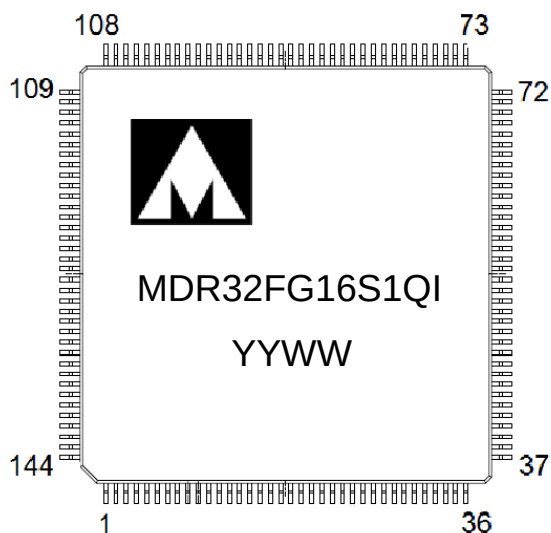


Двухъядерный процессор цифровой обработки сигналов с фиксированной запятой K1901ВЦ1QI



YY – год выпуска

WW – неделя выпуска

Основные параметры микросхемы:

- 32-разрядная архитектура RISC;
- 16-разрядная архитектура DSP;
- Встроенная память программ 128 Кбайт;
- Внешняя шина 32 разряда;
- 16-канальный 12-разрядный АЦП;
- Два 12-разрядных ЦАП;
- Три интерфейса UART;
- Интерфейс I2C;
- Аудиокодек;
- Блок поддержки шифрования ГОСТ 28147-89;
- Компаратор;
- Напряжение источника питания, от 3,0 до 3,6 В;
- Температурный диапазон от минус 40 °С до плюс 85 °С.

Тип корпуса:

- 144-выводной пластмассовый корпус LQFP144.

Масса микросхемы не более 2 г.

Основные характеристики

Особенности ядра и памяти RISC:

- 32-битное ядро RISC, тактовая частота до 100 МГц, производительность 1,25 DMIPS/МГц (Dhrystone 2.1) при нулевой задержке памяти;
- Блок аппаратной защиты памяти MPU;
- Умножение за один цикл, аппаратная реализация деления;
- Встроенная энергонезависимая память программ Flash типа размером 128 Кбайт;
- Встроенное ОЗУ размером 32 Кбайт;
- Контроллер внешней системной шины с поддержкой микросхем памяти СОЗУ, ПЗУ, NAND Flash.

Важно: спецификация действительна совместно с документом K1901ВЦ1QI Errata Notice.

Особенности ядра и памяти DSP:

- 16-битное ядро DSP аналог TMS320C54, тактовая частота до 100 МГц;
- встроенная память программ ОЗУ размером 128 Кбайт;
- встроенная память данных ОЗУ размером 128 Кбайт;
- максимальное быстроедействие при одновременной работе ядер RISC и DSP с памятью подсистемы DSP;
- отображение в адресное пространство RISC.

Особенности периферии и режима отладки:

- контроллер прямого доступа в память с функциями передачи
- Периферия-Память, Память-Память;
- контроллер интерфейса USB с функциями работы Device и Host;
 - контроллеры интерфейсов UART, SPI, I2C;
 - до 96 пользовательский линий ввода-вывода;
 - блок аппаратной поддержки шифрования ГОСТ 28147-89;
 - контроллер интерфейса SDIO;
 - последовательные отладочные интерфейсы SWD и JTAG.

Особенности аналоговых модулей:

- два 12-разрядных АЦП (до 16 каналов);
- измеряемый диапазон напряжений от 0 до 3,6 В;
- температурный датчик;
- двух канальный 12-разрядный ЦАП;
- встроенный компаратор;
- аудиокодек.

Особенности питания и тактовой частоты:

- внешнее питание 3,0...3,6 В;
- встроенный регулятор напряжения на 1,8 В для питания ядра;
- встроенные схемы контроля питания;
- встроенный домен с батарейный питанием;
- встроенный подстраиваемый RC-генератор 8 МГц;
- встроенный подстраиваемый RC-генератор 40 КГц;
- внешние два кварцевых резонатора на 2÷16 МГц и 32 кГц;
- встроенный умножитель тактовой частоты PLL для ядра RISC;
- встроенный умножитель тактовой частоты PLL для ядра DSP;
- встроенный умножитель тактовой частоты PLL для контроллера USB.

Специализированные особенности:

- режимы SLEEP, DEEPSLEEP и STANDBY;
- батарейный домен с часами реального времени и регистрами аварийного сохранения.

Содержание

1	Введение	23
2	Условное графическое обозначение	24
3	Описание выводов	26
4	Структурная блок-схема микросхемы	33
5	Указания по применению и эксплуатации	34
6	Описание функционирования микросхемы	35
6.1	Система питания	35
6.2	Структурная схема подачи питания	36
6.3	Схема сброса при включении и выключении основного питания	37
6.4	Организация памяти	38
6.4.1	Структурная схема	38
6.4.2	Карта памяти DSP	51
6.4.3	Отображение памяти DSP в памяти RISC	57
6.5	Загрузочное ПЗУ и режимы работы микросхемы	58
6.5.1	Микроконтроллер в режиме отладки	60
6.5.2	Микропроцессор в режиме отладки	61
6.5.3	UART-загрузчик без отладки	61
7	Контроллер FLASH-памяти программ	67
7.1	Работа FLASH-памяти программ в обычном режиме	67
7.2	Работа FLASH-памяти программ в режиме программирования	68
7.2.1	Стирание всей памяти	69
7.2.2	Стирание страницы памяти размером 4 Кбайт	70
7.2.3	Запись 32-битного слова в память	70
7.2.4	Чтение 32-битного слова из памяти	71
7.3	Описание регистров управления контроллера FLASH-памяти программ	72
7.3.1	MDR_EEPROM->CMD	73
7.3.2	MDR_EEPROM->ADR	74
7.3.3	MDR_EEPROM->DI	74
7.3.4	MDR_EEPROM->DO	74
7.3.5	MDR_EEPROM->KEY	75
7.3.6	MDR_EEPROM-> CTRL	75
8	Процессорное ядро RISC	76
8.1	Структурная схема процессорного ядра RISC	76
8.2	Программная модель	78
8.3	Стек	78
8.4	Регистры ядра	79
8.4.1	Регистры общего назначения R0-R12	80
8.4.2	Указатель стека SP R13	80
8.4.3	Регистр связи LR R14	80
8.4.4	Счетчик команд PC R15	81
8.4.5	Программный регистр состояния PSR	81
8.4.6	Программный регистр состояния приложения APSR	81

8.4.7	Программный регистр состояния прерываний IPSR	82
8.4.8	Программный регистр состояния выполнения EPSR	83
8.4.9	Регистр маски исключений Exception mask	84
8.4.10	Регистр маски приоритетов Priority Mask	84
8.4.11	Регистр маски сбоев Fault Mask	85
8.4.12	Регистр базового приоритета маски Base Priority Mask	85
8.4.13	Регистр управления CONTROL	86
8.5	Исключения и прерывания	87
9	Система команд RISC	88
9.1	Встроенные функции	92
9.2	Описание инструкций	93
9.2.1	Операнды	93
9.2.2	Ограничения на использование PC и SP	93
9.2.3	Формат второго операнда	93
9.2.4	Операции сдвига	95
9.2.5	Выравнивание адресов	97
9.2.6	Адресация относительно счетчика команд PC	98
9.2.7	Условное исполнение	98
9.2.8	Выбор размера кода инструкции	101
9.3	Команды доступа к памяти	101
9.3.1	ADR	102
9.3.2	LDR и STR, непосредственно заданное смещение	102
9.3.3	LDR и STR, смещение задано в регистре	105
9.3.4	LDR and STR, непривилегированный доступ	106
9.3.5	LDR, адресация относительно счетчика команд PC	107
9.3.6	PUSH и POP	110
9.3.7	LDREX и STREX	111
9.3.8	CLREX	113
9.4	Инструкции обработки данных	114
9.4.1	ADD, ADC, SUB, SBC и RSB	114
9.4.2	AND, ORR, EOR, BIC и ORN	117
9.4.3	ASR, LSL, LSR, ROR и RRX	118
9.4.4	CLZ	119
9.4.5	CMP и CMN	120
9.4.6	MOV и MVN	121
9.4.7	MOVT	122
9.4.8	REV, REV16, REVSH и RBIT	123
9.4.9	TST и TEQ	124
9.5	Инструкции умножения и деления	125
9.5.1	MUL, MLA и MLS	125
9.5.2	UMULL, UMLAL, SMULL и SMLAL	126
9.5.3	SDIV и UDIV	127
9.6	Инструкции преобразования данных с насыщением	128
9.6.1	SSAT и USAT	128

9.7	Команды работы с битовыми полями.....	130
9.7.1	BFC и BFI	130
9.7.2	SBFX и UBFX	131
9.7.3	SXT и UXT	132
9.8	Инструкции передачи управления	133
9.8.1	B, BL, BX и BLX.....	133
9.8.2	CBZ и CBNZ.....	135
9.8.3	IT	136
9.8.4	TBB и TBH	138
9.9	Прочие инструкции	139
9.9.1	CPS	139
9.9.2	DMB	140
9.9.3	DSB	140
9.9.4	ISB	141
9.9.5	MRS.....	141
9.9.6	MSR.....	142
9.9.7	NOP	143
9.9.8	SEV	143
9.9.9	SVC	144
9.9.10	WFE.....	144
9.9.11	WFI.....	145
10	Системный таймер SysTick.....	146
10.1	Описание регистров системного таймера SysTick	146
10.1.1	SysTick->CTRL	146
10.1.2	SysTick->LOAD	147
10.1.3	SysTick->VAL	147
10.1.4	SysTick->CAL	148
10.2	Советы и особенности при применении системного таймера	148
11	Модуль защиты памяти.....	149
11.1	Описание регистров MPU	150
11.1.1	MPU->TYPE	150
11.1.2	MPU->CTRL.....	151
11.1.3	MPU->RNR.....	152
11.1.4	MPU->RBAR	153
11.1.5	MPU->RASR	154
11.1.6	Атрибуты разрешения доступа MPU.....	155
11.1.7	Несоответствие MP	157
11.1.8	Обновление MPU региона	157
11.2	Советы и особенности применения MPU.....	159
12	Сигналы тактовой частоты	161
12.1	Встроенный RC-генератор HSI	162
12.2	Встроенный RC-генератор LSI.....	162
12.3	Внешний осциллятор HSE	162
12.4	Внешний осциллятор LSE.....	162

12.5	Встроенный блок умножения системной тактовой частоты	163
12.6	Встроенный блок умножения системной тактовой частоты DSP	163
12.7	Встроенный блок умножения тактовой частоты USB	163
12.8	Описание регистров блока контроллера тактовой частоты	163
12.8.1	MDR_RST_CLK->CLOCK_STATUS	164
12.8.2	MDR_RST_CLK->PLL_CONTROL	165
12.8.3	MDR_RST_CLK->HS_CONTROL	166
12.8.4	MDR_RST_CLK->CPU_CLOCK.....	166
12.8.5	MDR_RST_CLK->USB_CLOCK.....	167
12.8.6	MDR_RST_CLK->ADC_MCO_CLOCK.....	168
12.8.7	MDR_RST_CLK->RTC_CLOCK.....	169
12.8.8	MDR_RST_CLK->PER_CLOCK	170
12.8.9	MDR_RST_CLK->TIM_CLOCK	171
12.8.10	MDR_RST_CLK->UART_CLOCK.....	172
12.8.11	MDR_RST_CLK->SSP_CLOCK.....	173
12.8.12	MDR_RST_CLK->DSP_CLOCK	174
12.8.13	MDR_RST_CLK->SSP_CLOCK2.....	175
13	Батарейный домен и часы реального времени MDR_BKP	176
13.1	Часы реального времени	176
13.2	Регистры аварийного сохранения	177
13.3	Описание регистров блока батарейного домена.....	177
13.3.1	MDR_BKP-> REG_[00...0D]	178
13.3.2	MDR_BKP->REG_0E	178
13.3.3	MDR_BKP->REG_0F	180
13.3.4	MDR_BKP->RTC_CNT.....	183
13.3.5	MDR_BKP->RTC_DIV.....	183
13.3.6	MDR_BKP->RTC_PRL.....	183
13.3.7	MDR_BKP->RTC_ALRM	184
13.3.8	MDR_BKP->RTC_CS	184
14	Порты ввода-вывода.....	186
14.1	Описание регистров портов ввода-вывода.....	189
14.1.1	MDR_PORTx->RXTX	190
14.1.2	MDR_PORTx->OE	190
14.1.3	MDR_PORTx->FUNC.....	191
14.1.4	MDR_PORTx->ANALOG	191
14.1.5	MDR_PORTx->PULL	192
14.1.6	MDR_PORTx->PD	192
14.1.7	MDR_PORTx->PWR.....	193
14.1.8	MDR_PORTx->GFEN.....	193
14.1.9	MDR_PORTx->SET	193
14.1.10	MDR_PORTx->CLR	194
14.1.11	MDR_PORTx->RD.....	194
15	Детектор напряжения питания MDR_POWER.....	195
15.1	MDR_POWER->PVDCS.....	195

16	Внешняя системная шина MDR_EBC	198
16.1	Работа с внешними статическими ОЗУ, ПЗУ и периферийными устройствами	198
16.2	Работа с внешней NAND FLASH-памятью.....	200
16.3	Описание регистров блока контроллера внешней системной шины	204
16.3.1	MDR_EBC->NAND_CYCLES.....	205
16.3.2	MDR_EBC->CONTROL.....	205
17	Контроллер прямого доступа в память MDR_DMA	207
17.1	Основные свойства контроллера DMA	207
17.2	Термины и определения.....	208
17.3	Функциональное описание	210
17.3.1	Распределение каналов DMA	210
17.3.2	Блок, подключенный к шине APB	211
17.3.3	Блок, подключенный к шине AHB	211
17.3.4	Управляющий блок DMA	211
17.3.5	Типы передач	211
17.3.6	Разрядность передач данных	212
17.3.7	Управление защитой данных	212
17.3.8	Инкремент адреса	213
17.4	Управление DMA.....	213
17.4.1	Правила обмена данными.....	213
17.4.2	Диаграммы работы контроллера DMA	216
17.4.3	Правила арбитража DMA	222
17.4.4	Приоритет.....	222
17.4.5	Типы циклов DMA	224
17.4.6	Индикация ошибок.....	236
17.5	Структура управляющих данных канала	236
17.6	Описание регистров контроллера DMA	247
17.6.1	MDR_DMA->STATUS.....	249
17.6.2	MDR_DMA->CFG	250
17.6.3	MDR_DMA->CTRL_BASE_PTR	251
17.6.4	MDR_DMA->ALT_CTRL_BASE_PTR	252
17.6.5	MDR_DMA->WAITONREQ_STATUS.....	252
17.6.6	MDR_DMA->CHNL_SW_REQUEST	253
17.6.7	MDR_DMA->CHNL_USEBURST_SET.....	253
17.6.8	MDR_DMA->CHNL_USEBURST_CLR.....	254
17.6.9	MDR_DMA->CHNL_REQ_MASK_SET	255
17.6.10	MDR_DMA->CHNL_REQ_MASK_CLR.....	256
17.6.11	MDR_DMA->CHNL_ENABLE_SET	256
17.6.12	MDR_DMA->CHNL_ENABLE_CLR.....	257
17.6.13	MDR_DMA->CHNL_PRI_ALT_SET	258
17.6.14	MDR_DMA->CHNL_PRI_ALT_CLR	259
17.6.15	DFMDR_DMA->CHNL_PRIORITY_SET	260
17.6.16	MDR_DMA->CHNL_PRIORITY_CLR.....	261

17.6.17	MDR_DMA->ERR_CLR.....	261
18	Организация управления подсистемой DSP	263
18.1	Средства управления подсистемой DSP.....	263
18.2	Синхросигналы подсистемы DSP	264
18.2.1	Режимы отключения синхросигналов подсистемы DSP (IDLE1, IDLE2, IDLE3).....	265
18.3	Сбросы подсистемы DSP	266
18.4	Прерывания и запросы DMA между подсистемами.....	266
18.5	Специальные возможности управления	268
18.6	Работа моста пересинхронизации RISC – DSP	268
18.7	Способы управления подсистемой DSP	269
18.7.1	Нормальная работа DSP.....	269
18.7.2	Динамическое изменение программного обеспечения ядра DSP	270
18.7.3	Работа с периферийными модулями и памятью DSP без участия ядра DSP	271
18.8	Регистры управления подсистемой DSP	271
18.8.1	Регистр управления синхронизацией CLKMD.....	272
18.8.2	Регистр прерываний от DSP к RISC	274
19	Процессорное ядро DSP	276
19.1	Основные особенности ядра DSP.....	276
19.2	Архитектура ядра DSP	276
19.2.1	Шинная структура DSP.....	278
19.2.2	Устройство управления выполнением программ.....	278
19.2.3	Арифметико-логическое устройство.....	278
19.2.4	Аккумуляторы А и В	280
19.2.5	Циклическое сдвиговое устройство	283
19.2.6	Устройство умножения/сложения	285
19.2.7	Устройство сравнения, выбора и хранения (CSSU).....	289
19.2.8	Шифратор показателя	290
19.3	Организация конвейера	291
19.3.1	Стадии конвейера	291
19.3.2	Инструкции перехода в конвейер	294
19.3.3	Инструкции вызова функций в конвейере	296
19.3.4	Инструкции возврата в конвейере	297
19.3.5	Условная инструкция в конвейере.....	301
19.3.6	Команды условного вызова и условного перехода в конвейере	302
19.3.7	Прерывания и конвейер	305
19.4	Адресация данных	306
19.4.1	Непосредственная адресация	306
19.4.2	Абсолютная адресация.....	307
19.4.3	Аккумуляторная адресация	308
19.4.4	Прямая адресация	309
19.4.5	Косвенная адресация	311
19.4.6	Адресация регистра отображенного в памяти.....	323

19.4.7	Стековая адресация	324
19.4.8	Типы Данных	325
19.5	Адресация программ	326
19.5.1	Генерация адреса программной памяти	327
19.5.2	Программный Счетчик (PC)	328
19.5.3	Ветвления	329
19.5.4	Вызов процедур	331
19.5.5	Возвраты	333
19.5.6	Условные операции	335
19.6	Функционирование ядра DSP после сброса	341
19.7	Прерывания DSP	341
19.7.1	Регистры управления прерываниями ядра	342
19.7.2	Регистр Маски Прерывания (IMR-interrupt mask register)	344
19.7.3	Обработка прерываний	345
19.8	Регистры управления и состояния ядра	351
19.8.1	Регистры Статуса (ST0 и ST1)	351
20	Система команд DSP	358
20.1	Обзор набора команд	358
20.1.1	Арифметические операции	358
20.1.2	Логические операции	362
20.1.3	Команды управления программой	364
20.1.4	Команды загрузки и сохранения	367
20.1.5	Повторения одной команды	370
20.2	Классы и циклы команд	372
20.3	Команды на языке ассемблера	411
20.3.1	Обозначения и сокращения	411
20.3.2	ABDST Xmem, Ymem	417
20.3.3	ABS src [, dst]	418
20.3.4	ADD	418
20.3.5	ADDC Smem, src	422
20.3.6	ADDM #lk, Smem	422
20.3.7	ADDS Smem, src	423
20.3.8	AND	424
20.3.9	ANDM #lk, Smem	425
20.3.10	B[D] pmad	426
20.3.11	BACC[D] src	427
20.3.12	BANZ[D] pmad, Sind	428
20.3.13	BC[D] pmad, cond [, cond [, cond]]	429
20.3.14	BIT Xmem, BITC	431
20.3.15	BITF Smem, #lk	432
20.3.16	BITT Smem	433
20.3.17	CALA[D] src	434
20.3.18	CALL[D] pmad	435
20.3.19	CC[D] pmad, cond [, cond [, cond]]	436

20.3.20	CMPL src [, dst]	438
20.3.21	CMPM Smem, #lk.....	438
20.3.22	CMPR CC, ARx	439
20.3.23	CMPS src, Smem.....	440
20.3.24	DADD Lmem, src [, dst]	441
20.3.25	DADST Lmem, dst.....	443
20.3.26	DELAY Smem	444
20.3.27	DLD Lmem, dst.....	445
20.3.28	DRSUB Lmem, src.....	446
20.3.29	DSADT Lmem, dst.....	447
20.3.30	DST src, Lmem	448
20.3.31	DSUB Lmem, src	449
20.3.32	DSUBT Lmem, dst.....	451
20.3.33	EXP src	452
20.3.34	FB[D] extpmad.....	453
20.3.35	FBACC[D] src.....	453
20.3.36	FCALA[D] src.....	454
20.3.37	FCALL[D] extpmad	456
20.3.38	FIRS Xmem, Ymem, pmad.....	457
20.3.39	FRAME K	458
20.3.40	FRET[D].....	459
20.3.41	FRETE[D]	459
20.3.42	IDLE K	460
20.3.43	INTR K.....	461
20.3.44	LD	462
20.3.45	LDM MMR, dst.....	466
20.3.46	LD Xmem, dst MAC[R] Ymem [, dst_]	467
20.3.47	LD Xmem, dst MAS[R] Ymem [, dst_]	468
20.3.48	LDR Smem, dst	469
20.3.49	LDU Smem, dst.....	469
20.3.50	LMS Xmem, Ymem	470
20.3.51	LTD Smem	471
20.3.52	MAC[R].....	471
20.3.53	MACA[R].....	474
20.3.54	MACD Smem, pmad, src	476
20.3.55	MACP Smem, pmad, src	477
20.3.56	MACSU Xmem, Ymem, src	478
20.3.57	MAR Smem.....	479
20.3.58	MAS[R]	481
20.3.59	MASA.....	483
20.3.60	MAX dst	484
20.3.61	MIN dst.....	485
20.3.62	MPY[R]	485
20.3.63	MPYA.....	487

20.3.64	MPYU Smem, dst.....	489
20.3.65	MVDD Xmem, Ymem	489
20.3.66	MVDK Smem, dmad.....	490
20.3.67	MVDM dmad, MMR.....	491
20.3.68	MVDP Smem, pmad	492
20.3.69	MVKD dmad, Smem.....	493
20.3.70	MVMD MMR, dmad.....	494
20.3.71	MVMM MMRx, MMRy	494
20.3.72	MVPD pmad, Smem	495
20.3.73	NEG src [, dst].....	496
20.3.74	NOP	497
20.3.75	NORM src [, dst]	498
20.3.76	OR.....	498
20.3.77	ORM #lk, Smem.....	500
20.3.78	POLY Smem	501
20.3.79	POPD Smem.....	501
20.3.80	POPM MMR	502
20.3.81	PORTR PA, Smem.....	503
20.3.82	PORTW Smem, PA.....	503
20.3.83	PSHD Smem.....	504
20.3.84	PSHM MMR	505
20.3.85	RC[D] cond [, cond [, cond]].....	505
20.3.86	READA Smem	507
20.3.87	RESET	508
20.3.88	RET[D]	509
20.3.89	RETE[D].....	509
20.3.90	RETF[D].....	510
20.3.91	RND src [, dst].....	511
20.3.92	ROL src.....	511
20.3.93	ROLTC src	512
20.3.94	ROR src	512
20.3.95	RPT	513
20.3.96	RPTB[D] pmad.....	514
20.3.97	RPTZ dst, #lk.....	515
20.3.98	RSBX N, SBIT	516
20.3.99	SACCD src, Xmem, cond	516
20.3.100	SAT src	517
20.3.101	SFTA src, SHIFT [, dst]	518
20.3.102	SFTC src	519
20.3.103	SFTL src, SHIFT [, dst].....	520
20.3.104	SQDST Xmem, Ymem.....	521
20.3.105	SQURA Smem, src.....	523
20.3.106	SQURS Smem, src	524
20.3.107	SRCCD Xmem, cond	525

20.3.108	SSBX N, SBIT.....	526
20.3.109	ST.....	526
20.3.110	STH.....	527
20.3.111	STL	529
20.3.112	STLM src, MMR.....	531
20.3.113	STM #lk, MMR.....	532
20.3.114	ST src, Ymem ADD Xmem, dst	533
20.3.115	ST LD.....	534
20.3.116	ST src, Ymem	535
20.3.117	ST src, Ymem MAS[R] Xmem, dst.....	536
20.3.118	ST src, Ymem MPY Xmem, dst	538
20.3.119	ST src, Ymem SUB Xmem, dst	538
20.3.120	STRCD Xmem, cond.....	539
20.3.121	SUB	540
20.3.122	SUBB Smem, src.....	544
20.3.123	SUBC Smem, src.....	545
20.3.124	SUBS Smem, src	546
20.3.125	TRAP K	547
20.3.126	WRITA Smem.....	547
20.3.127	XC n, cond [, cond [, cond]].....	548
20.3.128	XOR.....	550
20.3.129	XORM #lk, Smem.....	552
21	Контроллер McBSP (DSP).....	553
21.1	Общее описание McBSP.....	553
21.2	Регистры McBSP	554
21.2.1	McBSP1.....	554
21.2.2	McBSP2.....	555
21.2.3	McBSP3.....	556
21.3	Конфигурирование последовательного порта	557
21.3.1	SPCRH.....	557
21.3.2	SPCRL	558
21.3.3	SPSRH	560
21.3.4	PCRL	561
21.4	Управление приемом и передачей	562
21.4.1	RCRH, XCRH	562
21.4.2	RCRL, XCRL	563
21.5	Порядок приема и передачи данных.....	564
21.6	Сброс последовательного порта.....	564
21.7	Программный внутренний сброс (биты управления регистра SPCR).....	564
21.8	Обработка статусов.....	565
21.9	Состояние приемника: RRDY, RRINT, RFULL, RSYNCERR.....	565
21.10	Состояние передатчика	565
21.11	События и прерывания.....	565
21.12	Настройка битовой и кадровой синхронизации	566

21.12.1	Фазы кадровой синхронизации	567
21.12.2	Упаковка данных заданием размеров кадра и слова.	567
21.13	Задержка данных.....	568
21.14	Пример многофазного кадра (AC97)	570
21.15	Штатный режим работы McBSP	570
21.16	Режим игнорирования кадровой синхронизации	573
21.17	Упаковка данных при помощи режима пропуска кадровой синхронизации .	573
21.18	Пропуск нежелательных кадров при помощи режима пропуска кадровой синхронизации	574
21.19	Особые ситуации работы последовательного порта.....	575
21.20	Переполнение приемника (RFULL).....	575
21.21	Перезапись передаваемых данных.....	577
21.22	Выравнивание данных и расширение знака.....	578
21.23	Кодирование/декодирование данных	578
21.24	Компактирование/декомпактирование по u- и A-законам	579
21.25	Кодирование/декодирование кодом Манчестер-2.....	579
21.26	Порядок передачи бит в словах.....	580
21.27	Программируемые кадровая и битовая синхронизации	580
21.28	Внутренний генератора кадровой и битовой синхронизации.....	580
21.28.1	SRGRH.....	581
21.28.2	SRGRL	582
21.29	Сброс генератора кадровой и битовой синхронизации	582
21.30	Генератор битовой синхронизации.....	583
21.31	Полярность тактового сигнала генератора	583
21.32	Битовая и кадровая синхронизация	583
21.33	Режим КЗ	584
21.34	Генератор кадровой синхронизации	585
21.35	Период и длина кадрового синхроимпульса.....	585
21.36	Примеры битовой и кадровой синхронизации в различных форматах.....	586
21.36.1	ST-BUS	586
21.36.2	ST-BUS с удвоенной частотой	586
21.37	Работа в многоканальном режиме с выбором каналов	587
21.37.1	MCRH, MCRL, XMCR, RMCR.....	588
21.38	Включение многоканального режима	588
21.39	Регистр управления активностью каналов	589
21.39.1	XCERH, RCERH	589
21.39.2	XCERL, RCERL	590
21.39.3	Интерфейс A-bis	590
22	Системный Таймер (DSP)	591
22.1	Регистры таймера (DSP).....	591
22.1.1	Регистры таймера (TIM)	591
22.1.2	Регистр периода таймера (PRD).....	591
22.1.3	Управляющий регистр таймера (TCR).....	592
23	Работа таймера (DSP)	593

24	Контроллер DMA (DSP).....	594
24.1	Особенности контроллера DMA (DSP)	594
24.2	Возможности доступа DMA (DSP)	594
24.3	Регистры контроллера DMA(DSP).....	595
24.4	Распределение аппаратных запросов по каналам DMA(DSP)	596
25	Контроллер AudioCodec (DSP).....	597
25.1	Задание рабочей частоты аудиокодека	597
25.2	Архитектура модуля	597
25.3	Аналоговые площадки ввода-вывода	599
25.3.1	Вход микрофона	600
25.3.2	Входы INP и INM	600
25.3.3	Аналоговые выходы	601
25.4	Работа с модулем	601
25.4.1	Воспроизведение звука (ЦАП).....	601
25.4.2	Оцифровка входного потока (АЦП)	602
25.4.3	Запросы DMA	602
25.4.4	Прерывания модуля.....	602
25.5	Регистры управления.....	603
25.5.1	Регистр общего управления кодеком	603
25.5.2	Регистр управления АЦП	603
25.5.3	Регистр управления ЦАП	604
25.5.4	Флаги прерываний.....	606
26	Контроллер блока шифрования ГОСТ 28147-89 (DSP).....	607
26.1	Описание регистров.....	607
26.1.1	Регистр управления CRPT_CWR.....	609
26.1.2	Регистр состояния CRPT_SR.....	610
26.1.3	Регистр данных шифрации CRPT_DATA.....	610
26.1.4	Регистр ключа шифрации CRPT_KR	611
26.1.5	Регистр синхропосылки CRPT_SYNR	611
26.1.6	Регистр констант замены CRPT_CR.....	612
26.1.7	Регистр имитовставки CRPT_IMIT	612
26.1.8	Регистр числа итераций CRPT_ITER	613
26.2	Работа блока по шифрованию и дешифрованию данных.....	613
26.2.1	Шифрование данных. Режим простой замены	613
26.2.2	Дешифрование данных. Режим простой замены	614
26.2.3	Шифрование данных. Режим гаммирования	614
26.2.4	Дешифрование данных. Режим гаммирования	615
26.2.5	Шифрование данных. Режим гаммирования с обратной связью	615
26.2.6	Дешифрование данных. Режим гаммирования с обратной связью	616
26.2.7	Режим самопроверки.....	616
26.2.8	Порядок занесения констант замены и ключа в соответствии с ГОСТ Р 34.11-94	616
27	Контроллер интерфейса MDR_USB	618
27.1	Инициализация контроллера при включении.....	618

