». В пункте «4.5.5» переопределено условие входа в 8-битный режим шины данных PROM. Изменено описание сигнала READ. Изменены формулы расчета проверочных бит в режиме 8-битной шины области PROM (подпункт «8-битный режим PROM») и их описание	
	Скорректирован и дополнен подраздел «4.7 Контроллер SDRAM», добавлено описание режимов деактивации открытых рядов, скорректированы формулы расчета временных параметров, добавлен пункт «4.7.4 Блок управления трактом задержки SDCLK». Скорректирована формула расчета бита CB1 в подпункте «Код Хэмминга» пункта «4.7.8»	
	В подразделе «4.8 Контроллер прерываний» скорректировано содержание и описание полей регистров, добавлены ссылки на регистры, исправлено описание полей таблицы 4.8.1	
	В подразделе «4.9 Отладочный модуль LEON4» скорректировано описание в подразделе «4.9.1», исправлена и дополнена таблица 4.9.3, в пункты «4.9.3» – «4.9.5» добавлено детальное описание доступа к данным буферов трассировки и регистрового файла блока IU, приведен список реализованных %asr16 – %asr31, скорректировано смещение адреса регистра контрольной точки 2 AHB. Уточнено описание поведения таймеров при входе процессора в режим остановки. Добавлены значения по сбросу и режим доступа к полям регистров пункта «4.9.6 Регистры DSU», в пункт «Регистр управления буфером трассировки AHB» добавлено описание бита DF. Приведены реализованные разрядности полей DCNT, INDEX и ITPOINTER (таблицы 4.9.13, 4.9.14 и 4.9.17), скорректировано описание бита HL регистра управления DSU	
	В подразделе «4.14 Порт ввода-вывода общего назначения» скорректирован текст пунктов «4.14.1» и «4.14.2», внесено уточнение о синхронности захвата внешних прерываний с выводов, исправлен адрес регистра области plug & play модуля GPIO. Добавлены значения по сбросу и режим доступа к полям регистров, в таблицу 4.14.8 добавлены новые поля	
	Скорректирован и дополнен подраздел «4.15 Интерфейс UART»	

	В подразделе «4.16 Интерфейс MIL-STD-1553/AS15531» изменен формат записи данных в таблицах, скорректированы наименования части регистров. Отредактирован пункт «4.16.8 Регистры»: исправлен пропуск в нумерации после таблицы 4.16.15, добавлены значения по сбросу регистров, изменены формулировки некоторых примечаний, в таблице 4.16.17 переименованы битовые поля, изменен формат записи описания поля BCFEAT таблицы 4.16.22	
	В подраздел «4.18 Интерфейс Ethernet с поддержкой EDCL» добавлено описание типов полей регистров и их значения по сбросу	
	Исправлено описание полей в таблице 4.19.27	
	В подразделе «4.20 Интерфейс USB 2.0» скорректированы наименования регистров и их описание	
	В подразделе «4.21 Интерфейс PCI» по тексту исправлен номер генерируемого прерывания (в соответствии с таблицей 4.8.1), добавлены значения по сбросу и режим доступа к полям регистров. Скорректированы пункты «4.21.2», «4.21.6» и «4.21.8». Добавлен пункт «4.21.9 Арбитраж шины PCI». Выполнена частичная коррекция пункта «4.21.10 Регистры»	
	Подраздел «4.22 Контроллер шины AHB» отредактирован	
	Подраздел «4.23 Контроллер шины APB» отредактирован	
	Раздел «5 Описание архитектуры» отредактирован	
	Раздел «6 Введение в систему команд» отредактирован	
	«Приложение А» отредактировано	
	«Приложение Б» отредактировано	
	Добавлено «Приложение В» (справочное)	
	Именование исключения с tt =0x2B по тексту изменилось с «write error» на «data_store_error»	
	Обозначение операции умножения по тексту изменено с «*» на «×»	
	По тексту унифицировано наименование полей регистров состояния процессора icc, fcc и ccc – «коды условий»	
07.12.21 изм. 4	Скорректирован подраздел «3.4 Инициализация кэша, конфигурация доступа к PROM». В подраздел «3.5 Запуск программы из ПЗУ» добавлено пояснение обязательности инициализации кэшей при использовании утилиты **mkprom2**	396

Лист регистрации изменений								
Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ докум.	Подп.	Дата
	Измененных	Замененных	Новых	Аннулированных				