27.2	Задание параметров шины USB и события подключения/ отключения	619
27.3	Задание адреса и инициализация оконечных точек	619
27.4	Транзакция IN (USB Device).....	619
27.5	Транзакция SETUP/OUT (USB Device)	621
27.6	Транзакция SETUP/OUT (USB Host)	623
27.7	Транзакция IN (USB Host)	623
27.8	Отправка SOF пакетов и отсчет времени (USB Host)	624
27.9	Описание регистров управления контроллером интерфейса USB	625
27.9.1	MDR_USB->HSCR	627
27.9.2	MDR_USB->HSVR	627
27.9.3	MDR_USB->HTXC	628
27.9.4	MDR_USB->HTXT	628
27.9.5	MDR_USB->HTXLC	629
27.9.6	MDR_USB->HTXSE	629
27.9.7	MDR_USB->HTXA	630
27.9.8	MDR_USB->HTXE	630
27.9.9	MDR_USB->HFN	631
27.9.10	MDR_USB->HSI	631
27.9.11	MDR_USB->HIM	631
27.9.12	MDR_USB->HRXS	632
27.9.13	MDR_USB->HRXP	633
27.9.14	MDR_USB->HRXA	633
27.9.15	MDR_USB->HRXE	633
27.9.16	MDR_USB->HRXCS	633
27.9.17	MDR_USB->HSTM	634
27.9.18	MDR_USB->HRXFD	634
27.9.19	MDR_USB->HRXFDC	635
27.9.20	MDR_USB->HRXFC	635
27.9.21	MDR_USB->HTXFD	635
27.9.22	MDR_USB->HTXFC	636
27.9.23	MDR_USB->SEP[x].CTRL	636
27.9.24	MDR_USB->SEP[x].STS	637
27.9.25	MDR_USB->SEP[x].TS	637
27.9.26	MDR_USB->SEP[x].NTS	638
27.9.27	MDR_USB->SC	638
27.9.28	MDR_USB->SLS	639
27.9.29	MDR_USB->SIS	639
27.9.30	MDR_USB->SIM	640
27.9.31	MDR_USB->SA	641
27.9.32	MDR_USB->SFN	641
27.9.33	MDR_USB->SEP[x].RXFD	641
27.9.34	MDR_USB->SEP[x].RXFDC	642
27.9.35	MDR_USB->SEP[x].RXFC	642
27.9.36	MDR_USB->SEP[x].TXFD	642

27.9.37	MDR_USB->SEP[x].TXFDC	643
28	Таймеры общего назначения MDR_TIMERx	644
28.1	Основные характеристики	644
28.1.1	Структурная схема	645
28.2	Базовый таймер счета	646
28.2.1	Инициализация тактирования таймера	646
28.2.2	Инициализация основного счетчика таймера.....	646
28.3	Режимы счета	647
28.3.1	Счет прямой: CNT_MODE[1:0] = 00, DIR = 0	647
28.3.2	Счет обратный: CNT_MODE[1:0] = 00, DIR = 1	647
28.3.3	Счет двунаправленный: CNT_MODE = 01, DIR = 0	648
28.3.4	Счет двунаправленный: CNT_MODE = 01, DIR = 1	648
28.3.5	Тактовая частота F _{DTS}	649
28.4	Источники событий для счета	650
28.4.1	Внутренний тактовый сигнал (TIM_CLKd).....	651
28.4.2	Событие в другом таймере (CNT==ARR).....	652
28.4.3	Внешний тактовый сигнал, «Режим 1»: событие переднего фронта на входе канала CНуi	653
28.4.4	Внешний тактовый сигнал, «Режим 2»: событие переднего или заднего фронта на входе ETR.....	654
28.5	Режим захвата	655
28.6	Режим ШИМ.....	656
28.6.1	Генератор опорного сигнала REF	657
28.6.2	Генератор «мертвой зоны»	658
28.6.3	Выходные блоки	659
28.7	Блок цифрового фильтра.....	660
28.8	Флаги состояний, прерывания и запросы DMA	662
28.8.1	Флаги состояний	662
28.8.2	Прерывания	662
28.8.3	Запросы DMA	662
28.9	Примеры	662
28.9.1	Обычный счетчик	663
28.9.2	Режим захвата	663
28.9.3	Режим ШИМ	664
28.10	Описание регистров блока таймера	665
28.10.1	MDR_TIMERx->CNT	667
28.10.2	MDR_TIMERx->PSG.....	667
28.10.3	MDR_TIMERx->ARR.....	668
28.10.4	MDR_TIMERx->CNTRL	668
28.10.5	MDR_TIMERx->CCRY	669
28.10.6	MDR_TIMERx->CCRY1	670
28.10.7	MDR_TIMERx->CHy_CNTRL	670
28.10.8	MDR_TIMERx->CHy_CNTRL1	672
28.10.9	MDR_TIMERx->CHy_CNTRL2	673

28.10.10	MDR_TIMERx->CHy_DTG	674
28.10.11	MDR_TIMERx->BRKETR_CNTRL	674
28.10.12	MDR_TIMERx->STATUS	675
28.10.13	MDR_TIMERx->IE	677
28.10.14	MDR_TIMERx->DMA_RE	678
29	Контроллер MDR_ADC	680
29.1	Преобразование внешнего канала.....	681
29.2	Последовательное преобразование нескольких каналов.....	681
29.3	Преобразование с контролем границ.....	682
29.4	Внутренний источник опорного напряжения	682
29.5	Датчик температуры.....	683
29.6	Синхронный запуск двух АЦП.....	683
29.7	Время заряда внутренней емкости	684
29.8	Описание регистров блока контроллера АЦП.....	685
29.8.1	MDR_ADC->ADC1_CFG.....	686
29.8.2	MDR_ADC->ADC2_CFG.....	688
29.8.3	MDR_ADC->ADCx_H_LEVEL	690
29.8.4	MDR_ADC->ADCx_L_LEVEL	690
29.8.5	MDR_ADC->ADCx_RESULT	690
29.8.6	MDR_ADC->ADCx_STATUS	691
29.8.7	MDR_ADC->ADCx_CHSEL	692
30	Контроллер MDR_DAC	693
30.1	Описание регистров блока контроллера ЦАП.....	693
30.1.1	MDR_DAC->CFG	694
30.1.2	MDR_DAC->DAC1_DATA	694
30.1.3	MDR_DAC->DAC2_DATA	695
31	Контроллер схемы компаратора MDR_COMP	696
31.1	Сравнение внешних сигналов.....	696
31.2	Сравнение сигнала с внутренним источником опорного напряжения	697
31.3	Сравнение внешних сигналов с внутренней шкалой напряжений	697
31.4	Формирование внутренней шкалы напряжений.....	697
31.5	Описание регистров блока контроллера компаратора.....	699
31.5.1	MDR_COMP->CFG	699
31.5.2	MDR_COMP->RESULT	700
31.5.3	MDR_COMP->RESULT_LATCH.....	700
32	Контроллер интерфейса MDR_I2C	701
32.1	Конфигурация системы.....	701
32.2	Протокол I2C	701
32.3	Сигнал START	702
32.4	Передача адреса	702
32.5	Передача данных.....	702
32.6	Сигнал STOP	703
32.7	Описание регистров контроллера I2C	703
32.7.1	MDR_I2C->PRL	703

32.7.2	MDR_I2C->PRH.....	704
32.7.3	MDR_I2C->CTR.....	704
32.7.4	MDR_I2C->RXD	704
32.7.5	MDR_I2C->STA.....	705
32.7.6	MDR_I2C->TXD	706
32.7.7	MDR_I2C->CMD	706
33	Контроллер MDR_SSP	707
33.1	Основные характеристики модуля SSP	707
33.2	Программируемые параметры.....	708
33.3	Характеристики интерфейса SPI.....	708
33.4	Характеристики интерфейса Microwire.....	709
33.5	Характеристики интерфейса SSI.....	709
33.6	Общий обзор модуля SSP	709
33.6.1	Блок формирования тактового сигнала.....	710
33.6.2	Буфер FIFO передатчика.....	710
33.6.3	Буфер FIFO приемника	710
33.6.4	Блок приема и передачи данных.....	710
33.6.5	Блок формирования прерываний	711
33.6.6	Интерфейс прямого доступа к памяти	711
33.6.7	Конфигурирование приемопередатчика	711
33.6.8	Разрешение работы приемопередатчика.....	711
33.6.9	Соотношения между тактовыми сигналами.....	712
33.6.10	Программирование регистра управления CR0.....	713
33.6.11	Программирование регистра управления CR1	713
33.6.12	Формирование тактового сигнала обмена данными.....	713
33.6.13	Формат информационного кадра.....	714
33.6.14	Формат синхронного обмена SSI фирмы Texas Instruments	715
33.6.15	Формат синхронного обмена SPI фирмы Motorola.....	715
33.6.16	Формат синхронного обмена SPI фирмы Motorola, SPO=0, SPH=0.	716
33.6.17	Формат синхронного обмена SPI фирмы Motorola, SPO=0, SPH=1.	717
33.6.18	Формат синхронного обмена SPI фирмы Motorola, SPO=1, SPH=0.	718
33.6.19	Формат синхронного обмена SPI фирмы Motorola, SPO=1, SPH=1.	719
33.6.20	Формат синхронного обмена Microwire фирмы National Semiconductor	720
33.6.21	Примеры конфигурации модуля в ведущем и ведомом режимах	722
33.6.22	Интерфейс прямого доступа к памяти	724
33.7	Программное управление модулем.....	726
33.7.1	Общая информация	726
33.7.2	Описание регистров контроллера SSP	726
33.8	Прерывания	733
33.8.1	SSPRXINTR.....	733
33.8.2	SSPTXINTR.....	733
33.8.3	SSPRORINTR.....	734
33.8.4	SSPRTINTR.....	734

33.8.5	SSPINTR	734
34	Контроллер MDR_UART	735
34.1	Основные сведения	735
34.1.1	Основные характеристики модуля UART	735
34.1.2	Программируемые параметры	736
34.1.3	Отличия от контроллера UART 16C650	736
34.2	Функциональные возможности	737
34.3	Описание функционирования блока UART	738
34.3.1	Генератор тактового сигнала приемопередатчика	738
34.3.2	Буфер FIFO передатчика	739
34.3.3	Буфер FIFO приемника	739
34.3.4	Блок передатчика	739
34.3.5	Блок приемника	739
34.3.6	Блок формирования прерываний	739
34.3.7	Интерфейс прямого доступа к памяти	740
34.3.8	Блок и регистры синхронизации	740
34.4	Описание функционирования ИК-кодека IrDA SIR	740
34.4.1	Кодер ИК-передатчика	740
34.4.2	Декодер ИК-приемника	741
34.5	Описание работы UART	741
34.5.1	Сброс модуля	741
34.5.2	Тактовые сигналы	741
34.5.3	Работа универсального асинхронного приемопередатчика	742
34.5.4	Коэффициент деления частоты	742
34.5.5	Передача и прием данных	743
34.5.6	Биты ошибки	744
34.5.7	Бит переполнения буфера	744
34.5.8	Запрет буфера FIFO	744
34.5.9	Работа кодека ИК-обмена данными IrDA SIR	745
34.5.10	Модуляция данных IrDA	746
34.6	Интерфейс прямого доступа к памяти	747
34.7	Прерывания	749
34.7.1	UARTRXINTR	750
34.7.2	UARTTXINTR	750
34.7.3	UARTRTINTR	750
34.7.4	UARTEINTR	751
34.7.5	UARTINTR	751
34.8	Программное управление модулем	751
34.8.1	Общая информация	751
34.8.2	Обобщенные данные о регистрах устройства	752
35	Контроллер SDIO	767
35.1	Описание функционирования контроллера	767
35.1.1	Передача команды (по спецификации SDIO)	767
35.1.2	Прием команды (по спецификации SDIO)	768

35.1.3	Передача данных (по спецификации SDIO)	769
35.1.4	Прием данных (по спецификации SDIO)	771
35.2	Прерывания	771
35.3	Схемы включения контроллера.....	772
35.3.1	Подключение SD card к контроллеру.....	772
35.4	Регистры SD	772
35.4.1	Регистр управления	773
35.4.2	Регистр состояния.....	776
35.4.3	Регистр команд	776
35.4.4	Регистр данных	777
35.4.5	Регистр контрольной суммы линии команд	777
35.4.6	Регистры контрольных сумм линий данных	777
35.4.7	Регистр-счетчик принимаемых/ передаваемых бит линии команд ..	778
35.4.8	Регистр-счетчик принимаемых/ передаваемых бит линий данных .	778
36	Прерывания и исключения RISC	780
36.1	Типы исключений	780
36.1.1	RESET	780
36.1.2	NON MASKABLE INTERRUPT (NMI).....	780
36.1.3	Hard Fault	781
36.1.4	Memory Management fault	781
36.1.5	Bus Fault.....	781
36.1.6	Usage Fault	781
36.1.7	SVCall	781
36.1.8	PendSV	781
36.1.9	SysTick	781
36.2	Прерывания (IRQ).....	782
36.3	Обработчики исключений.....	782
36.4	Таблица векторов	783
36.5	Приоритеты исключений	783
36.5.1	Группировка приоритетов прерываний	784
36.6	Вход в обработчик и выход из обработчика	784
36.6.1	Приоритетное прерывание	784
36.6.2	Возврат.....	784
36.6.3	Передача управления без восстановления контекста (tail-chaining)	785
36.6.4	Запоздавшее исключение (late-arriving exception)	785
36.6.5	Вход в процедуру обработки исключения	785
36.6.6	Возврат из обработчика исключения	786
36.7	Обработка отказов	787
36.7.1	Типы отказов	787
36.7.2	Эскалация отказов и аппаратные отказы	788
36.7.3	Регистры состояния и адреса отказа.....	789
36.7.4	Блокировка	789
36.8	Управление электропитанием	789
36.8.1	Переход в режим пониженного энергопотребления.....	790

36.8.2	Ожидание прерывания	790
36.8.3	Ожидание события	790
36.8.4	Переход в режим ожидания по выходу из обработчика исключения (режим Sleep)	790
36.8.5	Выход из состояния ожидания	790
36.8.6	Рекомендации по программированию режима энергопотребления	791
37	Контроллер прерываний NVIC (RISC)	793
37.1	Логика работы прерываний контроллера NVIC.	794
37.2	Упрощенный доступ к регистрам контроллера прерываний	797
37.2.1	NVIC->ISER[x]	798
37.2.2	NVIC->ICER[x]	798
37.2.3	NVIC->ISPR[x]	799
37.2.4	NVIC->ICPR[x]	799
37.2.5	NVIC->IABR[x]	800
37.2.6	NVIC->IP[x]	800
37.2.7	NVIC->STIR	801
37.3	Прерывания, срабатывающие по уровню сигнала	802
37.4	Аппаратное и программное управление прерываниями	802
37.5	Рекомендации по работе с контроллером прерываний	803
38	Блок управления системой RISC	804
38.1	Упрощенный доступ к регистрам блока управления системой	805
38.1.1	InterrupType->ACTLR	805
38.1.2	SCB->CPUID	806
38.1.3	SCB->ICSR	806
38.1.4	SCB->VTOR	808
38.1.5	SCB->AIRCRR	809
38.1.6	SCB->SCR	810
38.1.7	SCB->CCR	811
38.1.8	SCB->SHP[x]	813
38.1.9	SCB->SHCSR	814
38.1.10	SCB->CFSR	816
38.1.11	SCB->HFSR	821
38.1.12	SCB->MMFAR	821
38.1.13	SCB->BFAR	822
39	Сторожевые таймеры	824
39.1	Описание регистров блока сторожевых таймеров	824
39.1.1	Блок сторожевого таймера IWDG	824
39.1.2	Блок сторожевого таймера WWDG	829
40	Типовая схема включения	832
41	Типовые зависимости	833
42	Электрические параметры микросхемы	835
43	Предельные и предельно-допустимые режимы работы	838
44	Справочные данные	841
45	Габаритный чертеж микросхемы	844

46 Информация для заказа 845

1 Введение

Двухъядерный микропроцессор является системой, основанной на двух вычислительных ядрах: RISC (32 разряда) и DSP (аналог TMS320C54). В микросхеме реализована встроенная FLASH-память программ, размером 128 Кбайт, ОЗУ RISC размером 32 Кбайт и две ОЗУ DSP (память программ размером 128 Кбайт и память данных размером 128 Кбайт). Ядро RISC работает на тактовой частоте до 100 МГц, ядро DSP работает на частоте до 100 МГц. Набор интерфейсных периферийных модулей включает в себя контроллер интерфейса USB со встроенным аналоговым приемопередатчиком (12 Мбит/с Full Speed, либо 1,5 Мбит/с Low Speed), стандартные интерфейсы UART(×3), SPI(×4), I2C(×1) и McBSP(×3). Контроллер внешней системной шины позволяет работать с внешними микросхемами статического ОЗУ и ПЗУ, FLASH-памятью и другими периферийными устройствами. Также в микросхеме имеется интерфейс SDIO. Микросхемы содержат три 16-разрядных таймера с четырьмя каналами схем захвата и интегрированными модулями ШИМ с функциями формирования «мертвой зоны» и аппаратной блокировки, системный 24-разрядный таймер (RISC), системный таймер (DSP) и два сторожевых таймера. Для работы с аналоговыми устройствами микросхема имеет два 12-разрядных высокоскоростных (до 0,5 Мвыб/с) АЦП с возможностью оцифровки информации из 16 каналов, встроенные датчики температуры и опорного напряжения, 12-разрядный ЦАП и схему встроенного компаратора с тремя входами и внутренней шкалой напряжений.

Встроенные RC-генераторы HSI (8 МГц) и LSI (40 кГц) и внешние генераторы HSE (2...16 МГц) и LSE (32 кГц) и три схемы умножения тактовой частоты PLL отдельно для ядра RISC, интерфейса USB и всей DSP части устройства позволяют гибко настраивать скорость работы микроконтроллера.

Архитектура доступа к памяти и периферийным устройствам при помощи матрицы системных шин позволяет минимизировать возможные конфликты при работе системы и повысить общую производительность. В систему включены два контроллера DMA. Контроллер DMA RISC части позволяет организовать обмен между любыми блоками памяти и периферийными устройствами (включая устройства домена DSP) без участия процессорного ядра RISC. Также в системе имеется отдельный контроллер DMA для DSP части, который может управляться как при помощи ядра RISC, так и при помощи ядра DSP и имеет доступ ко всему домену DSP (память программ DSP, память данных DSP, таймер DSP, блоки McBSP, модуль шифрования и аудиокодек).

В систему включен внутренний регулятор напряжения питания для цифровой части на 1,8 В. Таким образом, для питания микросхемы достаточно подать внешнее напряжение в диапазоне от 2,2 до 3,6 В.

Кроме того, в микроконтроллере реализован батарейный домен для хранения основных настроек системы и часы реального времени, которые могут работать от альтернативного источника питания (внешней батареи) в случае отсутствия основного питания. Встроенные детекторы напряжения питания могут отслеживать уровни напряжения внешнего основного и батарейного питания. Аппаратные схемы сброса по снижению питания позволяют исключить сбойную работу микросхемы при выходе уровня напряжения питания за допустимые пределы.

2 Условное графическое обозначение

144	PA[0]	MPU	PD[0]	74
143	PA[1]		PD[1]	76
142	PA[2]		PD[2]	79
141	PA[3]		PD[3]	81
140	PA[4]		PD[4]	80
139	PA[5]		PD[5]	82
138	PA[6]		PD[6]	83
137	PA[7]		PD[7]	77
136	PA[8]		PD[8]	78
135	PA[9]		PD[9]	84
134	PA[10]		PD[10]	75
133	PA[11]		PD[11]	72
132	PA[12]		PD[12]	71
131	PA[13]		PD[13]	70
130	PA[14]		PD[14]	69
129	PA[15]		PD[15]	68
125	PB[0]		PE[0]	64
124	PB[1]		PE[1]	63
123	PB[2]		PE[2]	55
122	PB[3]		PE[3]	54
121	PB[4]		PE[4]	57
120	PB[5]		PE[5]	56
119	PB[6]		PE[6]	39
118	PB[7]		PE[7]	38
117	PB[8]		PE[8]	53
116	PB[9]		PE[9]	62
115	PB[10]		PE[10]	61
109	PB[11]		PE[11]	25
107	PB[12]		PE[12]	23
106	PB[13]		PE[13]	22
104	PB[14]		PE[14]	52
105	PB[15]		PE[15]	21
101	PC[0]		PF[0]	5
103	PC[1]		PF[1]	6
100	PC[2]		PF[2]	7
102	PC[3]		PF[3]	8
98	PC[4]		PF[4]	9
99	PC[5]		PF[5]	10
96	PC[6]		PF[6]	11
97	PC[7]		PF[7]	12
94	PC[8]		PF[8]	13
95	PC[9]		PF[9]	14
92	PC[10]		PF[10]	15
93	PC[11]		PF[11]	16
91	PC[12]		PF[12]	17
90	PC[13]		PF[13]	18
89	PC[14]		PF[14]	19
88	PC[15]		PF[15]	20
34	OSC_IN		OSC_OUT	37
47	INP1		OUTP1	42
48	INM1		OUTM1	43
49	INP2		BIAS	44
50	INM2			
51	MICIN			
28	RESET		DP	30
40	WAKEUP		DN	29
24	SHDN	STANDBY		41
112	JTAG_EN			
3, 35, 36, 85, 110, 111	U _{CC}		GND	4, 31, 59, 60, 65, 66, 86, 113
58, 67	U _{CCA}			
45	U _{CCAU}		GND _{AU}	46
27	U _{CCB}			
2, 32, 33, 87, 114	U _{CCD}		NC	1, 26, 73, 108, 126-128

Рисунок 2–1 – Условное графическое обозначение микросхемы

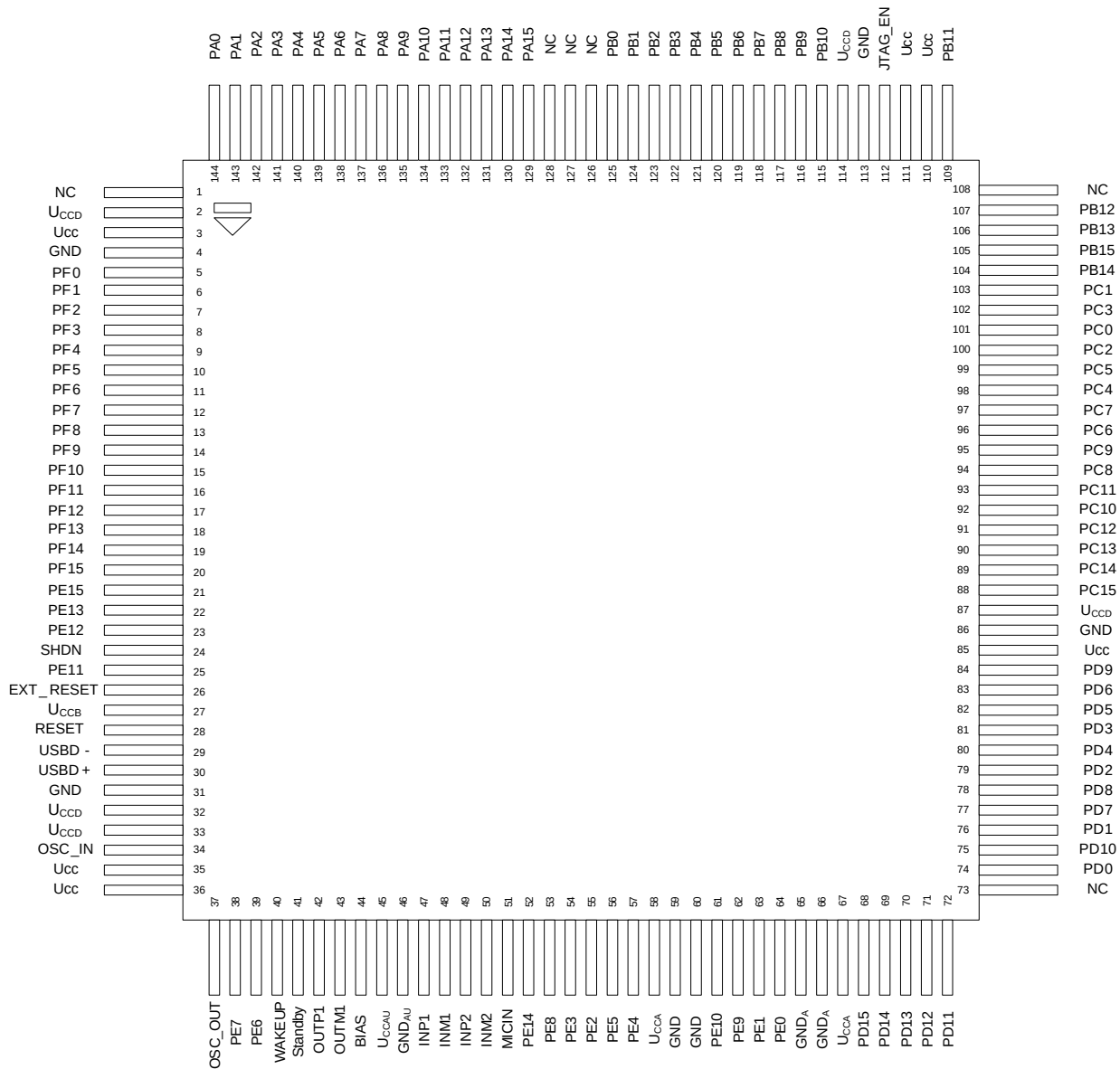


Рисунок 2-2 – Диаграмма расположения выводов

3 Описание выводов

Таблица 3–1 – Описание выводов микросхемы

Номер вывода	Обозначение вывода	Назначение вывода
1	NC	Не разваривается
2	U _{CCD}	Питание 1,8 (тестовый), подключить через конденсатор 0,1 мкФ на шину общий, соединить с выводами цепи U _{CCD}
3	U _{CC}	Питание 3,0 – 3,6 В
4	GND	Общий
5	PF0	Входной-выходной порт F
6	PF1	Входной-выходной порт F
7	PF2	Входной-выходной порт F
8	PF3	Входной-выходной порт F
9	PF4	Входной-выходной порт F
10	PF5	Входной-выходной порт F
11	PF6	Входной-выходной порт F
12	PF7	Входной-выходной порт F
13	PF8	Входной-выходной порт F
14	PF9	Входной-выходной порт F
15	PF10	Входной-выходной порт F
16	PF11	Входной-выходной порт F
17	PF12	Входной-выходной порт F
18	PF13	Входной-выходной порт F
19	PF14	Входной-выходной порт F
20	PF15	Входной-выходной порт F
21	PE15	Входной-выходной порт E
22	PE13	Входной-выходной порт E
23	PE12	Входной-выходной порт E
24	SHDN	Вывод отключения внутреннего регулятора питания 1,8 В. Используется для тестовых целей. Оставлять неподключенным
25	PE11	Входной-выходной порт E
26	EXT_RESET	Тестовый вывод. Рекомендуется не подсоединять
27	U _{CCB}	Питание батарейного домена
28	RESET	Вход Сброс
29	DN	Входной-выходной интерфейс USB
30	DP	Входной-выходной интерфейс USB
31	GND	Общий
32, 33	U _{CCD}	Питание 1,8 (тестовый), подключить через конденсатор 0,1 мкФ на шину общий, соединить с выводами цепи U _{CCD}
34	OSC_IN	Входной генератор HSE
35, 36	U _{CC}	Питание 3,0 – 3,6 В
37	OSC_OUT	Выходной генератор HSE
38	PE7	Входной-выходной порт E
39	PE6	Входной-выходной порт E
40	WAKEUP	Входной сигнал WakeUP

Номер вывода	Обозначение вывода	Назначение вывода
41	STANDBY	Входной сигнал StandBy
42	OUTP1	Вход Аудиокодек
43	OUTM1	Выход Аудиокодек
44	BIAS	Вход на задающий резистор
45	U _{CCAU}	Питание Аудиокодека 3,0 – 3,6 В
46	GNDAU	Общий Аудиокодека
47	INP1	Вход Аудиокодека 1
48	INM1	Вход Аудиокодека 1
49	INP2	Вход Аудиокодека 2
50	INM2	Вход Аудиокодека 2
51	MICIN	Вход микрофона Аудиокодека
52	PE14	Входной-выходной порт E
53	PE8	Входной-выходной порт E
54	PE3	Входной-выходной порт E
55	PE2	Входной-выходной порт E
56	PE5	Входной-выходной порт E
57	PE4	Входной-выходной порт E
58	U _{CCA}	Питание аналоговое 3,0 – 3,6 В
59, 60	GND	Общий
61	PE10	Входной-выходной порт E
62	PE9	Входной-выходной порт E
63	PE1	Входной-выходной порт E
64	PE0	Входной-выходной порт E
65, 66	GND	Общий
67	U _{CCA}	Питание аналоговое 3,0 – 3,6 В
68	PD15	Входной-выходной порт D
69	PD14	Входной-выходной порт D
70	PD13	Входной-выходной порт D
71	PD12	Входной-выходной порт D
72	PD11	Входной-выходной порт D
73	NC	Не разваривается
74	PD0	Входной-выходной порт D
75	PD10	Входной-выходной порт D
76	PD1	Входной-выходной порт D
77	PD7	Входной-выходной порт D
78	PD8	Входной-выходной порт D
79	PD2	Входной-выходной порт D
80	PD4	Входной-выходной порт D
81	PD3	Входной-выходной порт D
82	PD5	Входной-выходной порт D
83	PD6	Входной-выходной порт D
84	PD9	Входной-выходной порт D
85	U _{CC}	Питание 3,0 – 3,6 В
86	GND	Общий

Номер вывода	Обозначение вывода	Назначение вывода
87	U _{CCD}	Питание 1,8 (тестовый), подключить через конденсатор 0,1 мкФ на шину общий, соединить с выводами цепи U _{CCD}
88	PC15	Входной-выходной порт С
89	PC14	Входной-выходной порт С
90	PC13	Входной-выходной порт С
91	PC12	Входной-выходной порт С
92	PC10	Входной-выходной порт С
93	PC11	Входной-выходной порт С
94	PC8	Входной-выходной порт С
95	PC9	Входной-выходной порт С
96	PC6	Входной-выходной порт С
97	PC7	Входной-выходной порт С
98	PC4	Входной-выходной порт С
99	PC5	Входной-выходной порт С
100	PC2	Входной-выходной порт С
101	PC0	Входной-выходной порт С
102	PC3	Входной-выходной порт С
103	PC1	Входной-выходной порт С
104	PB14	Входной-выходной порт В
105	PB15	Входной-выходной порт В
106	PB13	Входной-выходной порт В
107	PB12	Входной-выходной порт В
108	NC	Не разваривается
109	PB11	Входной-выходной порт В
110, 111	U _{CC}	Питание 3,0 – 3,6 В
112	JTAG_EN	Тестовый. Рекомендуется подключать к выводу общий
113	GND	Общий
114	U _{CCD}	Питание 1,8 (тестовый), подключить через конденсатор 0,1 мкФ на шину общий, соединить с выводами цепи U _{CCD}
115	PB10	Входной-выходной порт В
116	PB9	Входной-выходной порт В
117	PB8	Входной-выходной порт В
118	PB7	Входной-выходной порт В
119	PB6	Входной-выходной порт В
120	PB5	Входной-выходной порт В
121	PB4	Входной-выходной порт В
122	PB3	Входной-выходной порт В
123	PB2	Входной-выходной порт В
124	PB1	Входной-выходной порт В
125	PB0	Входной-выходной порт В
126–128	NC	Не развариваются
129	PA15	Входной-выходной порт А
130	PA14	Входной-выходной порт А
131	PA13	Входной-выходной порт А

Номер вывода	Обозначение вывода	Назначение вывода
132	PA12	Входной-выходной порт А
133	PA11	Входной-выходной порт А
134	PA10	Входной-выходной порт А
135	PA9	Входной-выходной порт А
136	PA8	Входной-выходной порт А
137	PA7	Входной-выходной порт А
138	PA6	Входной-выходной порт А
139	PA5	Входной-выходной порт А
140	PA4	Входной-выходной порт А
141	PA3	Входной-выходной порт А
142	PA2	Входной-выходной порт А
143	PA1	Входной-выходной порт А
144	PA0	Входной-выходной порт А

Таблица 3–2 – Описание выводов по блокам

Обозначение вывода	Номер вывода	Дополнительные функции вывода			
		Аналог.	Основ.	Альтер.	Переопр & DSP
Порт A					
PA0	144	-	DATA0	SDIO_CLK	TMR1_CH1
PA1	143	-	DATA1	SDIO_CMD	TMR1_CH1n
PA2	142	-	DATA2	SDIO_DATA0	TMR1_CH2
PA3	141	-	DATA3	SDIO_DATA1	TMR1_CH2n
PA4	140	-	DATA4	SDIO_DATA2	TMR1_CH3
PA5	139	-	DATA5	SDIO_DATA3	TMR1_CH3n
PA6	138	-	DATA6	UART1_RXD	TMR1_CH4
PA7	137	-	DATA7	UART1_TXD	TMR1_CH4n
PA8	136	-	DATA8	SSP2_CLK	BSP1_RTCLK
PA9	135	-	DATA9	SSP2_FSS	BSP1_RTFR
PA10	134	-	DATA10	SSP2_TXD	BSP1_TX
PA11	133	-	DATA11	SSP2_RXD	BSP1_RX
PA12	132	-	DATA12	SSP1_RXD	TMR1_ETR
PA13	131	-	DATA13	SSP1_TXD	TMR1_BLK
PA14	130	-	DATA14	SSP1_CLK	TMR2_CH1
PA15	129	-	DATA15	SSP1_FSS	EXT_INT1/XF
Порт B					
PB0/ JA_TDO	125	-	DATA16	UART2_TXD	TMR2_CH1n
PB1/ JA_TMS	124	-	DATA17	TMR1_ETR	TMR2_CH2
PB2/ JA_TCK	123	-	DATA18	TMR1_BLK	TMR2_CH2n
PB3/ JA_TDI	122	-	DATA19	UART2_RXD	TMR2_CH3

Обозначение вывода	Номер вывода	Дополнительные функции вывода			
		Аналог.	Основ.	Альтер.	Переопр & DSP
PB4/ JA_TRST	121	-	DATA20	TMR1_CH1	TMR2_CH3n
PB5	120	-	DATA21	SSP3_TXD	BSP1_TX
PB6	119	-	DATA22	SSP3_FSS	BSP1_TFR
PB7	118	-	DATA23	SSP3_RXD	BSP1_RX
PB8	117	-	DATA24	SSP3_CLK	BSP1_TCLK
PB9	116	-	DATA25	TMR1_CH1n	TMR2_ETR
PB10	115	-	DATA26	TMR1_CH2	TMR2_BLK
PB11	109	-	DATA27	TMR1_CH2n	EXT_INT2
PB12	107	-	DATA28	SSP2_CLK	BSP1_RCLK
PB13	106	-	DATA29	SSP2_FSS	BSP1_RFR
PB14	104	-	DATA30	SSP2_TXD	BSP1_TX
PB15	105	-	DATA31	SSP2_RXD	BSP1_RX
Попр C					
PC0	101	-	COMP_OUT	SSP4_FSS	SDIO_CLK
PC1	103	-	OE	UART1_TXD	SDIO_CMD
PC2	100	-	WE	UART1_RXD	SDIO_DATA0
PC3	102	-	BE0	SSP4_RXD	SDIO_DATA1
PC4	98	-	BE1	SSP4_TXD	SDIO_DATA2
PC5	99	-	BE2	SSP4_CLK	SDIO_DATA3
PC6	96	-	BE3	TMR1_CH3	UART2_TXD
PC7	97	-	CLOCK	TMR1_CH3n	UART2_RXD
PC8	94	-	SDIO_CLK	SSP3_TXD	BSP2_TX
PC9	95	-	SDIO_CMD	SSP3_FSS	BSP2_RTFR
PC10	92	-	SDIO_DATA0	SSP3_RXD	BSP2_RX
PC11	93	-	SDIO_DATA1	SSP3_CLK	BSP2_RTCLK
PC12	91	-	SDIO_DATA2	SSP4_FSS	BSP3_RTFR
PC13	90	-	SDIO_DATA3	SSP4_CLK	BSP3_RTCLK
PC14	89	-	I2C1_SCK	SSP4_TXD	BSP3_TX
PC15	88	-	I2C_SDA	SSP4_RXD	BSP3_RX
Попр D					
PD0/ JB_TMS	74	ADC0_REF+	TMR1_BLK	I2C1_SCK	BSP3_RX
PD1/ JB_TCK	76	ADC1_REF-	TMR1_ETR	I2C1_SDA	BSP3_TX
PD2/ JB_TRST	79	ADC2	BUSY1	SSP1_TXD	BSP2_TX
PD3/ JB_TDI	81	ADC3	TMR1_CH1	SSP1_FSS	BSP2_TFR
PD4/ JB_TDO	80	ADC4	TMR1_CH1n	SSP1_RXD	BSP2_RX
PD5	82	ADC5	CLE	SSP1_CLK	BSP2_TCKL
PD6	83	ADC6	ALE	TMR1_CH4	BSP3_TCLK
PD7	77	ADC7	TMR1_CH2	UART1_SROUT	BSP3_TFR

Обозначение вывода	Номер вывода	Дополнительные функции вывода			
		Аналог.	Основ.	Альтер.	Переопр & DSP
PD8	78	ADC8	TMR1_CH2n	UART1_SIRIN	BSP3_TX
PD9	84	ADC9	TMR1_CH3	TMR1_CH4n	BSP3_RFR
PD10	75	ADC10	TMR1_CH3n	TMR2_BLK	BSP3_RX
PD11	72	ADC11	TMR1_CH4	TMR2_ETR	BSP3_RCLK
PD12	71	ADC12	TMR1_CH4n	SSP2_FSS	BSP2_RFR
PD13	70	ADC13	UART3_TXD	SSP2_RXD	BSP2_RX
PD14	69	ADC14	UART3_RXD	SSP2_CLK	BSP2_RCKL
PD15	68	ADC15	EXT_INT1	SSP2_TXD	BSP2_TX
Порт E					
PE0	64	DAC2_OUT	ADDR16	SSP4_FSS	BSP1_RTFR
PE1	63	DAC2_REF	ADDR17	SSP4_CLK	BSP1_RTCLK
PE2	55	COMP_IN1	ADDR18	SSP4_TXD	BSP1_TX
PE3	54	COMP_IN2	ADDR19	SSP4_RXD	BSP1_RX
PE4	57	COMP_REF+	ADDR20	SSP2_TXD	UART1_TXD
PE5	56	COMP_REF-	ADDR21	SSP2_FSS	UART1_RXD
PE6	39	OSC_IN32	ADDR22	SSP2_RXD	EXT_INT3
PE7	38	OSC_OUT32	ADDR23	SSP2_CLK	TMR2_CH4
PE8	53	COMP_IN3	ADDR24	TMR2_CH4	TMR2_CH4n
PE9	62	DAC1_OUT	ADDR25	TMR2_CH4n	TMR3_CH1
PE10	61	DAC1_REF	ADDR26	UART2_TXD	TMR3_ETR
PE11	25	-	ADDR27	UART2_RXD	TMR3_BLK
PE12	23	-	ADDR28	SSP1_RXD	BSP2_RTFR
PE13	22	-	ADDR29	SSP1_TXD	BSP2_RTCLK
PE14	52	-	ADDR30	SSP1_CLK	BSP2_TX
PE15	21	-	ADDR31	SSP1_FSS	BSP2_RX
Порт F					
PF0	5	-	ADDR0	UART3_RXD	TMR3_CH1n
PF1	6	-	ADDR1	UART3_TXD	TMR3_CH2
PF2	7	-	ADDR2	SSP4_RXD	BSP3_RX
PF3	8	-	ADDR3	SSP4_TXD	BSP3_TX
PF4/ MODE[0]	9	-	ADDR4	SSP4_CLK	BSP3_TCKL
PF5/ MODE[1]	10	-	ADDR5	SSP4_FSS	BSP3_TFR
PF6/ MODE[2]	11	-	ADDR6	TMR2_CH3n	TMR3_CH2n
PF7	12	-	ADDR7	TMR2_CH3	TMR3_CH3
PF8	13	-	ADDR8	TMR2_CH2n	TMR3_CH3n
PF9	14	-	ADDR9	TMR2_CH2	TMR3_CH4
PF10	15	-	ADDR10	TMR2_CH1n	TMR3_CH4n
PF11	16	-	ADDR11	TMR2_CH1	EXT_INT4
PF12	17	-	ADDR12	SSP3_TXD	BSP3_TX
PF13	18	-	ADDR13	SSP3_FSS	BSP3_RFR
PF14	19	-	ADDR14	SSP3_RXD	BSP3_RX
PF15	20	-	ADDR15	SSP3_CLK	BSP3_RCKL

Обозначение вывода	Номер вывода	Дополнительные функции вывода			
		Аналог.	Основ.	Альтер.	Переопр & DSP
Аудио интерфейс					
MICIN	51	-			
INP2	49	-			
INM2	50	-			
INP1	47	-			
INM1	48	-			
OUTP1	42	-			
OUTM1	43	-			
BIAS	44	-			
Системное управление					
RESET	28	Сигнал внешнего сброса			
WAKEUP	40	Сигнал внешнего выхода из режима Standby			
STANDBY	41	Флаг режима Standby			
OSC_IN	34	Вход генератора HSE			
OSC_OUT	37	Выход генератора HSE			
Интерфейс USB					
DP	30	Шина USB D+			
DN	29	Шина USB D-			
Питание					
U _{CC}	3, 35, 36, 85, 110, 111	Питание 3,0...3,6 В			
U _{CCA}	58, 67	Аналоговое питание АЦП и ЦАП 3,0...3,6 В (должно совпадать с U _{CC})			
U _{CCAU}	45	Аналоговое питание аудиокодека 3,0...3,6 В (должно совпадать с U _{CC})			
U _{CCB}	27	Батарейное питание 1,8...3,6 В			
GND	4, 31, 59, 60, 65, 66, 86, 113	Общий			
GND _{AU}	46	Общий			
Выводы для тестирования и исследования					
SHDN	24	Вывод отключения внутреннего регулятора питания 1,8 В. Используется для тестовых целей. Рекомендуется оставить неподключенным			
U _{CCD}	2, 32, 33, 87, 114	Питание 1,8 В (тестовый), рекомендуется оставить неподключенным			
Не используются					
NC	1, 73, 108, 126-128	Не используются			

4 Структурная блок-схема микросхемы

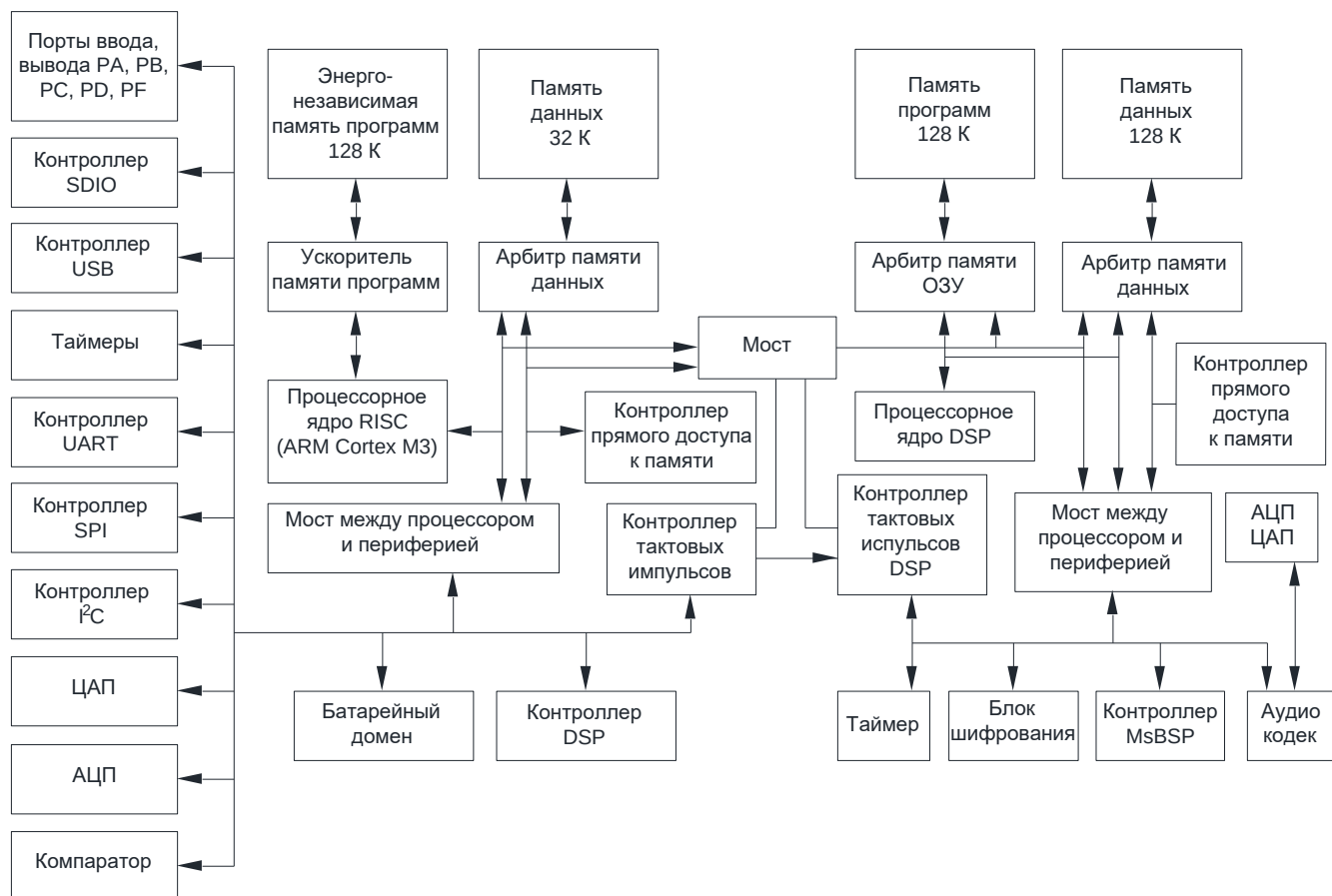


Рисунок 4–1 – Структурная блок-схема

5 Указания по применению и эксплуатации

При ремонте аппаратуры и измерении параметров микросхем замену микросхем необходимо проводить только при отключенных источниках питания.

Инструмент для пайки (сварки) и монтажа не должен иметь потенциал, превышающий 0,3 В относительно шины общий.

Запрещается подведение каких-либо электрических сигналов (в том числе шин питания, общий) к выводам микросхем, не используемым согласно схеме электрической.

Неиспользуемые пользовательские выводы PA – PF, nRESET, WAKEUP в цифровом режиме должны быть доопределены до допустимых логических уровней напряжения (высокого или низкого) с помощью внутренних программируемых резисторов доопределения или через внешние резисторы номиналом (1 – 100) кОм, или должны быть переведены в аналоговый режим.

В зависимости от рабочей частоты микроконтроллера для правильной работы регулятора LDO должны быть настроены поля SelectRI[2:0] и LOW[2:0] регистра REG_0E блока батарейного домена (при переходе на более высокую рабочую частоту настройка полей выполняется до перехода, при переходе с высокой частоты на низкую настройка полей выполняется после смены частоты).

Между выводами U_{CC} и GND устанавливается фильтрующая емкость не менее 0,1 мкФ.

Выводы питания 1,8 В U_{CCD} рекомендуется подключать в соответствии с типовой схемой включения, допускается оставлять эти выводы не подключёнными.

Типовая схема включения приведена в разделе 40 «Типовая схема включения» (Рисунок 40–1).

Порядок подачи и снятия напряжения питания и входных сигналов на микросхему:

- подача (включение микросхемы) – общий, питание батарейного домена U_{CCB} , напряжение питания U_{CC} , входные сигналы или одновременно;
- снятие (выключение микросхемы) – в обратном порядке или одновременно.

6 Описание функционирования микросхемы

6.1 Система питания

Микросхема имеет несколько типов выводов питания:

- **U_{CC} выводы:** Основное питание микросхемы включает питание пользовательских выводов, встроенного регулятора напряжения, USB PNY и генераторов. Входное напряжение должно быть в пределах от 3,0 до 3,6 В.

- **U_{CCD} выводы:** Питание внутренней цифровой части, памяти ОЗУ и FLASH-памяти. В нормальном режиме работы это питание формируется внутренним регулятором напряжения из U_{CC}. Выводы питания U_{CCD} рекомендуется подключать в соответствии с типовой схемой включения, допускается оставлять эти выводы неподключенными. Напряжение на выводе U_{CCD} должно быть в пределах от 1,62 до 1,98 В.

- **U_{CCB} вывод:** Питание батарейного домена используется при отсутствии основного питания U_{CC} для питания батарейного домена и LSE генератора. Переключение с основного питания на батарейное происходит автоматически при снижении ниже 2,0 В уровня основного питания U_{CC}. Переключение с батарейного питания на основное происходит автоматически, спустя примерно 4 мс, после повышения питания U_{CC} выше уровня 2,0 В. Входное напряжение должно быть в пределах от 1,8 до 3,6 В. Если в системе не требуется батарейного питания, вывод U_{CCB} должен быть объединен с U_{CC}.

- **U_{CCA} выводы:** Питание аналоговых блоков выведено на отдельные выводы для уменьшения помех, создаваемых работой других блоков. На данные выводы рекомендуется подавать напряжение с того же источника, что и U_{CC}, но при этом на печатной плате должны быть применены меры по снижению наводки помех. Допускается использование отдельного источника для питания аналоговых блоков, но при этом его выходное напряжение не должно отличаться от U_{CC} более чем на $\pm 0,2$ В.

- **U_{CCA}U выводы:** Питание аналоговой части аудиокодека выведено на отдельные выводы для уменьшения помех, создаваемых работой других блоков. На данные выводы рекомендуется подавать напряжение с того же источника, что и U_{CC}, но при этом на печатной плате должны быть применены меры по снижению наводки помех. Допускается использование отдельного источника для питания аудиокодека, но при этом его выходное напряжение не должно отличаться от U_{CC} более чем на $\pm 0,2$ В.

- **GND выводы:** Основная земля питания.

- **GND_{AU} выводы:** Земля аналогового питания U_{CCA}U. Данные выводы должны соединяться с GND, но при этом на печатной плате должны быть применены меры по снижению наводки помех.

6.2 Структурная схема подачи питания

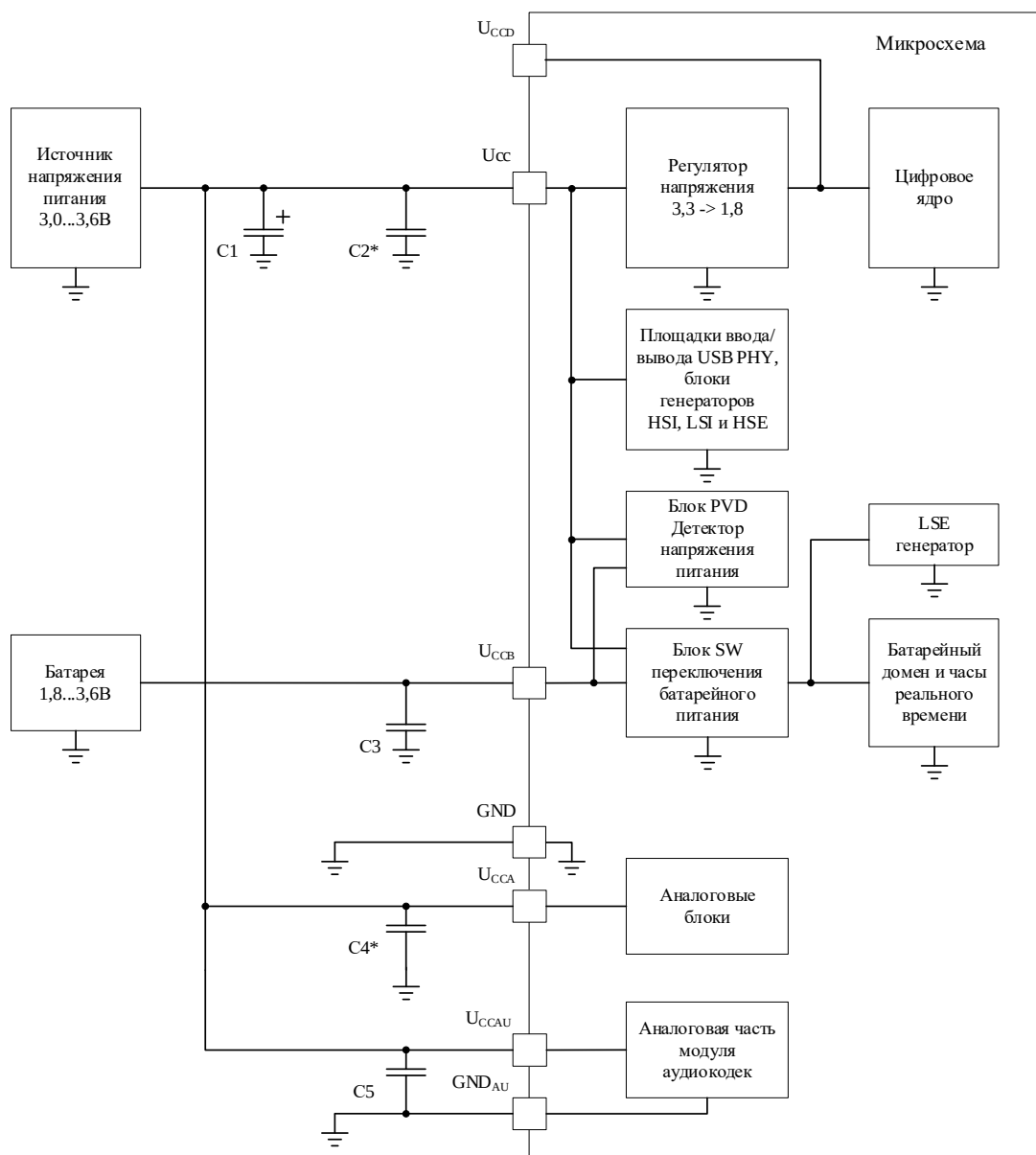


Рисунок 6-1 – Структурная блок-схема

Примечания

* Конденсаторы должны быть установлены у каждого вывода питания.

1 Конденсаторы:

C1 = 22 мкФ $\pm$ 5%, 16 В,

C2 = C3 = C4 = C5 = 0,1 мкФ $\pm$ 5%, 16 В.

2 Если не используется батарейное питание, то вывод UCCB должен быть объединен с UCC.

3 Допускается использование отдельного источника для питания АЦП, ЦАП и аудиокодека, но при этом его выходное напряжение не должно отличаться от UCC более чем на $\pm 0,2$ В.

Микросхема имеет несколько режимов энергопотребления, подробнее смотри подраздел 36.8 «Управление электропитанием».

Микросхема имеет встроенный детектор напряжения питания, подробнее смотри раздел 15 «Детектор напряжения питания MDR_POWER».

6.3 Схема сброса при включении и выключении основного питания

При включении питания, пока питание U_{CC} не превысило уровень U_{POR} (2,0В) вырабатывается внутренний сигнал сброса POR для цифровой части. После превышения уровня U_{POR} , сигнал POR выдается еще на протяжении трог (~4 мс) для того что бы гарантировано установилось напряжение питания, после чего сигнал POR снимается и схема может начать работать.

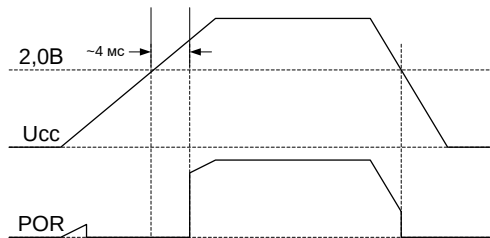


Рисунок 6-2 – Диаграмма формирования сигнала POR

При снижении напряжения питания U_{CC} ниже уровня U_{POR} сигнал POR вырабатывается без задержки.

Сигнал POR также служит для переключения питания батарейного домена между U_{CCB} и U_{CC} .

При включении основного напряжения питания U_{CC} автоматически включается встроенный регулятор напряжения для формирования напряжения U_{CCD} питания цифрового ядра. В ходе работы микросхемы встроенный регулятор может быть отключен, подробнее в см. в подразделе 36.8 «Управление электропитанием».

Микросхема также может быть сброшена внешним сигналом сброса nRESET, внутренними сигналами сброса сторожевых таймеров или программным сбросом. При этом сигнал сброса формируется специальной схемой сброса, содержащий фильтр «иголок» по сигналу сброса и одновибратор для увеличения длительности сигнала сброса.

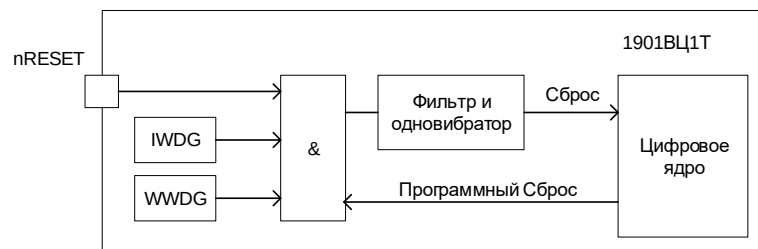


Рисунок 6-3 – Структурная блок-схема сброса

При подаче на вход nRESET импульсов сброса длительностью менее 10 нс эти импульсы отфильтровываются и не приводят к сбросу процессора. Если длительность импульса больше 200 нс, вырабатывается сигнал сброса. При этом длительность сформированного сигнала сброса будет не менее 20 мкс.

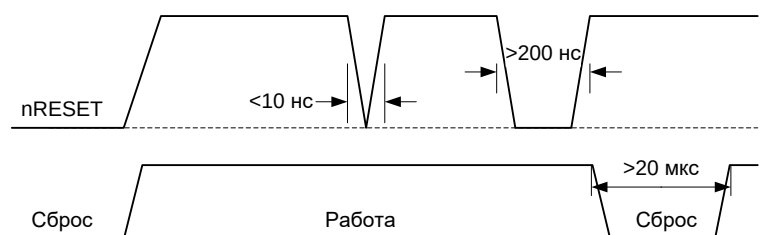


Рисунок 6-4 – Формирование сигнала сброса

6.4 Организация памяти

6.4.1 Структурная схема

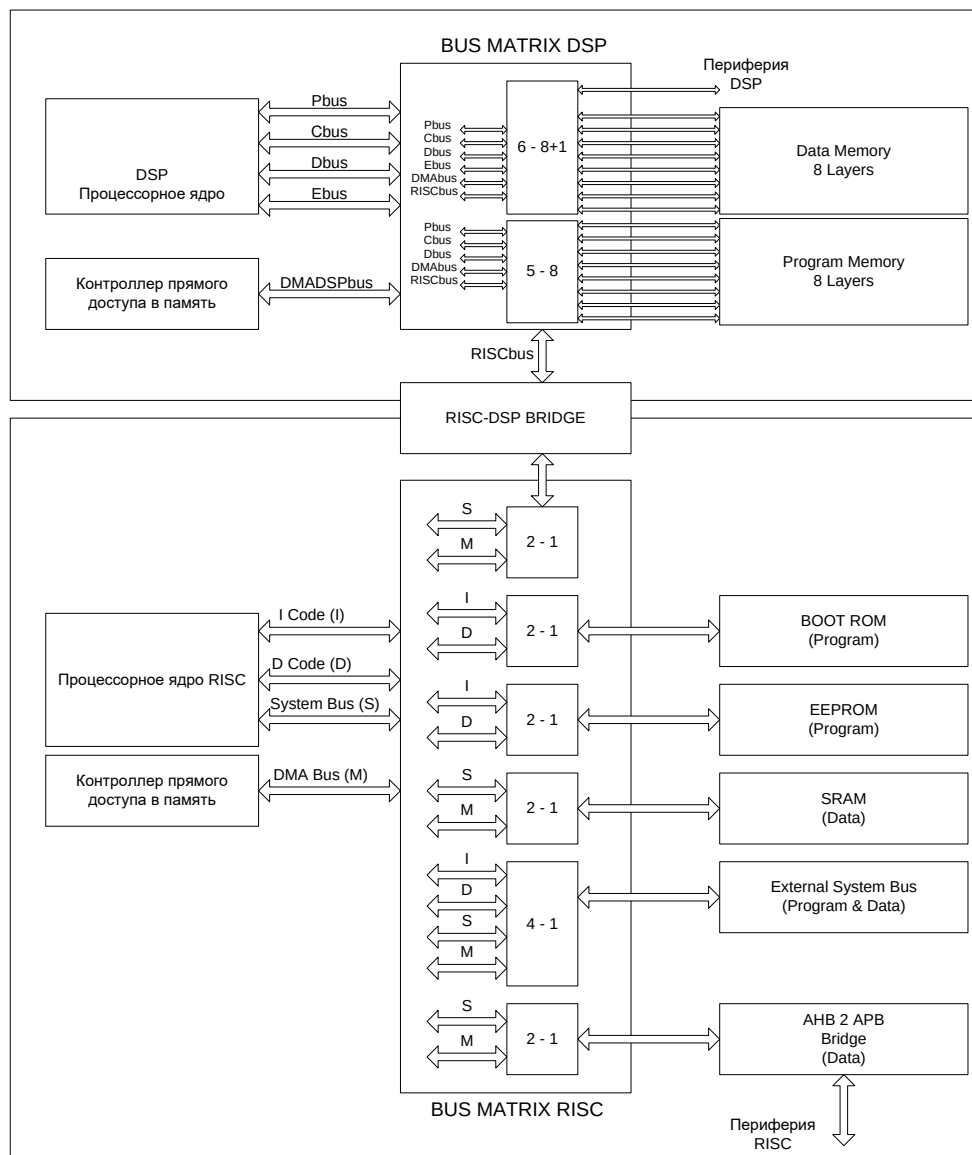


Рисунок 6-5 – Структурная схема микросхемы

Вся система структурно разделена на две подсистемы: RISC и DSP. В подсистему DSP входят память программ (128 Кбайт), память данных (128 кБайт), периферийные модули McBSP, Аудиокодек, Криптомодуль, таймер, некоторые отдельные специальные управляющие регистры. Все перечисленные устройства отображены в памяти DSP и памяти RISC. Кроме того, в память RISC отображены и другие устройства: FLASH-память, RAM (RISC), внешние устройства, другие периферийные модули. Таким образом, ядро DSP имеет доступ к подсистеме DSP микросхемы, ядро RISC является ведущим в системе и имеет доступ ко всем частям системы и к внешним устройствам через 32-разрядную внешнюю шину.

В подсистеме RISC реализован контроллер прямого доступа в память (DMA) осуществляющий выборку через шину DMA Bus. Он может работать с памятью RAM (RISC), внешней шиной, всей подсистемой DSP и всеми периферийными устройствами системы. В подсистеме DSP также реализован контроллер прямого доступа в память (DMA) осуществляющий выборку ко всем элементам подсистемы DSP.

Блок RISC-DSP BRIDGE

Подсистемы RISC и DSP работают на независимых синхросигналах. Для обеспечения их синхронизации в системы введен модуль RISC-DSP BRIDGE. За счет использования асинхронного FIFO данный модуль обеспечивает запись без потери скорости на пересинхронизацию на наименьшей из двух (RISC или DSP) частот.

Потери на пересинхронизацию при чтении составляют три периода наименьшей частоты. При этом в модуль интегрирован механизм предчтения следующих, запрашиваемых данных. Это позволяет читать последовательные данные с потерями на пересинхронизацию не более одного периода наименьшей частоты.

Блоки BUS MATRIX (RISC и DSP)

В системе содержится два модуля BUS MATRIX. BUS MATRIX RISC предназначен для переключения системных шин I Code, D Code, System Bus и DMA Bus между различными областями памяти. BUS MATRIX DSP выполняет аналогичные операции для шин подсистемы DSP (Pbus, Cbus, Dbus, Ebus, DMADSPbus и RISCbus). Переключение производится автоматически на основании адреса запроса каждой конкретной шины. Если адреса запросов не пересекаются, то они могут быть выполнены одновременно. Если адреса запросов пересекаются, то они выполняются в порядке приоритета. Приоритеты обращений заданы аппаратно. Наивысшим приоритетом в подсистеме RISC обладает запрос по шине SYSTEM BUS, затем следует запрос D Code, затем I Code и наименьшим приоритетом обладает запрос DMA Bus. Если два запроса пришли одновременно, то выполняется запрос с большим приоритетом, а запрос с меньшим задерживается до окончания запроса с большим приоритетом. При переключении между шинами возникает дополнительная задержка в один цикл. Если запросы идут непосредственно друг за другом, то дополнительных задержек не возникает.

Память BOOT ROM

Память области BOOT ROM реализована в виде MASK ROM, с занесением информации одним из технологических слоев при изготовлении кристалла микросхемы. Скорость доступа к памяти BOOT ROM – 1 цикл системной частоты.

Память EEPROM

Память области EEPROM реализована в виде перепрограммируемой энергонезависимой памяти. Скорость доступа к памяти EEPROM – порядка 40 нс. При работе микросхемы на скорости до 100 МГц, скорость доступа к памяти может составлять до 5 циклов системной частоты. При последовательной выборке за счет упреждающего чтения задержка может быть сокращена до 1 цикла системной частоты. Более подробная информация о работе контроллера EEPROM памяти программ представлена в разделе 7 «Контроллер FLASH-памяти программ».

Память SRAM

Память области SRAM реализована в виде блока статической памяти. Скорость доступа к памяти SRAM – 1 цикл системной частоты.

Память Program Memory (DSP)

Память области Program Memory (DSP) реализована в виде блока статической расслоенной памяти. Скорость доступа к памяти Program Memory (DSP) – 1 цикл системной частоты DSP.

Память DATA Memory (DSP)

Память области Program Memory (DSP) реализована в виде блока статической расслоенной памяти. Скорость доступа к памяти Program Memory (DSP) – 1 цикл системной частоты DSP.

Карта памяти RISC

Все адресное пространство RISC едино и имеет максимальный объем 4 Гбайта (Рисунок 6-6). В данное адресное пространство отображаются все модули памяти и периферии микросхемы, в том числе и все компоненты подсистемы DSP (кроме регистров ядра DSP).

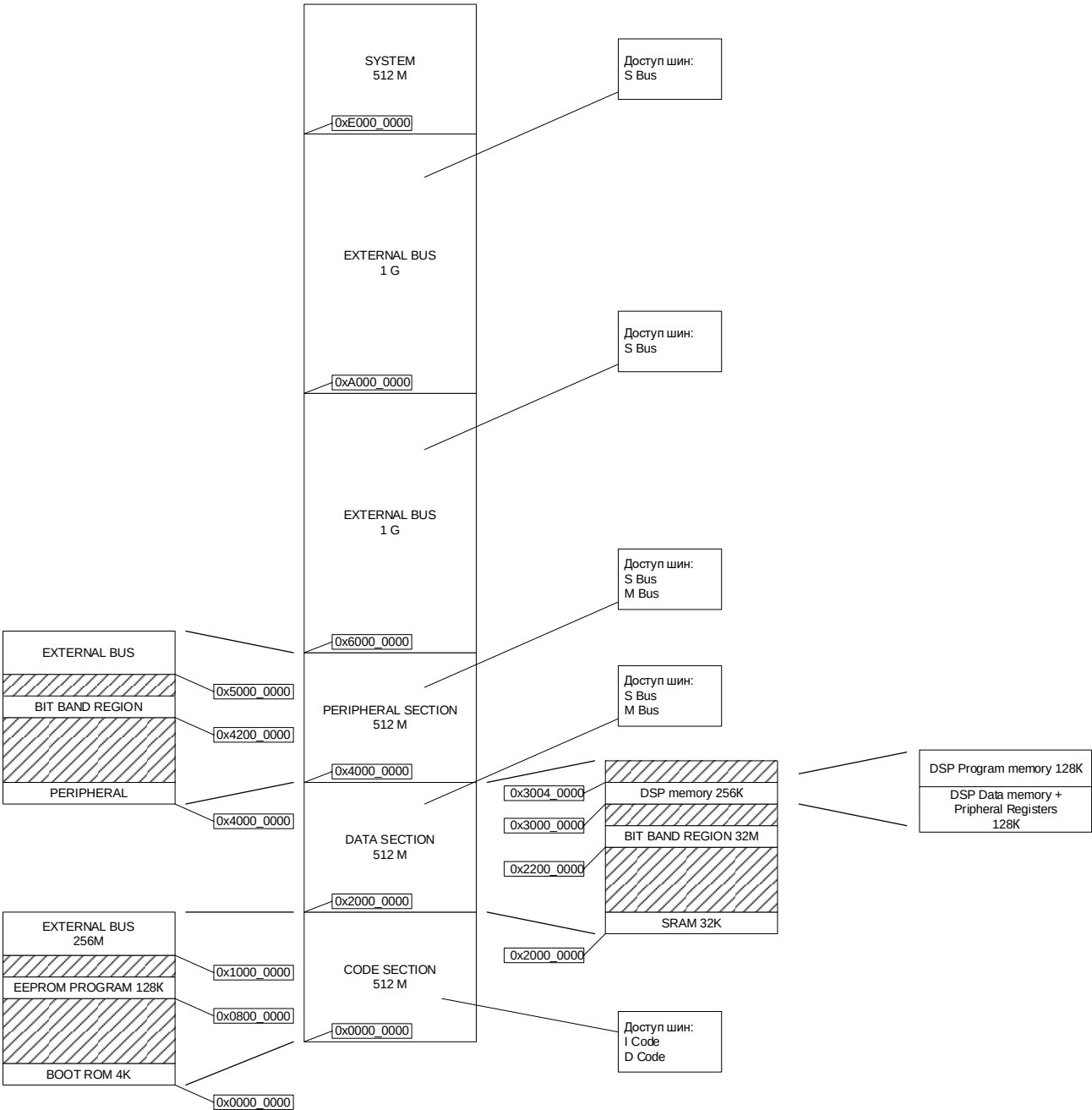


Рисунок 6-6 – Карта памяти RISC

Процессорное ядро RISC имеет три системных шины (Рисунок 6-5):

- I Code – шина выборки инструкций;
- D Code – шина выборки данных, расположенный в коде программы;
- S Bus – шина выборки данных, расположенный в области ОЗУ.

Также в подсистеме RISC реализован шина контроллера прямого доступа в память (Рисунок 6-5) :

- M Bus.

Секция CODE:

Область BOOT ROM:

Предназначена для хранения программы запуска микросхемы, в ходе выполнения этой программы определяется режим запуска основной программы или переход в режим программирования микросхемы.

Область EEPROM PROGRAM:

Основная область энергонезависимой памяти программы доступной для перепрограммирования пользователем. Память предназначена для хранения основной рабочей программы.

Область EXTERNAL BUS:

Область отображения внешней системной шины в адресное пространство области программы. Предназначена для хранения кода программ во внешних микросхемах памяти подсоединенных к внешней системной шине.

Секция DATA:

Область Internal SRAM (Data):

Основная область ОЗУ предназначенная для хранения данных программы. В данной области также располагается стек (stack) и «куча» (heap) программы. Адресные диапазоны стека и «кучи» задаются пользователем при написании программы.

Область BIT BAND REGION TO SRAM (Data):

Виртуальная Область памяти данных, предназначенная для осуществления побитного доступа к области Internal SRAM. Работа с BIT BAND REGION позволяет осуществлять операции «Чтение-Модификация-Запись», «Установка бита» и «Сброс Бита» одной инструкцией.

Область DSP MEMORY:

Область отображения адресного пространства подсистемы DSP в адресное пространство RISC. Предназначена для обмена данными с ядром DSP и периферией и задания программ для работы DSP.

Секция PERIPHERAL:

Область PERIPHERAL (Data):

Область отображения регистров периферии в общее адресное пространство памяти.

Область BIT BAND REGION TO PERIPHERAL (Data):

Виртуальная Область памяти данных, предназначенная для осуществления побитного доступа к области PERIPHERAL. Работа с BIT BAND REGION позволяет осуществлять операции «Чтение-Модификация-Запись», «Установка бита» и «Сброс Бита» одной инструкцией.

Область EXTERNAL BUS:

Область отображения внешней системной шины в адресное пространство области периферии. Предназначена для хранения данных во внешних микросхемах памяти или работы с периферийными устройствами подсоединенными к внешней системной шине.

Секция EXTERNAL RAM:

Область EXTERNAL BUS:

Область отображения внешней системной шины в адресное пространство области внешней памяти и периферии. Предназначена для хранения данных во внешних микросхемах памяти или работы с периферийными устройствами подсоединенными к внешней системной шине.

Секция SYSTEM:

Предназначена для отображения системных регистров ядра и системной периферии.

6.4.1.1 Регионы памяти, типы и атрибуты RISC

Отображение памяти и программирования блока MPU разбивает все адресное пространство на регионы. Каждый регион имеет определенный тип памяти, а некоторые регионы имеют дополнительные атрибуты. Тип памяти и атрибуты определяют поведение системы при доступе к этим регионам. Подробнее смотри раздел Модуль защиты памяти.

Normal

Процессор может переопределить последовательность обращений для большей эффективности или проведения спекулятивного считывания

Device

Процессор сохраняет последовательность обращений по отношению к другим обращениям в области Device и Strongly-ordered памяти

Strongly-ordered

Процессор сохраняет последовательность обращений по отношению ко всем обращениям. Отличие последовательности для Device и Strongly-Ordered памяти означает что система памяти может буферизировать запись в Device, но никогда не буферизирует запись в Strongly-ordered память.

Shareable

Для shareable регионов система памяти обеспечивает синхронизацию между различными мастерами на шине, например DMA и само ядро. Strongly-ordered память всегда shareable. При наличии работе нескольких мастеров в не shareable памяти, программное обеспечение должно отслеживать когерентность данных различных мастеров.

Execution Never (XN)

Область памяти из которой не могут извлекаться инструкции. Любая попытка извлечь инструкцию из XN региона приведет в исключению memory management fault.

Последовательность обращений в память

Для большинства обращений в память выполняемых инструкциями доступа, система памяти не гарантирует последовательность их выполнения в соответствии с последовательностью выполнения этих инструкций, за исключением когда эта последовательность может повлиять на последовательность инструкций. Обычно, когда выполнение программы требует последовательно выполнить два обращения в память, то программно должна быть выполнена барьерная инструкция между инструкциями обращения. Смотри раздел программное определение последовательности доступов в память

Однако, система памяти гарантирует однозначную последовательность доступа в регионы памяти Device и Strongly-ordered. Для двух инструкций доступа в память A1 и A2, если A1 выполняется перед A2 в коде программы, последовательность обращений двух инструкций будет такой как показано ниже (Таблица 6-1).

Таблица 6-1 – Последовательность обращений инструкций к памяти

A1	A2			
	Доступ в Normal	Доступ в Device He shareable	Доступ в Device shareable	Strongly-ordered
Доступ в Normal	–	–	–	–
Доступ в Device He shareable	–	<	–	<
Доступ в Device shareable	–	–	<	<
Strongly-ordered	–	<	<	<
Обозначения: « – » – система памяти не гарантирует последовательность выполнения обращений; « < » – обращение инструкции A1 всегда будет выполнено перед инструкций A2				

Поведение обращений к памяти

Поведение обращений описано ниже (Таблица 6-2).

Таблица 6-2 – Последовательность обращений инструкций к памяти

Адресный диапазон	Регион памяти	Тип памяти	XN	Описание
0x00000000-0x1FFFFFFF	Code	Normal	-	Область памяти для кода программы, данные также могут храниться здесь.
0x20000000-0x2FFFFFFF	SRAM	Normal	-	Область памяти для данных. Код программы также может располагаться здесь. Этот регион содержит области bit-band доступа
0x30000000-0x3FFFFFFF	DSP	Normal	-	Область памяти DSP и регистров периферии DSP. Не рекомендуется использовать память DSP для расположения кодов команд RISC. При записи и чтения этой области памяти возможны задержки вследствие пересинхронизации обращений. Подробнее см. подраздел 18.6 «Работа моста пересинхронизации RISC – DSP».
0x40000000-0x5FFFFFFF	Peripheral	Device	XN	Этот регион содержит области bit-band доступа
0x60000000-0x9FFFFFFF	External RAM	Normal	-	Область памяти для данных и кода
0xA0000000-0xDFFFFFFF	External Device	Deivce	XN	Область памяти для внешних устройств
0xE0000000-0xE00FFFFF	Private Peripheral Bus	Strongly-ordered	XN	Этот регион содержит регистры NVIC, system timer и регистры блока управления ядра
0xE0100000-0xFFFFFFFF	Зарезервировано	Device	XN	Зарезервировано

Области Code, SRAM и External RAM могут содержать код программы. Однако рекомендуется что бы код программы располагался в секции Code. Так как процессор имеет отдельные шины доступа к этой секции, что позволяет одновременно выполнять выборку инструкций и данных.

Блок MPU может влиять на стандартное поведения при доступе в память. Для большей информации обратитесь в раздел блок защиты памяти

Дополнительные условия доступа в shared память

Если система содержит shared память, некоторые регионы могут иметь дополнительные условия доступа, и некоторые области имеют свое собственное разбиение.

Таблица 6-3 – Поведение обращений к памяти

Адресный диапазон	Секция памяти	Тип памяти	Возможность совместного использования
0x00000000-0x1FFFFFFF	Code	Normal	-
0x20000000-0x2FFFFFFF	SRAM	Normal	-
0x30000000-0x3FFFFFFF	DSP	Normal	-
0x40000000-0x5FFFFFFF	Peripheral	Device	-
0x60000000-0x7FFFFFFF	External RAM	Normal	-
0x80000000-0x9FFFFFFF			
0xA0000000-0xBFFFFFFF	External device	Device	Shareable
0xC0000000-0xDFFFFFFF			«non-shareable»
0xE0000000-0xE0FFFFFFF	Private peripheral bus	Strongly-ordered	Shareable
0xE0100000-0xFFFFFFFF	Vendor-specific device	Device	-

Программное определение последовательности доступов в память

Последовательность инструкций в потоке программы не всегда гарантирует последовательность соответствующих обращений в память, это происходит потому, что:

- процессор может изменить последовательность обращений для увеличения производительности, но при этом не изменяется общее поведение программы;
- процессор имеет несколько шин обращений в память;
- память или устройства могут иметь различные скорости доступа;
- для некоторых обращений к памяти имеет место буферизация или упреждающее выполнение.

Этот раздел описывает как гарантировать при необходимости корректную последовательность обращений в память. Если порядок обращений в память критичен, программное обеспечение должно содержать инструкции барьерной синхронизации для задания корректной последовательности. Процессор предлагает следующие инструкции барьерной синхронизации:

DMB

Инструкция Data Memory Barrier (DMB) позволяет быть уверенным, что выполняемая инструкция транзакции в память будет завершена до следующей транзакции в память. Смори описание инструкции DMB.

DSB

Инструкция Data Synchronization Barrier позволяет быть уверенным, что выполняемая инструкция транзакции в память будет завершена до начала выполнения следующей инструкции. Смори описание инструкции DSB.

ISB

Инструкция Instruction Synchronization Barrier (ISB) позволяет быть уверенным, что эффект от выполнения всех завершенных обращений к памяти будет распространяться на последующие команды. Смотрите описание инструкции ISB.

Инструкции барьерной синхронизации используются, например, в таких случаях:

- Программирование MPU:

- используйте инструкцию DSB для уверенности в том, что изменение настроек MPU будет иметь эффект немедленно вслед за изменением контекста;

- используйте инструкцию ISB для уверенности в том, что изменение настроек MPU будет иметь эффект немедленно после программирования новых MPU регионов, если конфигурационный код MPU вызывается через переход или вызов функции. Если конфигурационный код MPU вызывается через механизм исключений (прерывания), то инструкция ISB не требуется.

- Таблица векторов прерывания. Если программа изменяет таблицу векторов прерывания и затем разрешает прерывания, то перед разрешением должна быть поставлена инструкция DMB. Это гарантирует, что в случае прерывания процессор уйдет на обработчик по новому адресу таблицы.

- Самомодифицируемый код. Если программа содержит самомодифицируемый код, используйте инструкцию ISB сразу после модификации кода программы. Это гарантирует, что после этого будет выполняться уже модифицированный код.

- Переключение карты памяти. Если система содержит механизм переключения карты памяти, то используйте инструкцию DSB после переключения карты памяти в программе. Это гарантирует, что дальнейшее выполнение инструкций будет идти с новой картой памяти.

- Динамическое изменение приоритетов исключений. Когда приоритеты исключений изменяются во время обработки исключения, используйте инструкцию DSB после изменения. Это гарантирует, что изменение произойдет при завершении инструкции DSB.

- Использование семафоров в системе с несколькими устройствами управления передачей данных по шине. Если система содержит несколько таких устройств управления, например, другой процессор, то оба процессора должны использовать DMB инструкции после каждой инструкции работы с семафорами. Это гарантирует, что другой мастер будет видеть обращения к памяти в той последовательности, в которой они выполняются.

Обращения к памяти типа Strongly-ordered, например, к системному блоку управления ядра (NVIC, System Timer и так далее) не требуют использовать инструкции DMB.

Bit-band регионы

Механизм bit-band отображает каждый бит в bit-band регионе в слово в bit-band alias регионе. Bit-band регион занимает младший 1 Мбайт области SRAM и области периферийных устройств, им соответствуют области по 32 Мбайта bit-band alias (Таблица 6-4).

Таблица 6-4 – Описание bit-band регионов

Адресный диапазон	Регион памяти	Доступ инструкций и данных
0x2000_0000-0x200F_FFFF	SRAM bit-band	Доступ к словам
0x2200_0000-0x23FF_FFFF	SRAM bit-band alias	Доступ к битам в области SRAM bit-band через доступ к словам в области SRAM bit-band alias
0x4000_0000-0x400F_FFFF	Peripheral bit-band	Доступ к словам
0x4200_0000-0x43FF_FFFF	Peripheral bit-band alias	Доступ к битам в области Peripheral bit-band через доступ к словам в области Peripheral bit-band alias

Следующая формула показывает как регион bit-band alias отображается в bit-band region:

$$bit\_word\_offset = (byte\_offset \cdot 32) + (bit\_number \cdot 4)$$

$$bit\_word\_addr = bit\_band\_base + bit\_word\_offset$$

где bit_word_offset – позиция необходимого бита в bit-band регионе;

bit_word_addr – адрес слова в bit-band alias регионе отображающего необходимый бит в bit-band регионе;

bit_band_base – начальный адрес bit-band alias региона;

$byte\_offset$ – номер байта с необходимым битом в bit-band регионе;

bit_number – номер необходимого бита в байте.

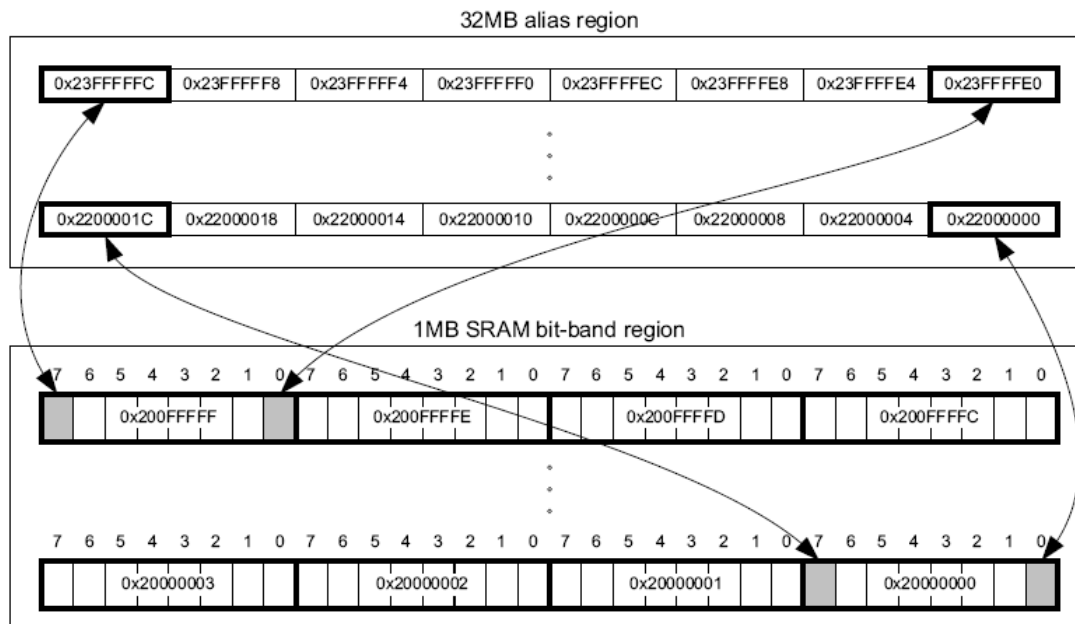


Рисунок 6-7 – Схема отображения региона bit-band alias в регионе bit-band region

Запись в слово в alias регионе обновляет бит в bit-band регионе.

Bit[0] записываемого слова будет определять значение устанавливаемого бита.

Bit[31:1] слова в bit-band alias регионе не имеют значения для бита в bit-band регионе. Запись 0x01 имеет тот же эффект что и запись 0xFF, а запись 0x00 имеет тот же эффект что и запись 0x0E.

Процессор имеет little-endian организацию расположения байтов. Т.е. байт с меньшей значимостью храниться по меньшему адресу, например:



Рисунок 6-8 – Организация расположения байтов в 32-битной памяти

6.4.1.2 Прimitives синхронизации

Система команд ядра RISC включает в себя несколько парных примитивов синхронизации. Это позволяет реализовать неблокирующий механизм, который поток или процесс могут использовать для эксклюзивного доступа в память. Программное обеспечение может использовать их для гарантированного выполнения последовательности read-modify-write или реализации механизма семафоров.

В каждую пару команд (инструкций) примитива синхронизаций входят:

- команда Load-Exclusive;
- команда Store-Exclusive.

Команда Load-Exclusive

Используется, чтобы прочесть значение из некоторого адреса памяти; запрашивает эксклюзивный доступ к этому адресу.

Команда Store-Exclusive

Используется для попытки записи в тот же самый адрес памяти; возвращает бит статуса. Этот бит принимает значения:

0 – если поток или процесс получил эксклюзивный доступ к памяти и запись выполнена;

1 – если поток или процесс не получил эксклюзивного доступа в память и запись не выполнена.

Команды Load Exclusive и Store-Exclusive образуют следующие пары:

- LDREX и STREX – работа с словами;
- LDREXH и STREXH – работа с полусловами;
- LDREXB и STREXB – работа с байтами.

Программное обеспечение должно использовать инструкцию Load Exclusive с соответствующей инструкций Store-Exclusive.

Чтобы гарантировать чтение-модификацию-запись по какому-либо адресу памяти, программа должна:

- инструкцией Load-Exclusive считать значение из памяти;
- изменить значение;
- инструкцией Store-Exclusive попытаться записать новое значение обратно в память; проверить возвращаемый статусный бит. Если этот бит – 0, то процедура «чтение-модификация-запись» выполнена успешно. Если этот бит – 1, то запись не была

выполнена. Это означает, что значение, считанное первоначально, возможно устарело и программа должна повторить цикл «чтение-модификация-запись».

Программное обеспечение может использовать примитивы синхронизации для реализации семафоров, как это описано ниже:

- использовать команду Load-Exclusive для чтения из адреса семафора, чтобы определить, свободен ли семафор;
- если семафор свободен, использовать Store-Exclusive для записи в семафор требуемого значения (признака захвата);
- если возвращаемый статусный бит указывает на успешное выполнение инструкции Store-Exclusive, то это означает, что семафор захвачен. Если же команда Store-Exclusive не была выполнена, то это означает, что другой процесс мог захватить семафор ранее.

Ядро RISC имеет монитор эксклюзивных доступов, который отмечает, что процессор выполнил команду Load-Exclusive. Если процессор является частью многопроцессорной системы, то система также отмечает на глобальном уровне адреса памяти, для которых имел место эксклюзивный доступ со стороны каждого процессора. Процессор удаляет свою отметку об эксклюзивном доступе в тех случаях, когда:

- процессор выполняет команду CLREX;
- процессор выполняет команду Store-Exclusive, при этом не имеет значения, была ли запись успешна;
- происходит исключение. Это означает, что процессор может разрешить конфликт между семафорами в различных потоках.

В многопроцессорной реализации:

- выполнение инструкции CLREX удаляет только локальную отметку об эксклюзивном доступе для данного процессора;
- выполнение инструкции Store-Exclusive или же обработка исключения удаляют локальную и все глобальные отметки об эксклюзивном доступе для данного процессора.

Подробнее см. описание инструкций LDREX, STREX и CLREX.

Указания по программированию синхронизационных примитивов

ANSI C не может создавать инструкции эксклюзивного доступа. Некоторые C компиляторы предлагают встроенные функции для создания этих инструкций:

```
LDREX, LDREXH, LDREXB – unsigned int __ldrex(volatile void *ptr);
STREX, STREXH, STREXB – int __strex(unsigned int val, volatile void *ptr);
CLREX                  – void __clrex(void).
```

Получаемые инструкции эксклюзивного доступа при создании зависят от типа данных указателя, передаваемого функции. Например, следующий код создаст инструкцию LDREXB:

```
__ldrex((volatile char *) 0xFF);
```

6.4.1.3 Базовые адреса RISC

Таблица 6-5 – Базовые адреса процессора

Адрес	Размер	Блок	Примечание
Память программ			
0x0000_0000		BOOT ROM	Загрузочная программа
0x0800_0000		EEPROM	Область FLASH-памяти программ с пользовательской программой
0x1000_0000		EXTERNAL BUS	Область доступа к внешней системной шине
Память данных			
0x2000_0000		SYSTEM RAM	Область внутреннего ОЗУ
0x2200_0000		SYSTEM RAM Bit Band Region	Область битового доступа внутреннего ОЗУ
0x3000_0000		DSP_SUBSYSTEM	Область доступа к подсистеме DSP (в т.ч. и периферийные модули подсистемы DSP: Timer(DSP), DMA(DSP), АудиоКодек, CryptoUnit, McBSP)
Периферийные устройства RISC			
0x4000_0000		SSP3	Регистры контроллера интерфейса SSP3
0x4000_8000		SSP4	Регистры контроллера интерфейса SSP4
0x4001_0000		USB	Регистры контроллера интерфейса USB
0x4001_8000		EEPROM_CNTRL	Регистры контроллера FLASH-памяти программ
0x4002_0000		RST_CLK	– Регистры контроллера сигналов тактовой частоты – Регистр управления DSP
0x4002_8000		DMA	Регистры контроллера прямого доступа в память RISC-подсистемы.
0x4003_0000		UART1	Регистры контроллера интерфейса UART1
0x4003_8000		UART2	Регистры контроллера интерфейса UART2
0x4004_0000		SSP1	Регистры контроллера интерфейса SSP1
0x4004_8000		SDIO	Регистры контроллера интерфейса SDIO
0x4005_0000		I2C1	Регистры контроллера интерфейса I2C1
0x4005_8000		POWER	Регистры детектора напряжения питания
0x4006_0000		WWDG	Регистры контроллера сторожевого таймера WWDG
0x4006_8000		IWDG	Регистры контроллера сторожевого таймера IWDG
0x4007_0000		TIMER1	Регистры управления Таймер 1
0x4007_8000		TIMER2	Регистры управления Таймер 2
0x4008_0000		TIMER3	Регистры управления Таймер 3
0x4008_8000		ADC	Регистры управления АЦП
0x4009_0000		DAC	Регистры управления ЦАП
0x4009_8000		COMP	Регистры управления Компаратора
0x400A_0000		SSP2	Регистры контроллера интерфейса SSP2
0x400A_8000		PORTA	Регистры управления порта А

Адрес	Размер	Блок	Примечание
0x400B_0000		PORTB	Регистры управления порта В
0x400B_8000		PORTC	Регистры управления порта С
0x400C_0000		PORTD	Регистры управления порта D
0x400C_8000		PORTE	Регистры управления порта E
0x400D_0000		UART3	Регистры контроллера интерфейса UART3
0x400D_8000		BKP	Регистры доступа и управления батарейным доменом
0x400E_8000		PORTF	Регистры управления порта F
0x400F_0000		EXT_BUS_CNTRL	Область доступа к внешней системной шине
0x4200_0000		PERIPHERAL Bit Band Region	Область битового доступа к регистрам периферии
0x5000_0000		EXTERNAL BUS	Область доступа к внешней системной шине
Внешняя системная шина			
0x6000_0000		EXTERNAL BUS	Область доступа к внешней системной шине
0xA000_0000		EXTERNAL BUS	Область доступа к внешней системной шине
SYSTEM REGION			
0xE000_0000			Системные регистры ядра RISC

6.4.2 Карта памяти DSP

Процессорное ядро DSP имеет четыре системные шины:

- Pbus – шина выборки инструкций.
- Cbus – шина выборки первого операнда.
- Dbus – шина выборки второго операнда.
- Ebus – шина записи результата.

Память DSP микропроцессора организована в два индивидуальные отдельные пространства: программ и данных. Оба пространства представляют собой расслоенные ОЗУ размером по 128 Кбайт каждое, с организацией $64K \times 16$.

Программное пространство памяти содержит инструкции для исполнения, а также таблицы, используемые при выполнении программ. Пространство памяти данных хранит данные используемые инструкциями. Биты OVLY расположен в регистре статуса режима процессора (PMST).

Для ядра DSP открыт полный доступ ко всему адресному пространству памяти данных и закрыт доступ на запись в программную область памяти. Для изменения программной области со стороны DSP необходимо использовать встроенный DMA контроллер подсистемы DSP. Кроме того, к области программ открыт полный доступ со стороны ядра RISC и DMA RISC-подсистемы.

Вектора сброса, прерывания и ловушек отображаются в адресах FF80h в пространстве программ. Тем не менее, эти вектора могут быть переадресованы в начало любой 128-словной страницы в пространстве программ после сброса устройства.

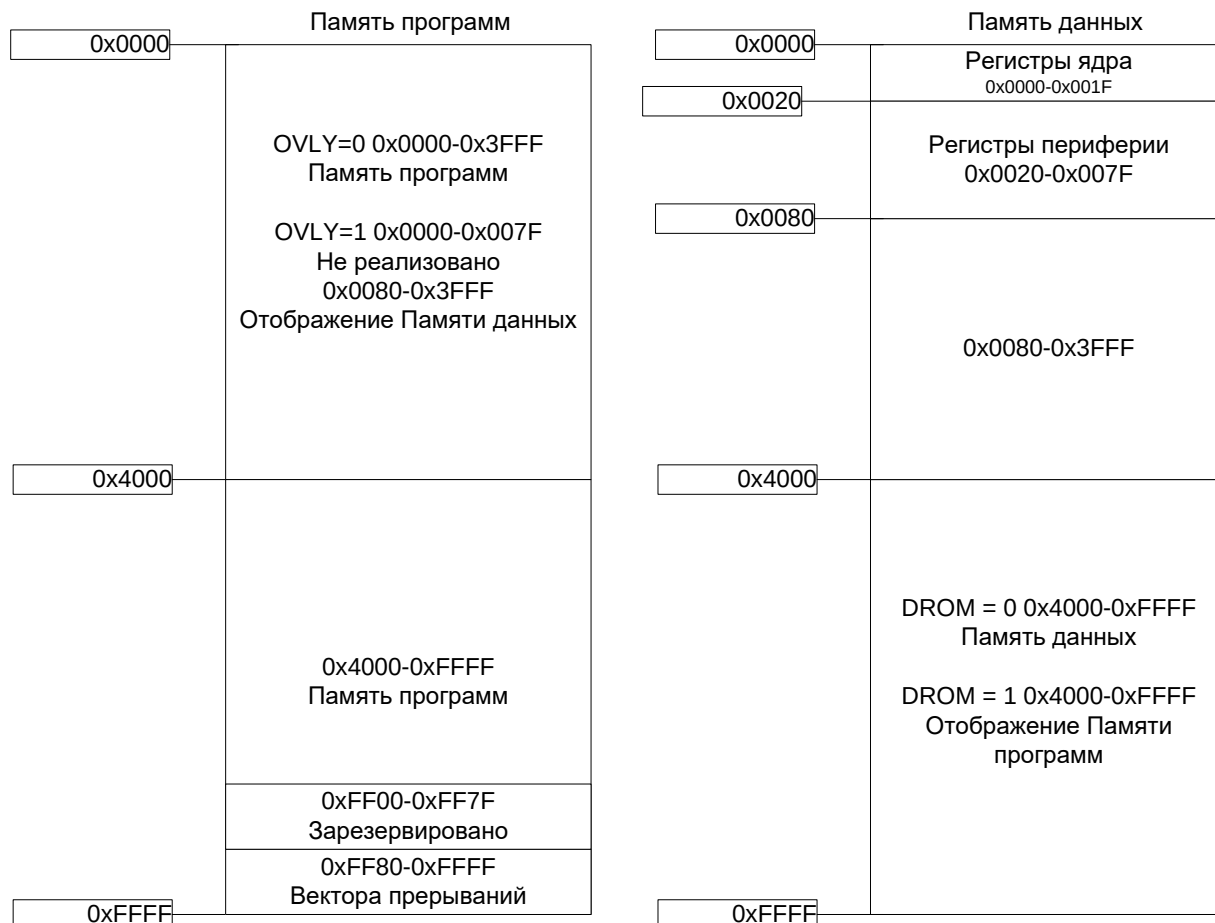


Рисунок 6-9 – Карта памяти DSP

6.4.2.1 Регистры, отображенные в памяти DSP

64K слов пространства памяти данных включает аппаратные регистры, отображаемые в памяти, которые находятся на странице данных 0 (адреса данных 0000h-007Fh). Страница данных состоит из следующих частей:

- регистры CPU (всего 26), доступны без состояний ожидания. Занимаемый диапазон 0020h-001F;
- периферийные регистры, использованные как управляющие регистры и регистры данных в периферийных цепях. Эти регистры занимают диапазон адресов 0020h-007F.

Регистры CPU DSP

Регистры прерывания (IMR, IFR)

Регистр маски прерывания (IMR) маскирует индивидуальные специфические прерывания в необходимое время. Регистр флага прерывания (IFR), указывает текущий статус прерываний.

Регистры состояния (ST0, ST1)

Регистры состояния ST0 и ST1 содержат статус различных условий и режимов для микропроцессора. ST0 содержит флаги (OVA, OVB, C, и TC) результатов арифметических и битовых операций, в дополнение к полям DP и ARP. ST1 отражает статус способов и инструкций, выполненных процессором.

Аккумуляторы (А, В)

Микропроцессор имеет два 40- битовых аккумулятора: аккумулятор А и аккумулятор В. Каждый аккумулятор отображен в память и разделен в младшее слово аккумулятора (AL, BL), старшее слово аккумулятора (AH, BH), и биты охраны (AG, BG).

Временный регистр (Т)

Временный регистр (Т) используется для различных целей, например он может быть использован как:

- один из сомножителей для операций умножения и умножения с накоплением;
- счетчик динамического сдвига (программируемый во время выполнения) для инструкций с операцией сдвига, как например, инструкции ADD, LD, и SUB;
- динамический адрес бита для инструкции BITT;
- метрика перехода используется инструкциями DADST и DSADT для операции ACS декодирования по Витерби.

Кроме того, инструкция EXP загружает вычисленную величину показателя в регистр Т, и затем инструкция NORM использует величину регистра Т для нормализации числа.

Регистр перехода (TRN)

16-битовый регистр перехода (TRN) держит решение о переходе в новую метрику, чтобы выполнять алгоритм Витерби. Инструкция CMPS (выбор максимального по сравнению и сохранение) обновляет содержание регистра TRN на основе сравнения между старшим словом аккумулятора и младшим словом аккумулятора.

Вспомогательные регистры (AR0 – AR7)

Восемь 16-битовых вспомогательных регистра (AR0 AR7) могут быть доступны ЦПУ и модифицируются арифметическим устройством вспомогательного регистра (ARAUs). Первичная функция вспомогательных регистров заключается в генерации 16-битовых адресов пространства данных. Тем не менее, эти регистры могут быть также задействованы как регистры общего назначения или счетчики.

Регистр указателя стека (SP)

16-битовый регистр указателя стека (SP), содержит адрес верхушки системного стека. SP всегда указывает на последний элемент, вытолкнутый в стек.

Стек изменяется прерываниями, перехватами, вызовами, возвратами, и инструкциями PSHD, PSHM, POPD, и POPM. Вталкивание и выталкивание осуществляется соответственно предкрементом и постинкрементом указателя стека, являющейся 16-битовой величиной.

Регистр размера циклического буфера (BK)

ARAUs использует 16-битовый регистр размера циклического буфера (BK) при циклической адресации, чтобы определять размер блока данных.

Регистры повторения блока (BRC, RSA, REA)

16-битовый счетчик повторения блока (BRC) определяет количество повторений программного блока тогда, когда выполняется повторение блока. 16-битовый регистр адреса начала (RSA) содержит стартовый адрес блока программной памяти, который должен повторяться. 16-битовый регистр адреса конца (REA) содержит адрес окончания блока программной памяти, которое нужно повторять.

Регистр состояния режима процессора (PMST)

Регистр состояния режима процессора (PMST) управляет конфигурацией памяти микропроцессора. PMST описывается подробно в разделе о регистре статуса и управления ЦПУ.

Таблица 6-6 – Регистры процессора, отображенные в памяти

Адрес (DSP)	Имя	Описание
0x0000	IMR	Регистр маски прерывания
0x0001	IFR	Регистр флага прерывания
0x0002...0x0005	-	Зарезервировано
0x0006	ST0	Регистр состояния 0
0x0007	ST1	Регистр состояния 1
0x0008	AL	Аккумулятор А, младшая часть(биты 15-0)
0x0009	АН	Аккумулятор А, старшая часть(биты 31-16)
0x000A	AG	Аккумулятор А, биты защиты(биты 39-32)
0x000B	BL	Аккумулятор В, младшая часть(биты 15-0)
0x000C	ВН	Аккумулятор В, старшая часть(биты 31-16)
0x000D	BG	Аккумулятор В, биты защиты(биты 39-32)
0x000E	T	Временный регистр
0x000F	TRN	Регистр перехода
0x0010	AR0	Вспомогательный регистр 0
0x0011	AR1	Вспомогательный регистр 1
0x0012	AR2	Вспомогательный регистр 2
0x0013	AR3	Вспомогательный регистр 3
0x0014	AR4	Вспомогательный регистр 4
0x0015	AR5	Вспомогательный регистр 5
0x0016	AR6	Вспомогательный регистр 6
0x0017	AR7	Вспомогательный регистр 7
0x0018	SP	Указатель стека
0x0019	BK	Регистр размера циклического буфера
0x001A	BRC	Счетчик повторения блока
0x001B	RSA	Начальный адрес повторяемого блока
0x001C	REA	Конечный адрес повторяемого блока
0x001D	PMST	Регистр состояния режима процессора
0x001E	XPC	Регистр расширения счетчика команд
0x001E...0x001F	-	Зарезервировано

Периферийные регистры DSP

В DSP части содержатся периферийный модули DMA (DSP), System timer (DSP), Audio Codec, три модуля McBSP, Crypto модуль. Также в таблицу адресов включены регистр управления синхросигналами CLKMD, регистры прерываний (AIRQ, DIRQ). Адреса всех периферийных модулей представлены в области данных DSP в диапазоне с адреса 0x0020 по 0x007F. Подробное описание регистров периферийных модулей приведены в соответствующих разделах. Адреса регистров DSP части для ядер DSP и RISC представлены в таблице ниже (Таблица 6-7).

Таблица 6-7 – Регистры периферийных модулей подсистемы DSP

Адрес DSP	Адрес RISC	Наименование регистра	Модуль, к которому принадлежит регистр
0x0020	0x30000040	DDRL1	BSP1
0x0021	0x30000042	DDRH1	BSP1
0x0022	0x30000044	DXRL1	BSP1
0x0023	0x30000046	DXRH1	BSP1
0x0024	0x30000048	SPSA1	BSP1
0x0025	0x3000004A	зарезервировано	
0x0026	0x3000004C	SPCRL1	BSP1
0x0027	0x3000004E	SPCRH1	BSP1
0x0028	0x30000050	DDRL2	BSP2
0x0029	0x30000052	DDRH2	BSP2
0x002A	0x30000054	DXRL2	BSP2
0x002B	0x30000056	DDRH2	BSP2
0x002C	0x30000058	SPSA2	BSP2
0x002D	0x3000005A	зарезервировано	
0x002E	0x3000005C	SPCRL2	BSP2
0x002F	0x3000005E	SPCRH2	BSP2
0x0030	0x30000060	DDRL3	BSP3
0x0031	0x30000062	DDRH3	BSP3
0x0032	0x30000064	DXRL3	BSP3
0x0033	0x30000066	DDRH3	BSP3
0x0034	0x30000068	SPSA3	BSP3
0x0035	0x3000006A	зарезервировано	BSP3
0x0036	0x3000006C	SPCRL3	BSP3
0x0037	0x3000006E	SPCRH3	BSP3
0x0038	0x30000070	TIM	Timer
0x0039	0x30000072	PRD	Timer
0x003A	0x30000074	TCR	Timer
0x003B	0x30000076	зарезервировано	
0x003C	0x30000078	DIRQ	DIRQ
0x003D	0x3000007A	AIRQ	AIRQ
0x003E	0x3000007C	зарезервировано	
0x0040	0x30000080	CRPT_CWR	Crypto
0x0041	0x30000082	зарезервировано	
0x0042	0x30000084	CRPT_SR	Crypto

Адрес DSP	Адрес RISC	Наименование регистра	Модуль, к которому принадлежит регистр
0x0043	0x30000086	зарезервировано	
0x0044	0x30000088	CRPT_DATA	Crypto
0x0045	0x3000008A	зарезервировано	
0x0046	0x3000008C	CRPT_KR	Crypto
0x0047	0x3000008E	зарезервировано	
0x0048	0x30000090	CRPT_SYNR	Crypto
0x0049	0x30000092	зарезервировано	
0x004A	0x30000094	CRPT_CR	Crypto
0x004B	0x30000096	зарезервировано	
0x004C	0x30000098	CRPT_IMIT	Crypto
0x004D	0x3000009A	зарезервировано	
0x004E	0x3000009C	CRPT_ITER	Crypto
0x004F	0x3000009E	зарезервировано	
0x0050	0x300000A0	POWCTL	Codec
0x0051	0x300000A2	зарезервировано	
0x0052	0x300000A4	ADCCTL	Codec
0x0053	0x300000A6	зарезервировано	
0x0054	0x300000A8	DACCTL	Codec
0x0055	0x300000AA	зарезервировано	
0x0056	0x300000AC	MASKCTL	Codec
0x0057	0x300000AE	зарезервировано	
0x0058	0x300000B0	IRQFLAG	Codec
0x0059	0x300000B2	зарезервировано	
0x005A	0x300000B4	ADCREG	Codec
0x005B	0x300000B6	зарезервировано	
0x005C	0x300000B8	DACREG	Codec
0x005D	0x300000BA	зарезервировано	
0x005E	0x300000BC	CLKMD	CLKMD
0x005F	0x300000BE	зарезервировано	
0x0060	0x300000C0	dma_statusL	DMA
0x0061	0x300000C2	dma_statusH	DMA
0x0062	0x300000C4	dma_configL	DMA
0x0063	0x300000C6	dma_configH	DMA
0x0064	0x300000C8	ctrl_base_ptrL	DMA
0x0065	0x300000CA	ctrl_base_ptrH	DMA
0x0066	0x300000CC	alt_ctrl_base_ptrL	DMA
0x0067	0x300000CE	alt_ctrl_base_ptrH	DMA
0x0068	0x300000D0	dma_waitonreq_statusL	DMA
0x0069	0x300000D2	dma_waitonreq_statusH	DMA
0x006A	0x300000D4	chnl_sw_requestL	DMA
0x006B	0x300000D6	chnl_sw_requestH	DMA
0x006C	0x300000D8	chnl_useburst_setL	DMA
0x006D	0x300000DA	chnl_useburst_setH	DMA
0x006E	0x300000DC	chnl_useburst_clrL	DMA
0x006F	0x300000DE	chnl_useburst_clrH	DMA

Адрес DSP	Адрес RISC	Наименование регистра	Модуль, к которому принадлежит регистр
0x0070	0x300000E0	chnl_req_mask_setL	DMA
0x0071	0x300000E2	chnl_req_mask_setH	DMA
0x0072	0x300000E4	chnl_req_mask_clrL	DMA
0x0073	0x300000E6	chnl_req_mask_clrH	DMA
0x0074	0x300000E8	chnl_enable_setL	DMA
0x0075	0x300000EA	chnl_enable_setH	DMA
0x0076	0x300000EC	chnl_enable_clrL	DMA
0x0077	0x300000EE	chnl_enable_clrH	DMA
0x0078	0x300000F0	chnl_pri_alt_setL	DMA
0x0079	0x300000F2	chnl_pri_alt_setH	DMA
0x007A	0x300000F4	chnl_pri_alt_clrL	DMA
0x007B	0x300000F6	chnl_pri_alt_clrH	DMA
0x007C	0x300000F8	chnl_priority_setL	DMA
0x007D	0x300000FA	chnl_priority_setH	DMA
0x007E	0x300000FC	chnl_priority_clrL	DMA
0x007F	0x300000FE	chnl_priority_clrH	DMA

6.4.3 Отображение памяти DSP в памяти RISC

Память подсистемы DSP (включая периферийные блоки) отображается в адресное пространство RISC в область DATA с адреса 0x30000000 по адрес 0x30040000. DMA RISC также имеет доступ к памяти DSP по тем же адресам. RISC-ядро и DMA RISC доступно все адресное пространство подсистемы DSP, кроме регистров ядра DSP. Соответствие карты памяти DSP и RISC показано на рисунке ниже. Ядро RISC использует байтовую адресацию, ядро DSP полусловную (16 бит). В таблице ниже (Таблица 6-8) представлено подробное отображение регистров периферийных модулей подсистемы DSP в памяти RISC.

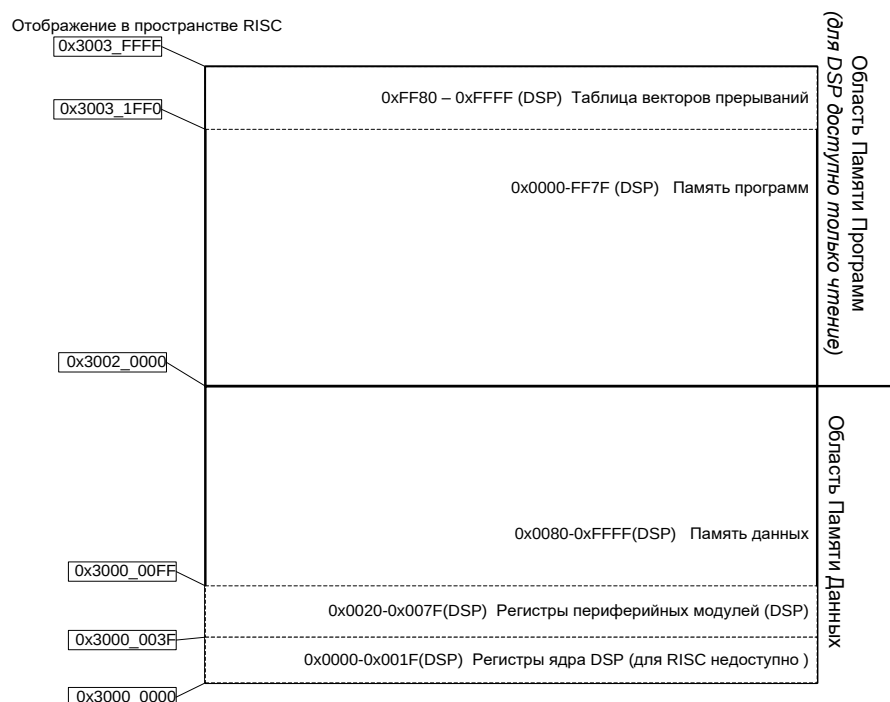


Рисунок 6-10 – Отображение памяти DSP в адресное пространство RISC

Таблица 6-8 – Отображение памяти DSP в адресное пространство RISC

Область памяти	Сектор памяти	Адреса RISC и тип доступа RISC	Адреса DSP и тип доступа DSP
Память данных	Регистры ядра DSP	0x3000_0000-0x3000_003F NA	0x0000-0x001F RW
	Регистры периферийных модулей	0x3000_0040-0x3000_00FF RW	0x0020-0x007F RW
	ОЗУ данных	0x3000_0100-0x3001_FFFF RW	0x0080-0xFFFF RW
Память программ	Программы пользователя	0x3002_0000-0x3003_FEFF RW	0x0000-0xFF7F RO
	Таблица векторов прерывания	0x3003_FF00-0x3003_FFFF RW	0xFF80-0xFFFF RO
Примечание – NA – нет доступа, RO – только чтение, RW – чтение и запись			

В данной версии кристалла используется два расслоенных модуля ОЗУ, состоящие из четырех блоков по 32 Кбайт каждый. Т.е. адресное пространство программ и данных DSP может быть разбито на четыре равные части каждое. При этом, обращение к любой из восьми частей (четыре части памяти программ и четыре части памяти данных) одним из устройств системы (ядром RISC, DMA RISC, ядром DSP или DMA DSP) не будет задержано, в случае, если все другие обращение в этот момент используют другие блоки ОЗУ DSP.

Например, если ядро DSP выполняет программу из первой четверти своей памяти программ, ядро RISC может без каких либо торможений системы записать новые функции для DSP во вторую четверть памяти программ. Одновременные обращения различных устройств к одному и тому же сектору ОЗУ программ или данных возможны, но приводят к необходимости ожидания освобождения доступа к ОЗУ, и, как следствие, к общему уменьшению быстродействия системы.

6.5 Загрузочное ПЗУ и режимы работы микросхемы

После включения питания и снятия внутренних (POR) и внешних (RESET) сигналов сброса, микросхема начинает выполнять программу из загрузочной области ПЗУ BOOT ROM. В загрузочной программе микросхема определяет, в каком из режимов она будет функционировать, и переходит в этот режим. Режим функционирования определяется внешними выводами MODE[2:0] (PF[6:4]), при этом перед опросом состояния этих выводов, для них включается внутренняя подтяжка к земле (встроенные резисторы подтяжки к земле имеют сопротивление ~50 кОм). Также устанавливается бит FPOR в регистре BKP_REG_0E, который может быть сброшен только при отключении основного питания U_{CC} . После перезапуска микросхемы уровни на выводах MODE[2:0] не влияют на режим функционирования микросхемы, если установлен бит FPOR. В пользовательской программе выводы PF[6:4] могут использоваться пользователем.

Таблица 6-9 – Режимы первоначального запуска микросхемы

MODE[2:0]	Режим	Стартовый адрес/таблица векторов прерываний	Описание
000	Микроконтроллер в режиме отладки	0x0800_0000	Микросхема начинает выполнять программу из внутренней FLASH-памяти программ. При этом установлен отладочный интерфейс JTAG_B
001	Микроконтроллер в режиме отладки	0x0800_0000	Микросхема начинает выполнять программу из внутренней FLASH-памяти программ. При этом разрешается работа отладочного интерфейса JTAG_A
010-011	Микропроцессор в режиме отладки	0x1000_0000	Микросхема конфигурирует внешнюю системную шину в режим работы ROM с Wait_States = 0xF и начинает выполнять программу из внешней памяти, установленной на внешней системной шине. При этом разрешается работа отладочного интерфейса JTAG_B
100	Зарезервировано	-	-
101	Зарезервировано	-	-
110	UART-загрузчик без отладки	Определяется пользователем	Микросхема через интерфейс UART3 на выводах PF[1:0] получает код программы в ОЗУ для исполнения. При этом отладочный интерфейс JTAG/SW заблокирован
111	Зарезервировано	-	-

При работе в режиме отладки разрешается работа отладочного интерфейса JTAG/SW. При этом к микросхеме может быть подключен JTAG/SW адаптер, с помощью которого программные средства разработки позволяют работать с микросхемой в отладочном режиме. Линии JTAG должны быть подтянуты к питанию сопротивлениями не менее 10 К с учетом, чтобы эти подтяжки не влияли на работу системы (Рисунок 6-11).

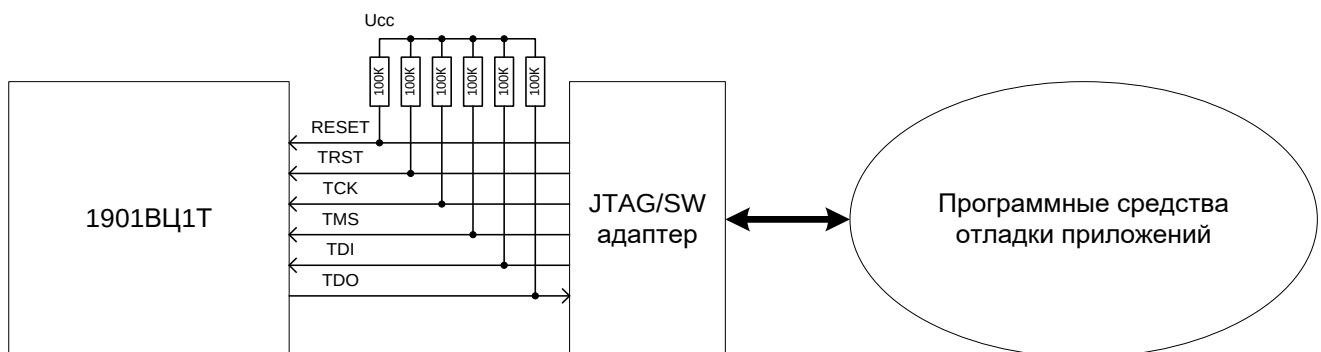


Рисунок 6-11 – Схема работы в режиме отладки

В отладочном режиме можно:

- стирать, записывать, считывать внутреннюю FLASH-память программ RISC;
- считывать и записывать содержимое ОЗУ и периферии;
- выполнять программу в пошаговом режиме;
- запускать программу в нормальном режиме;
- останавливать программу по точкам остановки;
- просматривать переменные выполняемой программы;
- проводить трассировку хода выполнения программного обеспечения.

В зависимости от режима работы выводы интерфейса JTAG/SW переопределяются на различные выводы микросхемы представленные в таблице ниже.

Таблица 6-10 – Переопределение выводов интерфейса JTAG/SW

Вывод JTAG/SW	Вывод микросхемы	Описание
JTAG_A		
TRST	PB4/DATA20/JA_TRST	В качестве выводов интерфейса используются выводы порта В, совмещенные с данными внешней системной шины, использование которых при отладке запрещено
TCK	PB2/DATA18/JA_TCK	
TMS	PB1/DATA17/JA_TMS	
TDI	PB3/DATA19/JA_TDI	
TDO	PB0/DATA16/JA_TDO	
JTAG_B		
TRST	PD2/ADC2/BUSY/SSP2RXD/JB_TRST	В качестве выводов интерфейса используются выводы порта D, совмещенные с каналами АЦП, выводами каналов Таймера 1, UART2 и SSP2, использование которых при отладке запрещено
TCK	PD1/ADC1_REF-/TMR1_CH1/ UART2_TXD/JB_TCK	
TMS	PD0/ADC0_REF+/TMR1_CH1N/ UART2_RXD/JB_TMS	
TDI	PD3/ADC3/CE/SSP2FSS/JB_TDI	
TDO	PD4/ADC4/TMR1_ETR/nSIROUT2/JB_TDO	

6.5.1 Микроконтроллер в режиме отладки

После определения данного режима работы (MODE[2:0] = 000 или 001) загрузочная программа выполняет следующие действия:

- разрешает работу отладочного интерфейса JTAG (регистр REG_0E):
 - при MODE[2:0] = 000 разрешается работа интерфейса JTAG_B;
 - при MODE[2:0] = 001 разрешается работа интерфейса JTAG_A.
- устанавливает адрес таблицы векторов в значение 0x0800_0000 (регистр VTOR);
 - устанавливает указатель стека в значение, считанное из ячейки с адресом 0x0800_0000 (таблица векторов, поле «Initial SP value»);
 - осуществляет безусловный переход по адресу, считанному из ячейки с адресом 0x0800_0004 (таблица векторов, поле «Reset»).

6.5.2 Микропроцессор в режиме отладки

После определения данного режима работы (MODE[2:0] = 010 или 011) загрузочная программа выполняет следующие действия:

- разрешает работу отладочного интерфейса JTAG_B (регистр REG_0E);
 - конфигурирует набор выводов для работы с внешней памятью:
 - шина данных D[31:0] ({PB[15:0], PA[15:0]});
 - шина адреса A[31:0] ({PE[15:0], PF[15:0]});
 - сигнал чтения данных OE (PC[1]);
 - сигнал записи данных WE (PC[2]);
 - сигналы выборки байтов BE[3:0] (PC[6:3]);
 - сигнал синхронизации CLOCK (PC[7]).
- конфигурирует контроллер внешней системной шины MDR_EBC:
 - режим работы – ROM (только чтение),
 - длительность фаз обращения WAIT_STATE[3:0] = 0xF,
 - полярность сигнала CLOCK – положительная.
- устанавливает адрес таблицы векторов в значение 0x1000_0000 (регистр VTOR);
 - устанавливает указатель стека в значение, считанное из ячейки с адресом 0x1000_0000 (таблица векторов, поле «Initial SP value»);
 - осуществляет безусловный переход по адресу, считанному из ячейки с адресом 0x1000_0004 (таблица векторов, поле «Reset»).

6.5.3 UART-загрузчик без отладки

Для загрузки программ в режиме UART-загрузчика (MODE[2:0] = 110) используется специальный протокол обмена, реализованный на периферийном модуле UART3. При этом для работы модуля используется выводы в соответствии с таблицей ниже (Таблица 6-11).

Таблица 6-11 – Используемые порты ввода/вывода UART-загрузчиком

Режим	Rx	Tx
110b	PF0	PF1

Данный режим работы предоставляет достаточный набор операций, необходимых для записи в ОЗУ какой-либо программы (в частности программатора FLASH-памяти), верификации ее и запуска на выполнение. Кроме того, существует возможность задания внешним устройством скорости обмена.

В качестве источника тактовой частоты UART3 используется внутренний RC-генератор HSI с частотой 8 МГц. Так как разброс значений частоты HSI (для различных образцов микросхем) весьма велик (от 6 до 10 МГц), то требуется этап подбора значения делителя частоты UART3 для синхронизации с внешним устройством.

После определения данного режима работы загрузочная программа запрещает работу отладочных интерфейсов JTAG_A и JTAG_B.

Параметры связи по UART

Для связи по UART выбраны следующие параметры канала связи:

- начальная скорость, [бод] – 13363 (BRDI = 4; BRDF = 40);
- количество бит данных – 8;
- четность – нет;
- количество Stop бит – 1.

Загрузчик не использует FIFO UART3.

Загрузчик всегда выступает в качестве Slave, а внешнее устройство, подающее команды – в качестве Master.

Данные передаются младшим битом вперед.

Протокол обмена по UART

После синхронизации с внешним устройством, подающим команды (Master), загрузчик переходит в диспетчер команд. Набор команд протокола представлен в таблице ниже (Таблица 6-12).

Таблица 6-12 – Команды загрузчика UART

Команда	Код	ASCII Символ	Описание
CMD_SYNC	0x00		Пустая команда. Загрузчик ее принимает, но ничего по ней не делает
CMD_CR	0x0D		Выдача приглашения Master-у
CMD_BAUD	0x42	'B'	Установка скорости обмена
CMD_LOAD	0x4C	'L'	Загрузка массива байт
CMD_VFY	0x59	'Y'	Выдача массива байт
CMD_RUN	0x52	'R'	Запуск программы на выполнение

6.5.3.1 Синхронизация с внешним устройством

Начальные условия

На этапе синхронизации с внешним устройством (Master) вывод Rx используется как вход.

Master постоянно посылает в канал синхросимвол – 0. Загрузчик подстраивает свою скорость таким образом что бы минимизировать ошибки обмена. Как только Загрузчик настроил скорость он переходит в диспетчер команд и выдает приглашение (3 байта 0x0D (перевод строки), 0x0A (возврат каретки), 0x3E ('>'),) Master-у.

Master завершает выдачу синхросимволов и, теперь, может подавать команды согласно протоколу обмена.

6.5.3.2 Команда CMD_SYNC

Пустая команда.

Загрузчик (Slave) ее принимает, но ничего по ней не делает. Код команды соответствует символу синхронизации.

Таблица 6-13 – Команда CMD_SYNC

Код команды	CMD_SYNC = 0x00
ASCII символ, соответствующий коду команды	нет
Количество параметров команды	0
Формат команды:	
Master: Выдает код команды CMD_SYNC	Slave: Если команда принята с ошибками, то выдает код ошибки ERR_CHN или ERR_CMD и завершает обработку текущей команды

6.5.3.3 Команда CMD_CR

Выдача приглашения Master-y

Таблица 6-14 – Команда CMD_CR

Код команды	CMD_CR = 0x0D
ASCII символ, соответствующий коду команды	нет
Количество параметров команды	0
Формат команды:	
Master: Выдает код команды CMD_CR	Slave: Если команда принята с ошибками, то выдает код ошибки ERR_CHN или ERR_CMD и завершает обработку текущей команды. Иначе выдает три байта: – код команды CMD_CR; – код 0x0A; – код 0x3E (ASCII символ '>')

6.5.3.4 Команда CMD_BAUD

Установка скорости обмена

Таблица 6-15 – Команда CMD_BAUD

Код команды	CMD_BAUD = 0x42
ASCII символ, соответствующий коду команды	'B'
Количество параметров команды	1
Параметр	Новое значение скорости обмена [бод]
Формат команды:	
Master: Выдает код команды CMD_BAUD	Slave: Если команда принята с ошибками, то выдает код ошибки ERR_CHN или ERR_CMD и завершает обработку текущей команды
Master: Выдает параметр	Slave: Если параметр принят с ошибками, то выдает код ошибки ERR_CHN или ERR_BAUD и завершает обработку текущей команды. Иначе: – выдает код команды CMD_BAUD; – устанавливает новое значение скорости обмена во время отправки кода команды CMD_BAUD

6.5.3.5 Команда CMD_LOAD

Загрузка массива байт в память микросхемы

Таблица 6-16 – Команда CMD_LOAD

Код команды	CMD_LOAD = 0x4C
ASCII символ, соответствующий коду команды	'L'
Количество параметров команды	2
Параметр 1	Адрес памяти приемника данных.
Параметр 2	Размер массива в байтах
Формат команды:	
Master: Выдает код команды CMD_LOAD	Slave: Если команда принята с ошибками, то выдает код ошибки ERR_CHN или ERR_CMD и завершает обработку текущей команды
Master: Выдает параметр 1.	Slave: Ожидает получения всех параметров
Master: Выдает параметр 2.	Slave: Если хотя бы один из параметров принят с ошибками, то выдает код ошибки ERR_CHN и завершает обработку текущей команды. Иначе выдает код команды CMD_LOAD
Master: Выдает массив байт младшим байтом вперед	Slave: Принимает массив байт. Если хотя бы один байт принят с ошибками, то выдает код ошибки ERR_CHN и завершает обработку текущей команды, не дожидаясь окончания принятия всего массива. По окончании принятия массива выдает код ответа REPLY_OK = 0x4B ('K')

6.5.3.6 Команда CMD_VFY

Выдача массива байт из памяти микросхемы

Таблица 6-17 – Команда CMD_VFY

Код команды	CMD_VFY = 0x59
ASCII символ, соответствующий коду команды	'Y'
Количество параметров команды	2
Параметр 1	Адрес памяти источника данных
Параметр 2	Размер массива в байтах
Формат команды:	
Master: Выдает код команды CMD_VFY	Slave: Если команда принята с ошибками, то выдает код ошибки ERR_CHN или ERR_CMD и завершает обработку текущей команды
Master: Выдает параметр 1	Slave: Ожидает получения всех параметров
Master: Выдает параметр 2	Slave: Если хотя бы один из параметров принят с ошибками, то выдает код ошибки ERR_CHN и завершает обработку текущей команды. Иначе: – выдает код команды CMD_VFY; – выдает массив байт младшим байтом вперед. – по окончании передачи массива выдает код ответа REPLY_OK = 0x4B ('K')

6.5.3.7 Команда CMD_RUN

Запуск программы на выполнение.

Таблица 6-18 – Команда CMD_RUN

Код команды	CMD_RUN = 0x52
ASCII символ, соответствующий коду команды	'R'
Количество параметров команды	1
Параметр	Адрес таблицы векторов загруженной программы
Формат команды:	
Master: Выдает код команды CMD_RUN	Slave: Если команда принята с ошибками, то выдает код ошибки ERR_CHN или ERR_CMD и завершает обработку текущей команды
Master: Выдает параметр.	Slave: Если параметр принят с ошибками, то выдает код ошибки ERR_CHN и завершает обработку текущей команды. Иначе: – выдает код команды CMD_RUN; – устанавливает значение MSP и PC согласно таблице векторов (NVIC не перепрограммируется) и, таким образом, Slave завершает свое выполнение. Передача управления загруженной программе происходит, не дожидаясь окончания отправки кода команды CMD_RUN

6.5.3.8 Прием параметров команды

Параметры команд – это четырех байтные числа.

Параметры передаются младшим байтом вперед.

В качестве значения параметра запрещено использовать число 0xFFFFFFFF.

Если при приеме параметра обнаружена аппаратная ошибка (UART установил в «1» какой-либо из флагов ошибки), то прием параметров не прекращается.

Анализ всех видов ошибок, связанных с передачей параметров, загрузчик производит только после принятия всех параметров команды.

6.5.3.9 Сообщения об ошибках

Сообщения об ошибках – это двух байтные последовательности символов. Первый символ всегда 0x45 ('E'). Второй символ определяет тип ошибки.

После выдачи сообщения об ошибке загрузчик переходит в режим ожидания следующей команды, поэтому Master после получения такого сообщения должен прекратить передачу байт, относящихся к текущей команде.

После принятия сообщения об ошибке Master должен подавать команду CMD_CR до тех пор, пока не получит корректный ответ, соответствующий этой команде.

Возможны следующие сообщения об ошибках: ERR_CHN, ERR_CMD, ERR_BAUD.

Ошибка ERR_CHN

Аппаратная ошибка UART.

Код ошибки 0x69 ('i').

Выдается, если UART установил в «1» один из аппаратных флагов ошибки при приеме очередного байта.

Ошибка ERR_CMD

Принята неизвестная команда.

Код ошибки 0x63 ('c').

Выдается диспетчером команд, если принят неизвестный код команды.

Ошибка ERR_BAUD

Принята неизвестная команда.

Код ошибки 0x62 ('b').

Выдается диспетчером команд, если по принятому от Master-а значению скорости обмена невозможно вычислить корректное значение делителя частоты UART.

7 Контроллер FLASH-памяти программ

Микросхема содержит встроенную FLASH-память программ для ядра RISC объемом 128 Кбайт основной памяти программ и 4 Кбайта информационной памяти.

В обычном режиме (бит CON = 0, регистр EEPROM_CMD) доступна основная память программ через системные шины I Code и D code для выборки инструкций и данных кода программы.

В режиме программирования (бит CON=1, регистр EEPROM_CMD) основная и информационная память доступны как периферийные устройства и могут быть использованы для нужд разработчика приложения. В режиме программирования программный код должен выполняться из области системной шины или ОЗУ. Выполнение программного кода из FLASH-памяти программ в режиме программирования невозможно.

7.1 Работа FLASH-памяти программ в обычном режиме

Скорость доступа во FLASH-память ограничена и составляет порядка 40 нс, в результате выдача новых значений из FLASH-памяти может происходить с частотой не более 25 МГц. Для того, что бы процессорное ядро могло получать новые инструкции на больших частотах в микросхеме реализуется FLASH-память с физической организацией 8К на 128 разрядов. Таким образом, за 40 нс из FLASH-памяти извлекается 16 байт, в которых может быть закодировано от четырех до восьми инструкций процессора. И пока ядро выполняет эти инструкции из памяти извлекается следующая порция данных. Таким образом, тактовая частота может превышать частоты извлечения данных из памяти в несколько раз при линейном выполнении программы. При возникновении переходов в выполнении программы, когда из памяти программ не выбраны нужные инструкции возникает пауза в несколько тактов процессора для того что бы данные успели считаться из FLASH. Число тактов паузы зависит от тактовой частоты процессора, так при работе с частотой ниже 25 МГц пауза не требуется, так как FLASH-память успевает выдать новые данные за один такт, при частоте от 25 до 50 МГц требуется один такт паузы, и так далее. Число тактов паузы задается в регистре EEPROM_CMD битами Delay[2:0]. В таблице приведены характеристики необходимой паузы для работы FLASH-памяти программ.

Таблица 7-1 – Дополнительная пауза для работы FLASH-памяти

Delay[2:0]	Тактов паузы	Тактовая частота	Примечание
0x00	0	До 25 МГц	
0x01	1	До 50 МГц	
0x02	2	До 75 МГц	
0x03	3	До 100 МГц	Работа микросхемы с частотой более 100 МГц не гарантируется
0x04	4	До 125 МГц	
0x05	5	До 150 МГц	
0x06	6	До 175 МГц	
0x07	7	До 200 МГц	Установлено по умолчанию после сброса

Число тактов паузы устанавливается до момента повышения тактовой частоты или после снижения тактовой частоты.

Также в контроллере EEPROM реализованы отдельные кэш для данных и для инструкций, объемом 16×4 32-битных слов каждый. В том случае, если запрашиваемая по шине I-bus команда или запрашиваемые по шине D-bus данные содержаться в кэш команд или данных соответственно, ядро получает требуемое слово без задержек.

Кэши команд и данных могут быть включены битами DCEN и ICEN регистра EEPROM_CTRL соответственно. После сброса значение обоих бит равно 0 (кэш выключен). В том случае, если кэш выключен, он никак не влияет на работу механизма доступа к данным FLASH-памяти.

7.2 Работа FLASH-памяти программ в режиме программирования

В режиме программирования FLASH-память программ не может выдавать инструкции и данные процессору, поэтому перевод памяти в режим программирования (установка бита CON = 1) возможен только программой, исполняемой из памяти, установленной на внешней системной шине, или ОЗУ. Перед переводом памяти в режим программирования необходимо в регистр EEPROM_KEY записать комбинацию 0x8AAA5551.

В режиме программирования возможны следующие операции как с основной (бит IFREN = 0, регистр EEPROM_CON), так и с информационной (бит IFREN = 1) памятью:

- стирание всей памяти;
- стирание страницы памяти размером 4 Кбайт;
- запись 32-битного слова в память;
- чтение 32-битного слова из памяти.

Внимание! Нельзя повторять циклы стирания – записи и стирания – стирания одной ячейки памяти с периодом менее 4 мс.

Страница 31	0x0801_FFFC ...	0x0801_FFF8 ...	0x0801_FFF4 ...	0x0801_FFF0 ...
256 × 128 4K × 8	0x0801_F00C	0x0801_F008	0x0801_F004	0x0801_F000
	...	...	...	...
Страница 1	0x0800_1FFC ...	0x0800_1FF8 ...	0x0800_1FF4 ...	0x0800_1FF0 ...
256 × 128 4K × 8	0x0800_100C	0x0800_1008	0x0800_1004	0x0800_1000
Страница 0	0x0800_0FFC ...	0x0800_0FF8 ...	0x0800_0FF4 ...	0x0800_0FF0 ...
256 × 128 4K × 8	0x0800_001C 0x0800_000C	0x0800_0018 0x0800_0008	0x0800_0014 0x0800_0004	0x0800_0010 0x0800_0000
	Sector_D	Sector_C	Sector_B	Sector_A
	256 × 32 1K × 8	256 × 32 1K × 8	256 × 32 1K × 8	256 × 32 1K × 8

Основная память (IFREN=0)

Страница 0	0x0800_0FFC	0x0800_0FF8	0x0800_0FF4	0x0800_0FF0
	...	...	...	...
256 × 128	0x0800_001C	0x0800_0018	0x0800_0014	0x0800_0010
4K × 8	0x0800_000C	0x0800_0008	0x0800_0004	0x0800_0000
	Sector_D	Sector_C	Sector_B	Sector_A
	256 × 32	256 × 32	256 × 32	256 × 32
	1K × 8	1K × 8	1K × 8	1K × 8

Информационная память (IFREN=1)

Рисунок 7-1 – Структура FLASH-памяти

7.2.1 Стирание всей памяти

Стирание всей памяти выполняется в четыре этапа:

- 1 этап – стирание Sector_A для всей памяти;
- 2 этап – стирание Sector_B для всей памяти;
- 3 этап – стирание Sector_C для всей памяти;
- 4 этап – стирание Sector_D для всей памяти.

Стирание одного сектора памяти возможно только в режиме программирования. Для стирания одного сектора памяти надо установить необходимое значение в бит IFREN (1 – для основной и информационной памяти и 0 – для основной памяти), и номер сектора EEPROM_ADR[3:2] (00 – Sector_A, 01 – Sector_B, 10 – Sector_C и 11 – Sector_D) затем установить биты XE, MAS1 и ERASE в «1», и спустя время t_{nvs} (не менее 5 мкс) установить бит NVSTR в «1». Полное стирание памяти длится время t_{me} (от 20 мс до 40 мс). Спустя это время необходимо очистить бит ERASE, и спустя время t_{nvhl} (не менее 100 мкс) очистить биты XE, MAS1 и NVSTR. Последующие операции с памятью можно выполнять спустя время t_{rcv} (не менее 1 мкс). Временная диаграмма стирания памяти представлена далее (см. Рисунок 7-2). При стирании информационной области, автоматически стирается и основная. Для стирания всей памяти необходимо осуществить стирание всех секторов памяти.

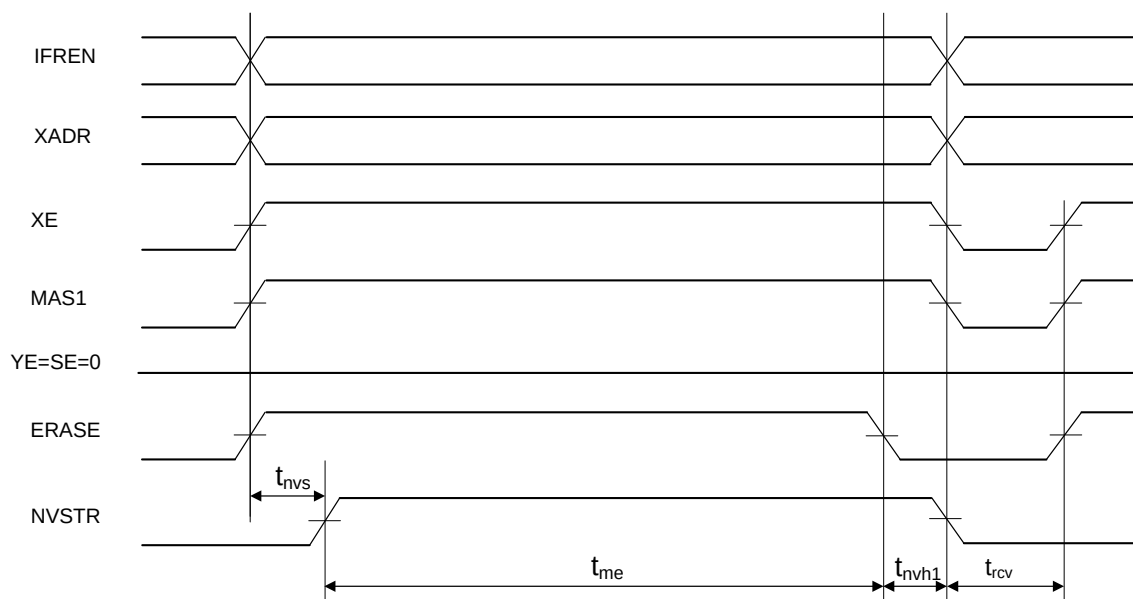


Рисунок 7-2 – Временная диаграмма стирания памяти

7.2.2 Стирание страницы памяти размером 4 Кбайт

Стирание одной страницы выполняется в четыре этапа:

- 1 этап - стирание Sector_A для одной страницы;
- 2 этап - стирание Sector_B для одной страницы;
- 3 этап - стирание Sector_C для одной страницы;
- 4 этап - стирание Sector_D для одной страницы.

Стирание одного сектора страницы памяти возможно только в режиме программирования. Для стирания страницы памяти надо установить необходимое значение в бит IFREN (1 – для информационной памяти и 0 – для основной памяти), затем установить адрес стираемой страницы в регистре EEPROM_ADR[16:12] и номер сектора EEPROM_ADR[3:2] (00 – Sector_A, 01 – Sector_B, 10 – Sector_C и 11 – Sector_D) и установить биты XE и ERASE в «1», и спустя время t_{nvs} (не менее 5 мкс) установить бит NVSTR в «1». Стирание страницы памяти длится время t_{erase} (от 20 мс до 40 мс). Спустя это время необходимо очистить бит ERASE, и спустя время t_{nvh} (не менее 5 мкс) очистить биты XE и NVSTR. Последующие операции с памятью можно выполнять спустя время t_{rcv} (не менее 1 мкс). Временная диаграмма стирания страницы памяти представлена далее (см. Рисунок 7-3). Для стирания всей страницы необходимо осуществить стирание всех секторов страницы.

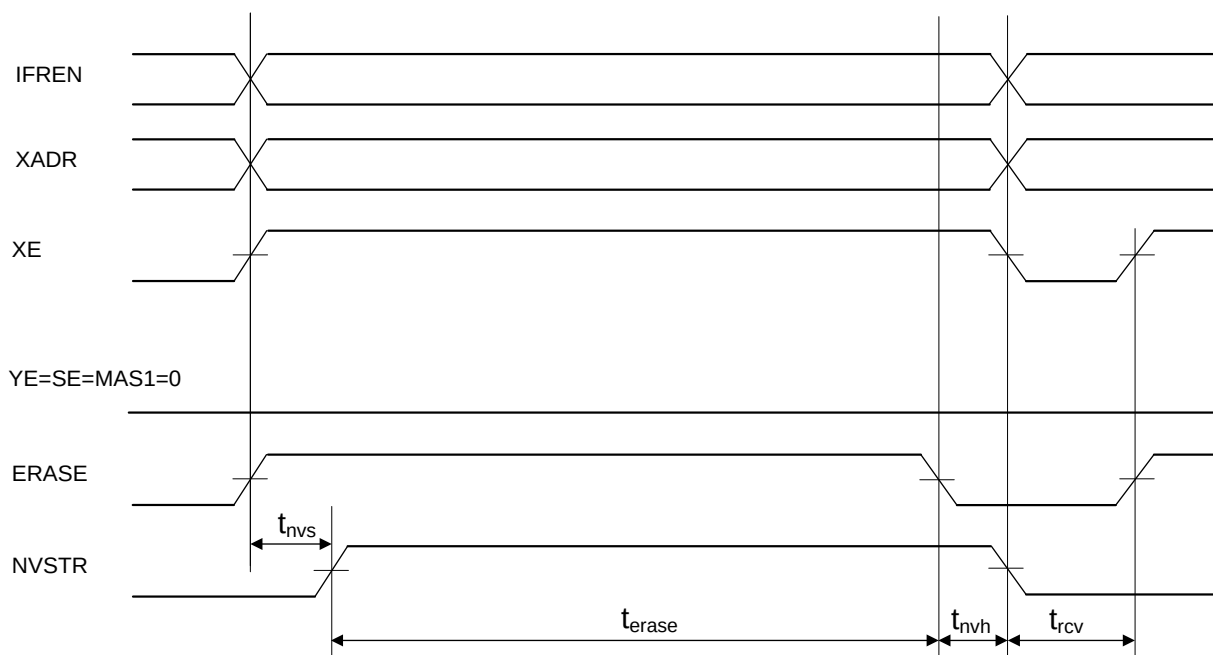


Рисунок 7-3 – Временная диаграмма стирания банка памяти

7.2.3 Запись 32-битного слова в память

Запись в память возможна только в режиме программирования.

Примечание – перед программированием ячейки необходимо выполнить ее стирание. Повторное программирование ранее запрограммированной ячейки уменьшает ресурс циклов записи/стирания.

Для записи в память надо установить необходимое значение в бит IFREN (1 – для информационной памяти и 0 – для основной памяти), затем установить адрес, по которому производится запись, в регистре EEPROM_ADR, в регистр EEPROM_DI поместить записываемое в память слово и установить биты XE и PROG в «1», и спустя время t_{nvs} (не менее 5 мкс) установить бит NVSTR в «1». Спустя время t_{pgs} (не менее 10 мкс) установить бит YE в «1». Запись в память длится время t_{prog} (от 20 мкс до 40 мкс). Спустя это время необходимо очистить бит YE, и через t_{adh} (не менее 20 нс) установить новый адрес (в пределах того же сектора и той же страницы) и значение для записи в другую ячейку памяти; затем через t_{ads} (не менее 20 нс) установить YE в «1» и записать следующее слово.

Примечание – В одном цикле допускается запись в пределах одного сектора одной страницы.

Если запись больше не требуется, то спустя время t_{pgh} (не менее 20 нс) после очистки бита YE необходимо очистить бит PROG и спустя время t_{nvh} (не менее 5 мкс) очистить биты XE и NVSTR. Последующие операции с памятью можно выполнять спустя время t_{rcv} (не менее 1 мкс).

Временная диаграмма записи памяти представлена ниже (см. Рисунок 7-4).

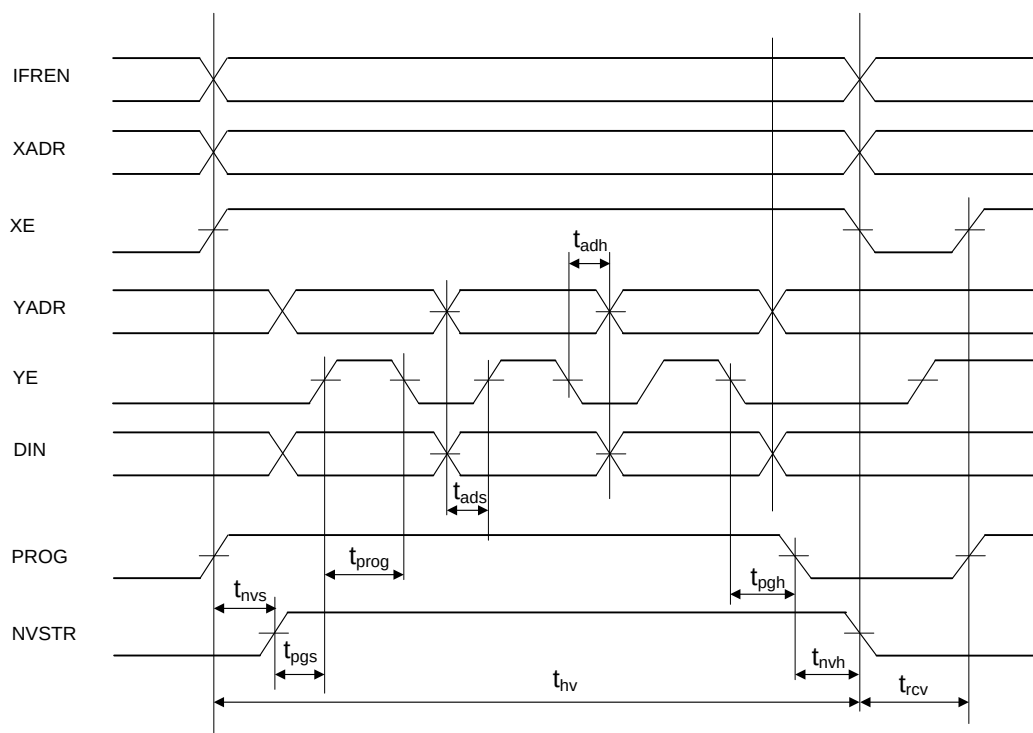


Рисунок 7-4 – Временная диаграмма записи памяти

7.2.4 Чтение 32-битного слова из памяти

В обычном режиме работы для чтения доступна только основная память, необходимо просто считать требуемый адрес памяти.

В режиме программирования для чтения доступна и основная, и информационная память. Для чтения из памяти надо установить необходимое значение в бит IFREN (1 – для информационной памяти и 0 – для основной памяти), затем установить адрес, из которого необходимо считать данные, в регистре EEPROM_ADR и установить биты XE, YE и SE в «1», и спустя время $t_{ха}$ (не менее 30 нс) из регистра EEPROM_DO можно считать

данные. Если необходимо считать следующее слово, то в регистр EEPROM_ADR следует записать новый адрес и, спустя время t_{xa} (не менее 30 нс), из регистра EEPROM_DO можно считать следующие данные. Если чтение больше не требуется, то можно очистить все биты управления. Временная диаграмма чтения памяти в режиме программирования представлена далее (см. Рисунок 7-5).

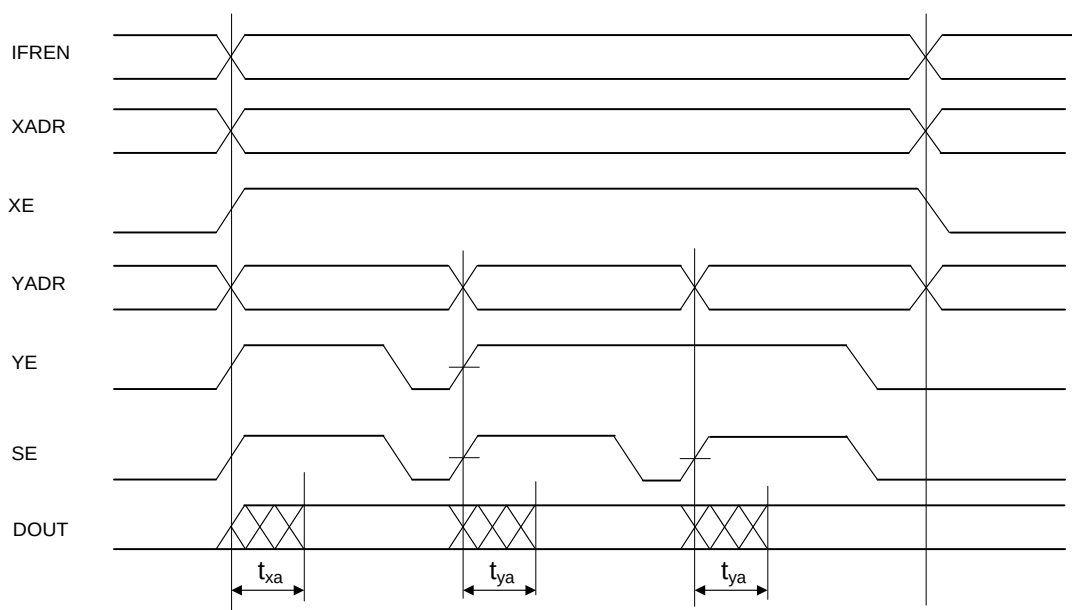


Рисунок 7-5 – Временная диаграмма чтения памяти

7.3 Описание регистров управления контроллера FLASH-памяти программ

Таблица 7-2 представляет перечень регистров управления контроллера FLASH-памяти программ.

Таблица 7-2 – Регистры управления контроллера FLASH-памяти программ

Базовый адрес	Название	Описание
0x4001_8000	MDR_EEPROM	Регистры контроллера FLASH-памяти программ
Смещение		
0x00	CMD	Регистр команды
0x04	ADR	Регистр адреса
0x08	DI	Регистр данных на запись
0x0C	DO	Регистр данных считанных
0x10	KEY	Регистр ключа
0x14	CTRL	Регистр управления

Далее каждый из регистров управления контроллера рассмотрен отдельно.

Обозначения:

R/W – бит доступен на чтение и запись;

RO – бит доступен только на чтение;

U – бит физически не реализован или зарезервирован.

7.3.1 MDR_EEPROM->CMD

Таблица 7-3 – Регистр команды EEPROM_CMD

Номер	31...14	13	12	11	10	9
Доступ	U	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0
		NVSTR	PROG	MAS1	ERASE	IFREN

Номер	8	7	6	5...3	2, 1	0
Доступ	R/W	R/W	R/W	R/W	U	R/W
Сброс	0	0	0	100	0	0
	SE	YE	XE	Delay[2:0]		CON

Таблица 7-4 – Описание бит регистра EEPROM_CMD

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...14	-	Зарезервировано
13	NVSTR	Операции записи или стирания: 0 – при чтении; 1 – при записи или стирании
12	PROG	Записать данные по ADR[16:2] из регистра EEPROM_DI: 0 – нет записи; 1 – есть запись
11	MAS1	Стереть весь блок, при ERASE = 1: 0 – стирается страница с адресом ADR[16:12] в секторе с адресом ADR[3:2]; 1 – стирается весь сектор с адресом ADR[3:2]
10	ERASE	Стереть строку с адресом ADR[16:9], ADR[8:0] значения не имеет: 0 – нет стирания; 1 – стирание
9	IFREN	Работа с блоком информации: 0 – основная память; 1 – информационный блок
8	SE	Усилитель считывания: 0 – не включен; 1 – включен
7	YE	Выдача адреса ADR[8:2]: 0 – не разрешено; 1 – разрешено
6	XE	Выдача адреса ADR[16:9]: 0 – не разрешено; 1 – разрешено
5...3	Delay[2:0]	Задержка памяти программ при чтении в циклах (в рабочем режиме): 000 – 0 цикл; 001 – 1 цикл; 111 – 7 циклов

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
2, 1	-	Зарезервированно
0	CON	Переключение контроллера памяти EEPROM на регистровое управление, не может производиться при исполнении программы из области EEPROM: 0 – управление EEPROM от ядра, рабочий режим; 1 – управление от регистров, режим программирования

7.3.2 MDR_EEPROM->ADR

Таблица 7-5 – Регистр адреса EEPROM_ADR

Номер	31...0
Доступ	R/W
Сброс	0
	ADR [31:0]

Таблица 7-6 – Описание бит регистра адреса EEPROM_ADR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...0	ADR[31:0]	Адрес обращения в память: ADR[1:0] – не имеет значения, минимально адресуемая ячейка 32 бита

7.3.3 MDR_EEPROM->DI

Таблица 7-7 – Регистр записываемых данных EEPROM_DI

Номер	31...0
Доступ	R/W
Сброс	0
	DATA [31:0]

Таблица 7-8 – Описание бит регистра записываемых данных EEPROM_DI

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...0	DATA[31:0]	Данные для записи в EEPROM

7.3.4 MDR_EEPROM->DO

Таблица 7-9 – Регистр считываемых данных EEPROM_DO

Номер	31...0
Доступ	R/W
Сброс	0
	DATA [31:0]

Таблица 7-10 – Описание бит регистра считываемых данных EEPROM_DO

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...0	DATA[31:0]	Данные, считанные из EEPROM

7.3.5 MDR_EEPROM->KEY

Таблица 7-11 – Регистр ключа EEPROM_KEY

Номер	31...0
Доступ	R/W
Сброс	0
	KEY [31:0]

Таблица 7-12 – Описание бит регистра ключа EEPROM_KEY

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...0	KEY[31:0]	Ключ для разрешения доступа к FLASH-памяти через регистровый доступ. Перед переводом памяти в режим программирования необходимо в регистр EEPROM_KEY записать комбинацию 0x8AAA5551

7.3.6 MDR_EEPROM-> CTRL

Таблица 7-13 – Регистр управления EEPROM_CTRL

Номер	31...3	1	0
Доступ	U	R/W	R/W
Сброс	0	0	0

Таблица 7-14 – Описание бит регистра EEPROM_CTRL

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...2	-	Зарезервировано
1	DCEN	Включение кеш данных 0 – выключено; 1 – включено;
0	ICEN	Включение кеш инструкций 0 – выключено; 1 – включено;

8 Процессорное ядро RISC

8.1 Структурная схема процессорного ядра RISC

- Процессорное ядро RISC – высокопроизводительный 32-разрядный процессор, разработанный для микроконтроллерных систем;
- Высокая производительность скомбинирована с быстрой обработкой прерываний;
- Расширенная система отладки с точками остановки и трассировки;
- Эффективное процессорное ядро для работы системы и памяти;
- Сверхнизкое потребление с встроенными режимами sleep;
- Защищенная система с интегрированным блоком защиты памяти MPU.

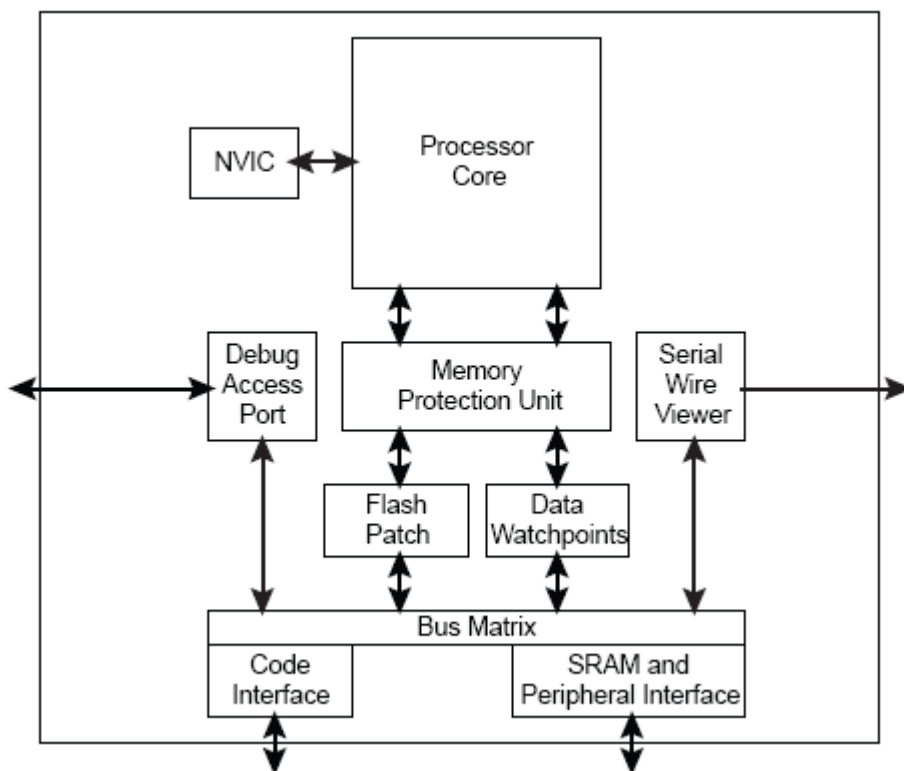


Рисунок 8-1 – Структурная схема процессорного ядра

Процессор построен на высокопроизводительном ядре с трех стадийным конвейером и Гарвардской архитектурой, что делает его идеальным для микроконтроллерных приложений. Процессор предлагает превосходную энергоэффективность, эффективный набор инструкций, оптимальный дизайн аппаратных средств, включающих одноктактную инструкцию умножения 32×32 и аппаратное деление. Процессор содержит интегрированный контроллер прерываний и встроенные средства отладки. Процессор реализует набор инструкций Thumb2, обеспечивающих высокую плотность кода и более экономное использование памяти программ. Процессор обеспечивает производительность 32-битных архитектур с размером кода, сравнимым с 8- и 16-битными микроконтроллерами.

Процессор содержит контроллер прерываний NVIC, обеспечивающий высокоскоростную обработку прерываний. NVIC обеспечивает до 16-ти уровней приоритетов прерываний. Интеграция контроллера прерываний в ядро позволяет

реализовать быстрое исполнение обработчиков прерываний (interrupt service routines – ISR), эффективно снижающее задержку обработки прерываний. Это обеспечивается аппаратным сохранением в стеке регистров, выполняемым одной инструкцией множественной записи и считывания памяти. Реализация обработчиков прерываний не требует их описания на ассемблере и позволяет удалить из обработчика код по перегрузке контекста. Оптимизация сцепления концов обработчиков позволяет снизить затраты при переключении с одного обработчика на другой.

Оптимизированный для пониженного энергопотребления контроллер NVIC обеспечивает режимы Sleep и Deep Sleep, которые позволяют быстро снизить потребление.

Ядро RISC обеспечивает большую скорость и низкую задержку обращения в память. Также поддерживаются невыровненные обращения и битовые манипуляции с областью ОЗУ и регистрами периферии.

Ядро RISC содержит блок защиты памяти (MPU) который обеспечивает граничное управление памятью, позволяющий приложениям реализовывать различные уровни привилегий безопасности, разделяя код, данные и стеки для различных задач, что требуется для критичных к сбоям решений.

Ядро RISC реализует аппаратную поддержку функций отладки. Отладка позволяет отображать состояние системы и памяти через стандартный разъем JTAG или двухпроводной интерфейс SWD.

Для трассировки в ядре реализован модуль ITM, отслеживающий точки просмотра данных и сообщения профилирования.

Периферийными блоками ядра являются:

- контроллер прерываний NVIC

Реализует высокоскоростную обработку прерываний

- блок системного управления SBC

Программный интерфейс процессора, реализует отображение информации о системной реализации, а также управление системой, включая конфигурирование, управление и отображение событий в системе

- системный таймер SysTick

24-битный счетчик, считающий вниз, используется операционными системами реального времени для подсчета тактов или как обычный счетчик

- блок защиты памяти MPU

Используется для повышения надежности системы путем задания различных атрибутов для регионов памяти. Поддерживает до 8-ми различных регионов и один опциональный предопределенный регион.

8.2 Программная модель

Процессор может функционировать в режимах:

- Thread

Используется для исполнения приложений, процессор находится в этом режиме сразу после сброса

- Handler

Используется для обработки исключений. После обработки исключения процессор переходит в режим Thread.

Уровни привилегий при исполнении программ:

- Unprivileged

Программное обеспечение:

- имеет ограниченный доступ к инструкциям MSR и MRS, и не может использовать инструкцию CPS;

- не имеет доступа к системному таймеру, NVIC и блоку системного управления;
- может иметь пониженный уровень доступа к памяти или периферии.

Непривилегированное программное обеспечение выполняется с уровнем *unprivileged*.

- Privileged

Программное обеспечение имеет полный доступ ко всем инструкциям и ресурсам.

Привилегированное программное обеспечение выполняется с уровнем *privileged*.

В режиме *Thread* регистр CONTROL определяет уровень исполнения программы *unprivileged* или *privileged*. Подробнее в описании регистра CONTROL. В режиме *handler* программное обеспечение всегда выполняется на *privileged* уровне.

Только привилегированное программное обеспечение может писать в регистр CONTROL для изменения уровня исполнения программы в режиме *Thread*. Непривилегированное программное обеспечение может использовать инструкцию SVC для выполнения *supervisor call* для передачи управления привилегированной программе.

8.3 Стек

Процессор использует нисходящий стек. Это означает, что указатель стека обозначает последний сохраненный в стеке элемент в стековой области памяти. Когда процессор записывает новый элемент в стек, сначала декрементируется указатель и затем записывается новый элемент в память. Процессор реализует два стека – *main* и *process* с независимыми указателями стеков, подробнее смотрите указатели стека.

В режиме *Thread* регистр CONTROL определяет, какой стек используется – *main* или *process*, подробнее в описании CONTROL регистра. В режиме *Handler* процессор всегда использует *main* стек.

Таблица 8-1 – Режимы работы процессора при выполнении программы

Режим процессора	Использование	Уровни привилегии для программного обеспечения	Используемый стек
Thread	Выполнение приложений	Privileged или Unprivileged*	Main или Process*
Handler	Обработка исключений	Всегда Privileged	Main стек
* Подробнее см. описание регистра CONTROL			

8.4 Регистры ядра

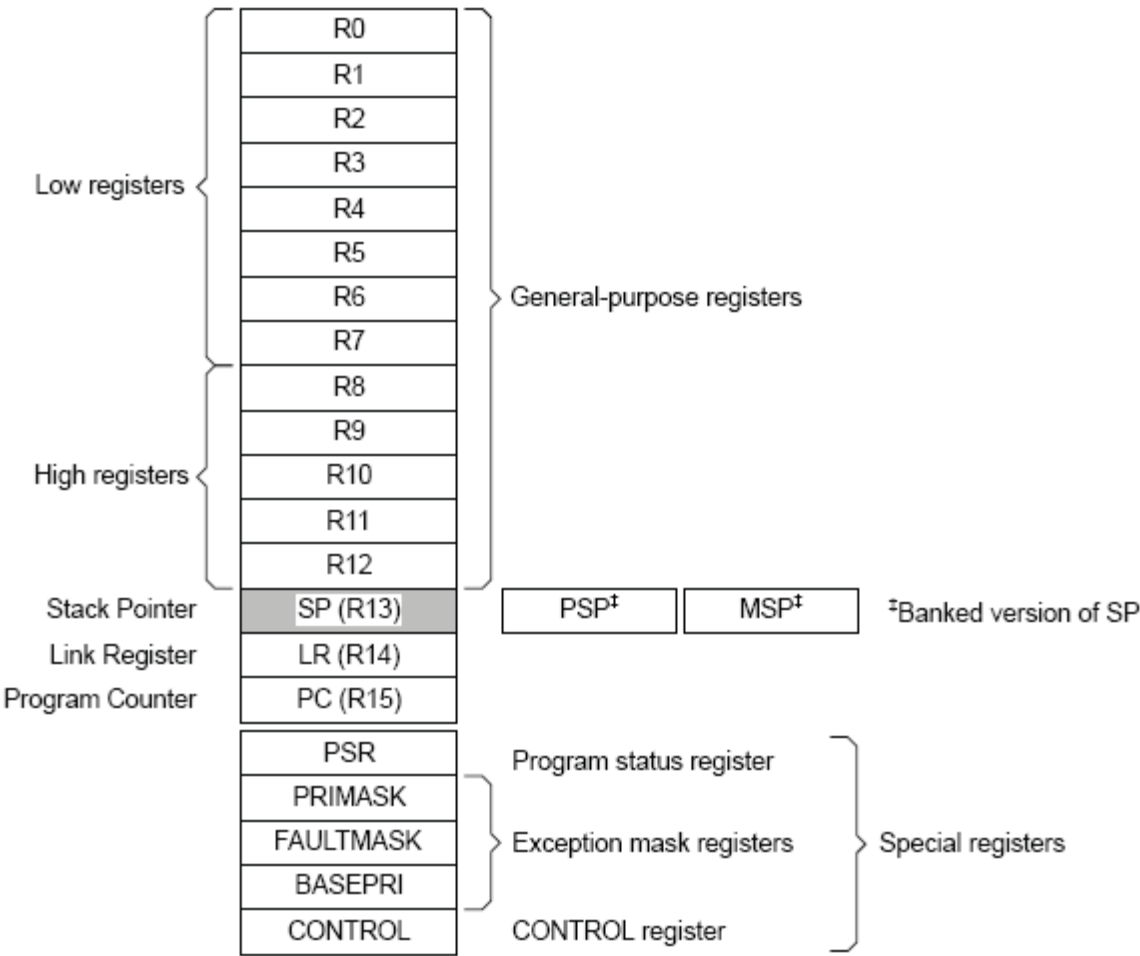


Рисунок 8-2 – Регистры ядра

Таблица 8-2 – Сводная таблица регистров ядра

Название	Тип*	Требуемый уровень привилегий	Значение после сброса	Описание
R0-R12	RW	Оба**	Неизвестно	Регистры общего назначения
MSP	RW	Privileged	См. описание	Указатель стека main Stack Pointer
PSP	RW	Оба**	Неизвестно	Указатель стека process Stack Pointer

Название	Тип*	Требуемый уровень привилегий	Значение после сброса	Описание
LR	RW	Оба**	0xFFFFFFFF	Регистр связи Link Register
PC	RW	Оба**	См. описание	Счетчик команд Program Counter
PSR	RW	Privileged	0x01000000	Программный регистр состояния Program Status Register
ASPR	RW	Оба**	0x00000000	Программный регистр состояния приложения Application Program Status Register
IPSR	RO	Privileged	0x00000000	Программный регистр состояния прерываний Interrupt Program Status Register
ESPR	RO	Privileged	0x01000000	Программный регистр состояния выполнения Execution Program Status Register
PRIMASK	RW	Privileged	0x00000000	Регистр маски приоритетов Priority Mask Register
FAULTMASK	RW	Privileged	0x00000000	Регистр маски сбоев Fault Mask Register
BASEPRI	RW	Privileged	0x00000000	Регистр базового приоритета маски Base Priority Mask Register
CONTROL	RW	Privileged	0x00000000	Регистр Управления CONTROL Register
* Определяет режим доступа при исполнении программы в режимах <i>thread</i> и <i>handler</i>. В режиме отладки может отличаться; ** Регистр доступен при исполнении программы с обоими уровнями привилегий				

8.4.1 Регистры общего назначения R0-R12

R0-R12 – это 32-разрядные регистры для данных при выполнении операций.

8.4.2 Указатель стека SP R13

Stack Pointer Register (SP) – это регистр R13. В Thread режиме бит 1 регистра CONTROL обозначает, какой указатель стека используется:

0 – Main Stack Pointer (MSP). Сразу после сброса;

1 – Process Stack Pointer (PSP).

При сбросе в MSP устанавливается 0x00000000.

8.4.3 Регистр связи LR R14

Link Register – это регистр R14. Регистр используется для сохранения информации об адресе возврата при уходе на обработку прерываний, вызовах функций и обработке исключений. При сбросе устанавливается в 0xFFFFFFFF.

8.4.4 Счетчик команд PC R15

Program Counter – это регистр R15. Он содержит адрес текущей инструкции. Бит 0 всегда 0, так как все инструкции выровнены на полуслово. При сбросе процессор считывает в этот регистр вектор сброса, который расположен по адресу 0x00000004.

8.4.5 Программный регистр состояния PSR

Регистр Program Status Register (PSR) объединяет регистры:

- Application Program Status Register (APSR)
- Interrupt Program Status Register (IPSR)
- Execution Program Status Register (EPSR)

Эти регистры разделяют различные битовые поля в 32-разрядном PSR. Описание регистров приведено ниже. Доступ к этим регистрам может быть как индивидуальный, так и комбинированный к двум или всем трем разом, с использованием имен регистров в качестве аргументов инструкций MSR или MRS.

Например:

- читать все регистры, используя PSR с инструкцией MRS;
- записать только в APSR, используя APSR с инструкцией MSR.

Таблица 8-3 – Комбинация PSR и их атрибуты

Регистр	Тип	Комбинация
PSR	RW ^{1), 2)}	APSR, EPSR и IPSR
IEPSR	RO	EPSR и IPSR
IAPSR	RW ¹⁾	APSR и IPSR
EAPSR	RW ²⁾	APSR и EPSR
¹⁾ Игнорируется запись в биты IPSR; ²⁾ При чтении битов EPSR читаются нули, и запись в них игнорируется		

Подробнее в описании инструкции MRS и MSR.

8.4.6 Программный регистр состояния приложения APSR

Регистр APSR содержит текущие флаги состояния выполнения предыдущей инструкции.

Таблица 8-4 – Регистр APSR

Номер	31	30	29	28	27	26...0
Доступ	R/W	R/W	R/W	R/W	R/W	
Сброс	0	0	0	0	0	
	N	Z	C	V	Q	-

Таблица 8-5 – Описание бит регистра APSR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31	N	Negative: 0 – результат операции положительный или нулевой, либо «больше чем или равно»; 1 – результат операции отрицательный, либо «меньше чем»
30	Z	Zero: 0 – результат операции не нулевой; 1 – результат операции нулевой
29	C	Carry: 0 – при суммировании не было переноса, либо при вычитании не было заема; 1 – при суммировании был перенос, либо при вычитании был заем
28	V	Overflow: 0 – в результате операции не было переполнения; 1 – в результате операции было переполнение
27	Q	Saturation: 0 – обозначает, что не было накопления с момента сброса либо с момента установки бита в ноль; 1 – обозначает, что в результате выполнения инструкций SSAT и USAT было накопление. Флаг сбрасывается в ноль программно инструкцией MRS
26...0	-	Зарезервировано

8.4.7 Программный регистр состояния прерываний IPSR

Регистр IPSR содержит номер типа исключения для текущего обработчика прерывания.

Таблица 8-6 – Регистр IPSR

Номер	31...9	8...0
Доступ	-	RO
Сброс	-	0
	-	ISR_NUMBER

Таблица 8-7 – Описание бит регистра IPSR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...9	-	Зарезервировано
8...0	ISR_NUMBER	Номер текущего исключения: 0 – Thread режим; 1 – зарезервировано; 2 – NMI; 3 – Hard Fault;

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
		4 – Memory Management Fault; 5 – Bus Fault; 6 – Usage Fault; 7...10 – зарезервировано; 11 – SVCcall; 12 – зарезервировано для отладки; 13 – PendSV; 15 – SysTick; 16 – IRQ0; ... 48 – IRQ31. Более подробно в разделе 36 «Прерывания и исключения»

8.4.8 Программный регистр состояния выполнения EPSR

Регистр EPSR содержит бит состояния инструкции Thumb, и биты состояния выполнения для инструкций:

- блок инструкций If-Then (IT);
- поле Interruptible-Continuable Instruction (ICI) для прерываемых инструкций

множественного сохранения и считывания

Таблица 8-8 – Регистр EPSR

Номер	31...27	26...25	24	23...16	15...10	9...0
Доступ		RO	RO		RO	
Сброс		0	1		0	
	-	ICI/IT	T	-	ICI/IT	-

Таблица 8-9 – Описание бит регистра EPSR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...27	-	Зарезервировано
26...25	ICI/IT	ICI: Биты Interruptible-Continuable Instruction; IT: Обозначает выполнение инструкции IT
24	T	Всегда 1
23...16	-	Зарезервировано
15...10	ICI/IT	ICI: Биты Interruptible-Continuable Instruction; IT: Обозначает выполнение инструкции IT
9...0	-	Зарезервировано

Попытка читать EPSR напрямую приложением, используя инструкцию MSR, всегда возвращает ноль. Попытка записать EPSR напрямую приложением, используя MSR, игнорируется. Обработчик сбойной ситуации может считать значение EPSR, сохранив его в стеке для отображения операции, вызвавшей сбой.

Interruptible-Continuable Instruction

Когда происходит прерывание при выполнении инструкций LDM или STM, то процессор:

- временно останавливает операцию множественного чтения или записи;
- сохраняет номер следующий регистр операнда в множественной операции в битах EPSR[15:12].

После обработки прерывания процессор:

- возвращает номер регистра для сохранения из бит EPSR[15:12];
- возобновляет выполнение операции множественного чтения или записи.

Когда EPSR содержит состояние выполнения ICI инструкции, то биты [26:25] и [11:10] содержат нули.

Блок инструкций If-Then

Блок If-Then содержит до четырех инструкций, следующих за 16-битной инструкцией IT. Каждая инструкция в этом блоке условная. Условие для инструкций может быть одним либо обратным для некоторых из них. Подробнее смотрите в пункте 9.8.1 «IT».

8.4.9 Регистр маски исключений Exception mask

Регистр маски исключений запрещает обработку исключений процессором. Запрещение исключений может потребоваться в критичных по времени задачах.

Для доступа к регистру маски исключений используются инструкции MSR и MRS, или инструкция CPS для изменения значения PRIMASK и FAULTMASK. Подробнее в см. в пунктах 9.9.1 «CPS», 9.9.5 «MRS» и 9.9.6 «MSR».

8.4.10 Регистр маски приоритетов Priority Mask

Регистр PRIMASK предотвращает активацию всех исключений с конфигурируемым приоритетом.

Таблица 8-10 – Регистр PRIMASK

Номер	31...1	0
Доступ	U	R/W
Сброс	0	0
	-	PRIMASK

Таблица 8-11 – Описание бит регистра PRIMASK

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...1	-	Зарезервировано
0	PRIMASK	0 – не влияет; 1 – предотвращает активацию всех исключений с конфигурируемым приоритетом

8.4.11 Регистр маски сбоев Fault Mask

Регистр FAULTMASK предотвращает активацию всех исключений.

Таблица 8-12 – Регистр FAULTMASK

Номер	31...1	0
Доступ	U	R/W
Сброс	0	0
	-	FAULTMASK

Таблица 8-13 – Описание бит регистра FAULTMASK

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...1	-	Зарезервировано
0	FAULTMASK	0 – не влияет; 1 – предотвращает активацию всех исключений

Процессор устанавливает в ноль бит FAULTMASK при выходе из всех обработчиков за исключением NMI обработчика.

8.4.12 Регистр базового приоритета маски Base Priority Mask

Регистр BASEPRI задает минимальный приоритет процессу обработки исключений. Когда BASEPRI установлен в ненулевое значение, это приводит к предотвращению активации всех исключений с таким же или более предотвращает активацию всех исключений с таким же или более низким уровнем приоритета как значение в BASEPRI. Подробнее смотрите сводную таблицу регистров ядра для данных атрибутов. Значение бит представлено ниже.

Таблица 8-14 – Регистр BASEPRI

Номер	31...8	7...0
Доступ	U	R/W
Сброс	0	0
	-	BASEPRI

Таблица 8-15 – Описание бит регистра BASEPRI

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...8	-	Зарезервировано
7...0	BASEPRI	0 – не влияет; Ненулевое – задает базовый приоритет для процесса обработки исключений

Процессор не обрабатывает какие-либо исключения со значением приоритета, большим или равным BASEPRI.

Это поле подобно полю приоритета в регистре приоритетов прерываний. Процессор анализирует только биты [7:4] этого поля, биты [3:0] читаются как нули и игнорируются при записи. Для более полной информации см. Таблица 37-8. Помните, что большее значение в поле приоритета соответствует меньшему приоритету обработчика.

8.4.13 Регистр управления CONTROL

Регистр CONTROL задает текущий стек и уровень привилегий выполняемой процессором программы в режиме Thread. Смотрите описание регистров процессорного ядра для атрибутов. Описание бит приведено ниже.

Таблица 8-16 – Регистр CONTROL

Номер	31...2	1	0
Доступ	U	R/W	R/W
Сброс	0	0	0
	-	Active Stack Pointer	Thread Mode Privilege Level

Таблица 8-17 – Описание бит регистра CONTROL

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...2	-	Зарезервировано
1	Active Stack Pointer	0 – MSP текущий указатель стека; 1 – PSP текущий указатель стека
0	Thread Mode Privilege Level	0 – привилегирован; 1 – непривилегирован

Режим Handler всегда использует указатель стека MSP, таким образом процессор игнорирует прямые записи в бит Active Stack Pointer регистра CONTROL в режиме handler. Вход и выход в обработчик исключений обновляют регистр CONTROL.

При работе с операционными средами рекомендуется, чтобы потоки, запущенные в режиме Thread, использовали стек PSP, а ядро и обработчики исключений использовали стек MSP.

По определению, режим Thread использует MSP. Для переключения указателя стека в режиме Thread на PSP используется MSR инструкция для установки бита Active Stack Pointer в 1, смотрите описание инструкции MSR.

Когда изменяется указатель стека, программное обеспечение должно использовать инструкцию ISB немедленно за инструкцией MSR. Это позволяет быть уверенным, что инструкции, следующие за выполнением ISB, будут использовать новый указатель стека. Смотрите описание инструкции ISB.

8.5 Исключения и прерывания

Процессорное ядро RISC поддерживает прерывания и системные исключения. Процессор и контроллер прерываний NVIC устанавливают приоритеты и обрабатывают все исключения. Исключение изменяет нормальный поток выполнения программы. Процессор использует handler режим для обработки всех исключений кроме сброса. Смотрите вход в исключение и выход из исключения для большей информации.

Регистры NVIC управляют обработкой прерываний. Смотрите раздел 37 «Контроллер прерываний NVIC (RISC)» для более подробной информации.

Типы данных

Процессор поддерживает следующие типы данных:

- 32-бита words;
- 16-бит halfwords;
- 8-бит bytes.

Процессор поддерживает 64-битную инструкцию передачи.

Процессор манипулирует всеми данными в режиме little-endian. Доступ в память инструкций и Private Peripheral Bus (PPB) всегда в режиме little-endian.

9 Система команд RISC

В процессоре реализована версия системы команд Thumb. Поддерживаемые команды представлены в Таблица 9-1.

В таблице используются следующие обозначения:

- в угловых скобках $\langle \rangle$ записываются альтернативные формы представления операндов;
- в фигурных скобках $\{ \}$ указываются необязательные операнды;
- информация в столбце «операнды» может быть неполной;
- второй операнд Op2 может быть либо регистром, либо константой;
- большинство команд могут содержать суффикс кода условного выполнения.
- более подробная информация представлена в детальном описании команд.

Таблица 9-1 – Система команд процессорного ядра RISC

Мнемокод команды	Операнды	Краткое описание	Флаги	Прим.
ADC, ADCS	{Rd,} Rn, Op2	Сложение с переносом	N,Z,C,V	
ADD, ADDS	{Rd,} Rn, Op2	Сложение	N,Z,C,V	
ADD, ADDW	{Rd,} Rn, #imm12	Сложение	N,Z,C,V	
ADR	Rd, label	Загрузка адреса, заданного относительно счетчика команд	-	
AND, ANDS	{Rd,} Rn, Op2	Логическое И	N,Z,C	
ASR, ASRS	Rd, Rm, <Rs/#n>	Арифметический сдвиг вправо	N,Z,C	
B	label	Переход	-	
BFC	Rd, #lsb, #width	Сброс элемента битового поля	-	
BFI	Rd,Rn,#lsb,#width	Запись заданного значения битового поля	-	
BIC, BICS	{Rd,} Rn, Op 2	Сброс бит по маске	N,Z,C	
BKPT	#imm	Точка останова	-	
BL	label	Переход со связью	-	
BLX	Rm	Косвенный переход со связью	-	
BX	Rm	Косвенный переход	-	
CBNZ	Rn, label	Сравнение с нулем и переход по неравенству	-	
CBZ	Rn, label	Сравнение с нулем и переход по равенству	-	
CLREX	-	Сброс эксклюзивного доступа	-	
CLZ	Rd, Rm	Определить количество ведущих нулей	-	
CMN, CMNS	Rn, Op2	Сравнить с противоположным знаком	N,Z,C,V	
CMP, CMPS	Rn, Op2	Сравнить	N,Z,C,V	
CPSID	iflags	Изменить состояние процессора, запретить прерывания	-	
CPSIE	iflags	Изменить состояние процессора, разрешить прерывания	-	

Мнемокод команды	Операнды	Краткое описание	Флаги	Прим.
DMB	-	Барьер синхронизации доступа к памяти данных	-	
DSB	-	Барьер синхронизации доступа к памяти данных	-	
EOR, EORS	{Rd,} Rn, Op2	Исключающее ИЛИ	N,Z,C	
ISB	-	Барьер синхронизации доступа к инструкциям	-	
IT	-	Начало блока условно исполняемых инструкций	-	
LDM	Rn{!}, reglist	Загрузка множества регистров, инкремент после доступа	-	
LDMDB, LDMEA	Rn{!}, reglist	Загрузка множества регистров, декремент перед доступом	-	
LDMFD, LDMIA	Rn{!}, reglist	Загрузка множества регистров, инкремент после доступа	-	
LDR	Rt, [Rn, #offset]	Загрузка слова в регистр	-	
LDRB, LDRBT	Rt, [Rn, #offset]	Загрузка байта в регистр	-	
LDRD	Rt,Rt2,[Rn,#offset]	Загрузка двойного слова в пару регистров	-	
LDREX	Rt, [Rn, #offset]	Эксклюзивное чтение регистра	-	
LDREXB	Rt, [Rn]	Эксклюзивное чтение регистра, байт	-	
LDREXH	Rt, [Rn]	Эксклюзивное чтение регистра, полуслово	-	
LDRH, LDRHT	Rt, [Rn, #offset]	Загрузка полуслова в регистр	-	
LDRSB, LDRSBT	Rt, [Rn, #offset]	Загрузка в регистр байта со знаком	-	
LDRSH, LDRSHT	Rt, [Rn, #offset]	Загрузка в регистр полуслова со знаком	-	
LDRT	Rt, [Rn, #offset]	Загрузка в регистр слова	-	
LSL, LSLS	Rd, Rm, <Rs #n>	Логический сдвиг влево	N,Z,C	
LSR, LSRS	Rd, Rm, <Rs #n>	Логический сдвиг вправо	N,Z,C	
MLA	Rd, Rn, Rm, Ra	Умножение и сложение, 32-битный результат	-	
MLS	Rd, Rn, Rm, Ra	Умножение и вычитание, 32-битный результат	-	
MOV, MOVs	Rd, Op2	Загрузка	N,Z,C	
MOVT	Rd, #imm16	Загрузка в старшее полуслово	-	
MOVW, MOV	Rd, #imm16	Загрузка 16-битной константы	N,Z,C	
MRS	Rd, spec_reg	Считать специальный регистр в регистр общего назначения	-	

Мнемокод команды	Операнды	Краткое описание	Флаги	Прим.
MSR	spec_reg, Rm	Записать регистр общего назначения в специальный регистр	N,Z,C,V	
MUL, MULS	{Rd,} Rn, Rm	Умножение, 32-разрядный результат	N,Z	
MVN, MVNS	Rd, Op2	Загрузка инверсного значения	N,Z,C	
NOP	-	Нет операции	-	
ORN, ORNS	{Rd,} Rn, Op2	Логическое ИЛИ-НЕ	N,Z,C	
ORR, ORRS	{Rd,} Rn, Op2	Логическое ИЛИ	N,Z,C	
POP	reglist	Извлечь регистры из стека	-	
PUSH	reglist	Занести регистры в стек	-	
RBIT	Rd, Rn	Изменить на обратный порядок бит в слове	-	
REV	Rd, Rn	Изменить на обратный порядок байтов в слове	-	
REV16	Rd, Rn	Изменить на обратный порядок байтов в полусловах	-	
REVSH	Rd, Rn	Изменить на обратный порядок байт в младшем полуслове, произвести распространение знакового бита в старшее полуслово	-	
ROR, RORS	Rd, Rm, <Rs/#n>	Циклический сдвиг вправо	N,Z,C	
RRX, RRXS	Rd, Rm	циклический сдвиг вправо на один бит с учетом переноса	N,Z,C	
RSB, RSBS	{Rd,} Rn, Op2	Вычитание с противоположным порядком аргументов	N,Z,C,V	
SBC, SBCS	{Rd,} Rn, Op2	Вычитание с учетом переноса	N,Z,C,V	
SBFX	Rd,Rn,#lsb, #width	Чтение значения битового поля, интерпретируемого как число со знаком	-	
SDIV	{Rd,} Rn, Rm	Деление чисел со знаком	-	
SEV	-	Установить признак события	-	
SMLAL	RdLo, RdHi, Rn, Rm	Умножение чисел со знаком с накоплением, 64-битный результат	-	
SMULL	RdLo, RdHi, Rn, Rm	Умножение чисел со знаком, 64-битный результат	-	
SSAT	Rd,#n,Rm{,shift#s}	Преобразование 32-разрядного числа в n-разрядное со знаком, с насыщением	Q	
STM	Rn{!}, reglist	Сохранение множества регистров, инкремент после доступа	-	
STMDB, STMEA	Rn{!}, reglist	Сохранение множества регистров, декремент перед доступом	-	
STMFD, STMIA	Rn{!}, reglist	Сохранение множества регистров, инкремент после доступа	-	

Мнемокод команды	Операнды	Краткое описание	Флаги	Прим.
STR	Rt, [Rn, #offset]	Сохранение регистра	-	
STRB, STRBT	Rt, [Rn, #offset]	Сохранение регистра, байт	-	
STRD	Rt, Rt2, [Rn, #offset]	Сохранение пары регистров, двойное слово	-	
STREX	Rd, Rt, [Rn, #offset]	Эксклюзивная запись регистра	-	
STREXB	Rd, Rt, [Rn]	Эксклюзивная запись регистра, байт	-	
STREXH	Rd, Rt, [Rn]	Эксклюзивная запись регистра, полуслово	-	
STRH, STRHT	Rt, [Rn, #offset]	Сохранение регистра, полуслово	-	
STRT	Rt, [Rn, #offset]	Сохранение регистра, слово	-	
SUB, SUBS	{Rd,} Rn, Op2	Вычитание	N,Z,C,V	
SUB, SUBW	{Rd,} Rn, #imm12	Вычитание	N,Z,C,V	
SVC	#imm	Вызов супервизора	-	
SXTB	{Rd,}Rm{,ROR#n}	Преобразовать байт со знаком в слово	-	
SXTH	{Rd,}Rm{,ROR#n}	Преобразовать полуслово со знаком в слово	-	
TBB	[Rn, Rm]	Табличный переход по индексу, смещения - байты	-	
TBH	[Rn, Rm, LSL #1]	Табличный переход по индексу, смещения - полуслова	-	
TEQ	Rn, Op2	Проверка равенства	N,Z,C	
TST	Rn, Op2	Проверка значения бит по маске	N,Z,C	
UBFX	Rd, Rn, #lsb, #width	Чтение значения битового поля, интерпретируемого как число без знака	-	
UDIV	{Rd,} Rn, Rm	Деление числе без знака	-	
UMLAL	RdLo, RdHi, Rn, Rm	Умножение чисел без знака с накоплением, 64-битный результат	-	
UMULL	RdLo, RdHi, Rn, Rm	Умножение чисел без знака, 64-битный результат	-	
USAT	Rd,#n,Rm{,shift#s}	Преобразование 32-разрядного число в n-разрядное со без знака, с насыщением	Q	
UXTB	{Rd,}Rm{,ROR#n}	Преобразовать байт без знака в слово	-	
UXTH	{Rd,}Rm{,ROR#n}	Преобразовать полуслово без знака в слово	-	
WFE	-	Ожидать событие	-	
WFI	-	Ожидать прерывание	-	

9.1 Встроенные функции

Стандарт ANSI языка C не обеспечивает непосредственного доступа к некоторым инструкциям процессора RISC. В данном разделе описаны встроенные (intrinsic) функции, которые указывают компилятору на необходимость генерации соответствующих инструкций. В случае, если используемый компилятор не поддерживает ту или иную встроенную функцию, рекомендуется включить в текст программы ассемблерную вставку с необходимой инструкцией.

В CMSIS предусмотрены следующие встроенные функции, расширяющие возможности стандарта ANSI C.

Таблица 9-2 – Встроенные функции CMSIS, позволяющие генерировать некоторые инструкции процессора RISC

Мнемокод команды процессора	Описание встроенной функции
CPSIE I	void __enable_irq(void)
CPSID I	void __disable_irq(void)
CPSIE F	void __enable_fault_irq(void)
CPSID F	void __disable_fault_irq(void)
ISB	void __ISB(void)
DSB	void __DSB(void)
DMB	void __DMB(void)
REV	uint32_t __REV(uint32_t int value)
REV16	uint32_t __REV16(uint32_t int value)
REVSH	uint32_t __REVSH(uint32_t int value)
RBIT	uint32_t __RBIT(uint32_t int value)
SEV	void __SEV(void)
WFE	void __WFE(void)
WFI	void __WFI(void)

Кроме того, CMSIS также обеспечивает возможность чтения и записи специальных регистров процессора, доступных с помощью команд MRS и MSR.

Таблица 9-3 – Встроенные функции CMSIS для доступа к специальным регистрам процессора

Наименование специального регистра	Режим доступа	Описание встроенной функции
PRIMASK	Чтение	uint32_t __get_PRIMASK (void)
	Запись	void __set_PRIMASK (uint32_t value)
FAULTMASK	Чтение	uint32_t __get_FAULTMASK (void)
	Запись	void __set_FAULTMASK (uint32_t value)
BASEPRI	Чтение	uint32_t __get_BASEPRI (void)
	Запись	void __set_BASEPRI (uint32_t value)
CONTROL	Чтение	uint32_t __get_CONTROL (void)
	Запись	void __set_CONTROL (uint32_t value)
MSP	Чтение	uint32_t __get_MSP (void)
	Запись	void __set_MSP (uint32_t TopOfMainStack)
PSP	Чтение	uint32_t __get_PSP (void)
	Запись	void __set_PSP (uint32_t TopOfProcStack)

9.2 Описание инструкций

В разделе представлена подробная информация об инструкциях процессора:

- операнды;
- ограничения на использование счетчика команд PC и указателя стека SP;
- формат второго операнда;
- операции сдвига;
- выравнивание адресов;
- выражения с участием счетчика команд;
- условное исполнение;
- выбор размера кода инструкции.

9.2.1 Операнды

В качестве операнда инструкции может выступать регистр, константа, либо другой параметр, специфичный для конкретной команды. Процессор применяет инструкцию к операндам и, как правило, сохраняет результат в регистре-получателе. В случае, если формат команды предусматривает спецификацию регистра-получателя, он, как правило, указывается непосредственно перед операндами.

Операнды в некоторых инструкциях допускают гибкий формат представления, то есть могут быть как регистром, так и константой. Подробнее см. пункт 9.2.3 «Формат второго операнда».

9.2.2 Ограничения на использование PC и SP

Многие инструкции не позволяют использовать регистры счетчика команд (PC) и указателя стека (SP) в качестве регистра-получателя. Подробная информация содержится в описании конкретных инструкций.

Бит [0] адреса, загружаемого в PC с помощью одной из команд BX, BLX, LDM, LDR или POP должен быть равен 1, так как этот бит указывает на требуемый набор команд, а процессор RISC поддерживает только инструкции из набора Thumb.

9.2.3 Формат второго операнда

Большинство команд обработки данных поддерживает гибкий формат задания второго операнда. Далее в описании синтаксиса инструкций процессора такой операнд будет обозначаться как *Operand2*. При этом в качестве операнда может выступать:

- константа;
- регистр с необязательным параметром сдвига.

9.2.3.1 Константа

Данный тип второго операнда задается в формате:

#constant

где constant может быть:

– любой константой, которая может быть получена путем сдвига восьмиразрядного числа влево на любое количество разрядов в пределах 32-разрядного слова;

- любой константа в виде 0x00XY00XY;
- любой константа в виде 0xXY00XY00;
- любой константа в виде 0xXYXYXYXY.

Во всех вышеописанных случаях X и Y представляют шестнадцатеричные цифры.

Кроме того, в небольшом количестве инструкций constant может принимать более широкий диапазон значений. Подробности изложены в описании соответствующих инструкций.

При использовании константного операнда Operand2 в командах MOVs, MVNS, ANDs, ORRs, ORNs, EORs, BICs, TEQ и TST в случае, если константа больше 255 и может быть получена путем сдвига восьмиразрядного числа, значение бита [31] константы влияет на значение флага переноса. Для всех остальных значений Operand2 изменения флага переноса не происходит.

Замена инструкций

В случае, если пользователь указывает константу, не удовлетворяющую требованиям, описанным в подпункте 9.2.3.1 «Константа», ассемблер может сгенерировать код с использованием другой инструкции, обеспечивающей необходимую функциональность.

Например, команда CMP Rd, #0xFFFFFFFFE может быть преобразована в эквивалентную команду CMN Rd, #0x2.

9.2.3.2 Регистр с необязательным параметром сдвига

В данном случае операнд Operand2 задается в форме:

Rm {, shift}

где Rm – регистр, содержащий данные для второго операнда инструкции;

shift – необязательный параметр, определяющий сдвиг данных регистра Rm. Он может принимать одно из следующих значений:

- ASR #n – арифметический сдвиг вправо на n бит, $1 \leq n \leq 32$;
- LSL #n – логический сдвиг влево на n бит, $1 \leq n \leq 31$;
- LSR #n – логический сдвиг вправо на n бит, $1 \leq n \leq 32$;
- ROR #n – циклический сдвиг вправо на n бит, $1 \leq n \leq 31$;
- RRX – циклический сдвиг вправо на один бит, с учетом переноса.

Случай, когда сдвиг не указан, эквивалентен заданию сдвига LSL #0. При этом в качестве операнда используется непосредственно значение регистра Rm без каких-либо дополнительных преобразований.

При указании параметра сдвига в качестве операнда используется преобразованное соответствующим образом 32-разрядное значение регистра Rm, однако содержимое самого регистра Rm не меняется.

Использование операнда со сдвигом в некоторых инструкциях влияет на значение флага переноса. Более подробно действие операций сдвига и их влияние на флаг переноса рассмотрено в пункте 9.2.4 «Операции сдвига».

9.2.4 Операции сдвига

Операции сдвига переносят значение бит содержимого регистра влево или вправо на заданное количество позиций - длина сдвига. Сдвиг может выполняться:

- непосредственно с помощью инструкций ASR, LSR, LSL, ROR и RRX, при этом результат сдвига заносится в регистр-получатель;
- во время вычисления значения второго операнда Operand2 команд, при этом результат сдвига используется как один из операндов инструкции.

Допустимая длина сдвига зависит от типа сдвига и инструкции, в которой он был применен (см. стр. 95). В случае, если этот параметр равен 0, фактически сдвиг не производится. Операции сдвига регистра влияют на значение флага переноса, за исключением случая, когда длина сдвига равна 0. Различные варианты сдвига и их влияние на флаг переноса описаны в следующем подразделе (R_m – сдвигаемый регистр, n – длина сдвига).

9.2.4.1 ASR

Арифметический сдвиг вправо на n бит переносит крайние слева 32- n бит регистра R_m вправо на n позиций, то есть на место крайних справа 32- n . Бит [31] исходного значения регистра записывается в n крайних слева бит результата. См. Рисунок 9-1.

Операцию ASR # n можно использовать для деления значения регистра R_m на 2^n , с округлением результата в меньшую сторону (в направлении минус бесконечности).

При использовании инструкции ASRS, а также в случае, если сдвиг ASR # n используется при вычислении второго операнда команд MOVS, MVNS, ANDS, ORRS, ORNS, EORS, BICS, TEQ или TST, флаг переноса принимает значение последнего бита, вытесненного в результате операции сдвига, то есть бита $[n-1]$ регистра R_m .

В случае, если $n \geq 32$, все биты результата устанавливаются в значение бита [31] регистра R_m . Если при этом операция влияет на флаг переноса, то значение этого флага устанавливается равным значению бита [31] регистра R_m .

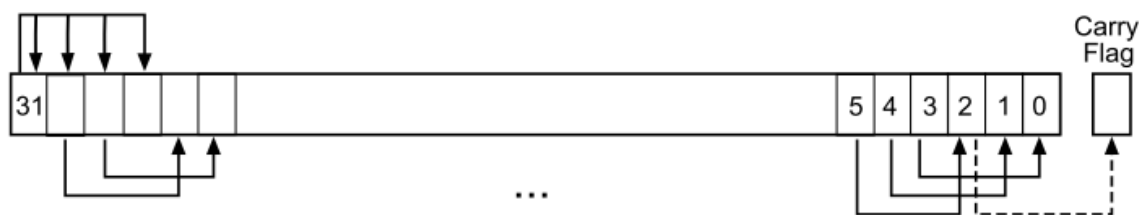


Рисунок 9-1 – Инструкция ASR # 3

9.2.4.2 LSR

Логический сдвиг вправо на n бит переносит крайние слева 32- n бит регистра R_m вправо на n позиций, то есть на место крайних справа 32- n . При этом в n крайних слева бит результата записывается 0. См. Рисунок 9-3.

Операцию $LSR \# n$ можно использовать для деления значения регистра R_m на 2^n в случае, если значение интерпретируется как целое число без знака.

При использовании инструкции $LSRS$, а также в случае, если сдвиг $LSR \# n$ используется при вычислении второго операнда команд $MOVS$, $MVNS$, $ANDS$, $ORRS$, $ORNS$, $EORS$, $BICS$, TEQ или TST , флаг переноса принимает значение последнего бита, вытесненного в результате операции сдвига, то есть бита $[n-1]$ регистра R_m .

В случае, если $n \geq 32$, все биты результата устанавливаются в 0. Если $n \geq 33$ и операция влияет на флаг переноса, то значение этого флага устанавливается равным 0.

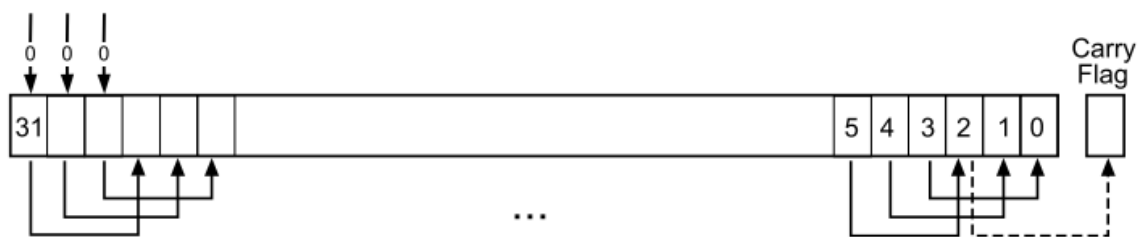


Рисунок 9-2 – Инструкция $LSR \# 3$

9.2.4.3 LSL

Логический сдвиг влево на n бит переносит крайние справа 32- n бит регистра R_m влево на n позиций, то есть на место крайних слева 32- n . При этом в n крайних слева бит результата записывается 0. См. Рисунок 9-3.

Операцию $LSL \# n$ можно использовать для умножения значения регистра R_m на 2^n в случае, если значение интерпретируется как целое число без знака, либо как целое число со знаком, записанное в дополнительном коде. Переполнение при выполнении умножения не диагностируется.

При использовании инструкции $LSLS$, а также в случае, если сдвиг $LSL \# n$ используется при вычислении второго операнда команд $MOVS$, $MVNS$, $ANDS$, $ORRS$, $ORNS$, $EORS$, $BICS$, TEQ или TST , флаг переноса принимает значение последнего бита, вытесненного в результате операции сдвига, то есть бита $[32-n]$ регистра R_m . Инструкция $LSL \# 0$ не влияет на значение флага переноса.

В случае, если $n \geq 32$, все биты результата устанавливаются в 0. Если $n \geq 33$ и операция влияет на флаг переноса, то значение этого флага устанавливается равным 0.

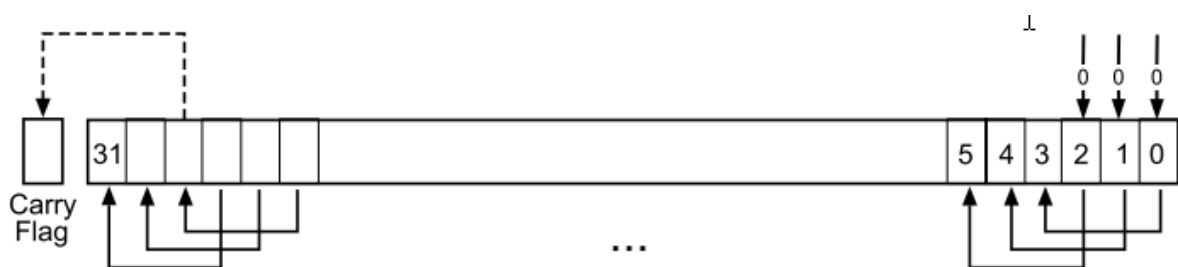


Рисунок 9-3 – Инструкция $LSL \# 3$

9.2.4.4 ROR

Циклический сдвиг вправо на n бит переносит крайние слева 32- n бит регистра R_m вправо на n позиций, то есть на место крайних справа 32- n . При этом n крайних справа разрядов регистра переносятся в n крайних слева разрядов результата. См. Рисунок 9-4.

При использовании инструкции RORS, а также в случае, если сдвиг ROR $\#n$ используется при вычислении второго операнда команд MOV_S, MVNS, AND_S, OR_S, ORNS, EOR_S, BIC_S, TEQ или TST, флаг переноса принимает значение последнего сдвинутого бита, то есть бита $[n-1]$ регистра R_m .

В случае, если $n = 32$, результат совпадает с исходным значением регистра. Если $n = 32$ и операция влияет на флаг переноса, то значение этого флага устанавливается равным биту $[31]$ регистра R_m .

Операция циклического сдвига ROR с параметром, большим 32, эквивалентна циклическому сдвигу с параметром $n-32$.

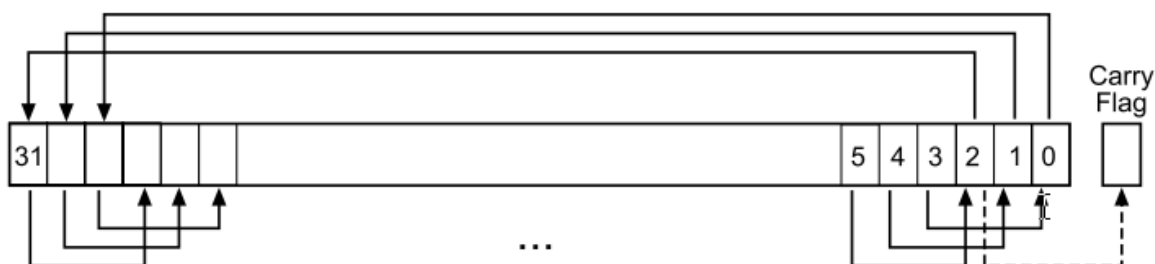


Рисунок 9-4 – Инструкция ROR $\#3$

9.2.4.5 RRX

Циклический сдвиг вправо на один бит с переносом переносит разряды регистра R_m вправо на одну позицию, при этом в позицию $[31]$ результата записывается значение флага переноса. См. Рисунок 9-5.

При использовании инструкции RRXS, а также в случае, если сдвиг RRX $\#n$ используется при вычислении второго операнда команд MOV_S, MVNS, AND_S, OR_S, ORNS, EOR_S, BIC_S, TEQ или TST, флаг переноса принимает значение бита $[0]$ регистра R_m .

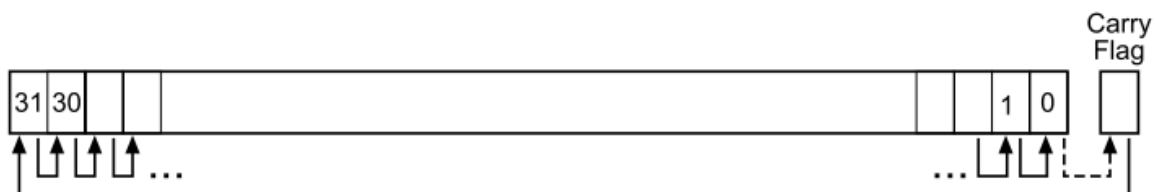


Рисунок 9-5 – Инструкция RRX

9.2.5 Выравнивание адресов

Под доступом по выровненным адресам понимаются операции, в которых чтение и запись слов, двойных слов и более длинных последовательностей слов осуществляется по адресам, выровненным по границе слова, а доступ к полусловам осуществляется по адресам, выровненным по границе полуслова. Чтение и запись байтов гарантированно являются выровненными.

Процессор поддерживает доступ по невыровненным адресам только для следующих инструкций:

- LDR, LDRT;
- LDRH, LDRHT;
- LDRSH, LDRSHT;
- STR, STRT;
- STRH, STRHT.

Все остальные инструкции при попытке доступа по невыровненному адресу генерируют исключение (usage fault). Более подробно данный вопрос рассмотрен в подразделе 0 «

Обработка отказов».

Невыровненный доступ к данным, как правило, осуществляется медленнее, чем выровненный. Кроме того, некоторые области адресного пространства могут не поддерживать доступ по невыровненному адресу. В связи с этим рекомендуется обеспечивать необходимое выравнивание данных. Для того, чтобы избежать случаев, в которых невыровненный доступ осуществляется непреднамеренно, следует установить в «1» бит UNALIGN_TRP регистра конфигурации и управления CCR, что приведет тогда к формированию процессором исключительной ситуации (см. пункт 38.1.7 «SCB->CCR»).

9.2.6 Адресация относительно счетчика команд PC

В системе команд RISC предусмотрена адресация команды или области данных в виде суммы значения счетчика команд PC плюс/минус численное смещение. Смещение вычисляется ассемблером автоматически, исходя из адреса метки и текущего адреса. В случае, если смещение слишком велико, диагностируется ошибка.

- для инструкций B, BL, CBNZ и CBZ текущий адрес определяется как адрес этой инструкции плюс 4 байта;
- для всех остальных инструкций текущий адрес определяется как адрес инструкции плюс 4 байта, при этом бит [1] результата должен быть установлен в 0 для обеспечения выравнивания адреса по границе слова.
- ассемблер может поддерживать расширенные варианты синтаксиса для адресации относительно PC, например, «метка плюс/минус число» или выражения типа [PC, #number].

9.2.7 Условное исполнение

Большая часть команд обработки данных способна изменять значения флагов в регистре состояния прикладной программы (APSR) в зависимости от результата выполнения (см. пункт 8.4.6 «Программный регистр состояния приложения APSR»).

Некоторые команды влияют на все флаги, некоторые только на часть. В случае, если инструкция не меняет значение данного флага, сохраняется его старое значение. Более подробно влияние на флаги рассмотрено в описании конкретных инструкций.

Возможность исполнения или неисполнения инструкции в зависимости от значения флагов, сформированных ранее, может быть достигнута либо за счет использования условных переходов, либо путем добавления суффикса условия исполнения к инструкции. список суффиксов, которые можно добавить к инструкции для того, чтобы сделать ее условной представлен в Таблица 9-4.

При наличии одного из указанных суффиксов процессор проверяет значение флагов на соответствие заданному условию. Если условие не выполняется, то инструкция:

- не исполняется;
- не записывает значение операции в регистр-получатель;
- не влияет на флаги;
- не генерирует исключений.

Условные инструкции, за исключением условных переходов, должны располагаться внутри блока условно исполняемых инструкций (далее по тексту – блок IT). Ассемблеры некоторых поставщиков могут самостоятельно вставлять инструкцию IT в случае, если программист разместит условную инструкцию за пределами блока IT.

Для сравнения регистра с нулем и условного перехода по результату рекомендуется использовать команды CBZ и CBNZ.

Ниже в разделе рассматриваются:

- флаги условий;
- суффиксы условного исполнения.

9.2.7.1 Флаги условий

Регистр состояния прикладной программы APSR содержит следующие флаги:

- N=1 в случае, если результат операции меньше нуля, 0 в противном случае.
- Z=1 в случае, если результат равен нулю, 0 в противном случае.
- C=1 в случае, если при выполнении операции возник перенос, 0 в противном случае.
- V=1 в случае, если при выполнении операции возникло переполнение, 0 в противном случае.

Более подробно регистр APSR рассмотрен в разделе 8.4.5 «Программный регистр состояния PSR».

Перенос возникает в следующих случаях:

- результат сложения оказался больше или равен 2^{32} ;
- результат вычитания больше или равен нулю;
- в результате работы внутренней логики процессора при операциях загрузки данных и логических операций.

Переполнение возникает в случае, если результат сложения, вычитания или сравнения больше или равен 2^{31} , либо меньше -2^{31} .

Большая часть инструкций меняют значение флагов только в случае, если у них указан суффикс S. Подробную информацию см. в описании конкретных команд.

9.2.7.2 Суффиксы условного исполнения

В мнемокодах команд, допускающих условное исполнение, предусмотрена возможность указания необязательного кода условия. В описании синтаксиса это обозначается как {cond}. Исполнению условной инструкции должна предшествовать инструкция IT.

Если код условия указан, инструкция выполняется только при удовлетворении соответствующему условию флагов регистра APSR. Используемые коды представлены далее (Таблица 9-4). Там же указаны соответствующие логические выражения для значений флагов.

Условные команды рекомендуется использовать для снижения количества ветвлений в программе.

Таблица 9-4 – Суффиксы условного исполнения

Суффикс	Флаги	Значение
EQ	$Z = 1$	Равенство
NE	$Z = 0$	Неравенство
CS или HS	$C = 1$	Больше или равно, беззнаковое сравнение
CC или LO	$C = 0$	Меньше, беззнаковое сравнение
MI	$N = 1$	Меньше нуля
PL	$N = 0$	Больше или равно нулю
VS	$V = 1$	Переполнение
VC	$V = 0$	Нет переполнения
HI	$C = 1 \text{ and } Z = 0$	Больше, беззнаковое сравнение
LS	$C = 0 \text{ or } Z = 1$	Меньше или равно, беззнаковое сравнение
GE	$N = V$	Больше или равно, знаковое сравнение
LT	$N \neq V$	Меньше, знаковое сравнение
GT	$Z = 0 \text{ and } N = V$	Больше, знаковое сравнение
LE	$Z = 1 \text{ and } N \neq V$	Меньше или равно, знаковое сравнение
AL	1	Безусловное исполнение

Пример. Вычисление абсолютного значения

В данном примере проиллюстрировано использование условных инструкций для вычисления абсолютного значения числа: $R0 = \text{ABS}(R1)$.

MOVS R0, R1 ; $R0 = R1$, установка флагов.

IT MI ; инструкция IT для условия отрицательного результата.

RSBMI R0, R1, #0 ; если результат был меньше нуля, присвоить $R0 = -R1$.

Пример. Сравнение и изменение значения регистра

В данном примере условные инструкции используются во фрагменте кода, изменяющего значение регистра R4 в случае, если число со знаком в регистр R0 больше числа в R1 и R2 больше R3.

CMP R0, R1 ; Сравнение R0 и R1, установка флагов.

ITT GT ; инструкция IT для двух условий «больше».

CMPGT R2, R3 ; если $R0 > R1$, сравнить R2 и R3, установить флаги.

MOVGT R4, R5 ; если условие «больше» все еще выполняется, присвоить $R4 = R5$.

9.2.8 Выбор размера кода инструкции

Многие команды процессора RISC могут быть представлены как 16-разрядными, так и 32-разрядными кодами инструкции. Во многих случаях выбор формата зависит от операндов и регистра-получателя результата.

Нередко существует возможность принудительно задать размер инструкции с помощью суффикса размера команды. Суффикс .W задает кодирование команды в 32-битном формате, суффикс .N- в 16-битном формате.

В случае, если суффикс размера указан, а ассемблер не в состоянии сгенерировать код команды соответствующего размера, диагностируется ошибка.

Для того, чтобы принудительно задать размер кода инструкции, необходимо поместить соответствующий суффикс непосредственно после мнемокода команды и кода условного выполнения, если он указан. В примере, приведенном ниже, представлены инструкции с заданным кодом размера.

BCS.W label ; формирует 32-битный код команды, даже для коротких переходов

ADDS.W R0, R0, R1 ; формирует 32-битный код команды, хотя данная операция может быть закодирована 16 битами.

9.3 Команды доступа к памяти

Обобщенные данные о командах доступа к памяти представлены ниже (Таблица 9-5).

Таблица 9-5 – Команды доступа к памяти

Мнемокод	Краткое описание	Прим.
ADR	Загрузка адреса, заданного относительно счетчика команд	
CLREX	Сброс эксклюзивного доступа	
LDM{mode}	Загрузка множества регистров	
LDR{type}	Загрузка регистра, непосредственно указанное смещение	
LDR{type}	Загрузка регистра, смещение в регистре	
LDR{type}T	Загрузка регистра с непривилегированным доступом	
LDR	Загрузка регистра по относительному адресу	
LDREX{type}	Эксклюзивное чтение регистра	
POP	Извлечение регистров из стека	
PUSH	Загрузка регистров в стек	
STM{mode}	Сохранение множества регистров	
STR{type}	Сохранение регистра, непосредственно указанное смещение	
STR{type}	Сохранение регистра, смещение в регистре	
STR{type}T	Сохранение регистра с непривилегированным доступом	
STREX{type}	Эксклюзивная запись регистра	

9.3.1 ADR

Загрузка адреса, заданного относительно счетчика команд.

Синтаксис

ADR{cond} Rd, label

где cond – необязательный код условия, см. пункт 9.2.7 «Условное исполнение».

Rd – регистр-получатель.

label – относительный адрес, см. пункт 9.2.6 «Адресация относительно счетчика команд PC».

Описание

Инструкция ADR вычисляет адрес доступа к памяти путем сложения текущего значения счетчика команд PC и непосредственно заданного смещения, после чего записывает результат в регистр-получатель.

Благодаря использованию относительной адресации, код команды не зависит от ее размещения в физической памяти.

При формировании с помощью команды ADR адреса перехода для команд BX или BLX программисту необходимо убедиться, что бит [0] формируемого адреса установлен в «1».

Значения смещения относительно PC должны находиться в пределах – 4095...+4095. Для того, чтобы использовать максимально широкий диапазон относительных адресов, а также чтобы иметь возможность генерировать адреса, не выровненные по границе слова, может потребоваться задать суффикс .W (см. пункт 9.2.8 «Выбор размера кода инструкции»).

Ограничения

В качестве регистра Rd нельзя использовать указатель стека SP и счетчик команд PC.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

ADR R1, TextMessage ; Загрузить адрес позиции, указанный меткой TextMessage, в регистр R1.

9.3.2 LDR и STR, непосредственно заданное смещение

Загрузка или сохранение регистра в режиме адресации со смещением, адресации с преиндексированием или адресации с постиндексированием.

Синтаксис

op{type}{cond} Rt, [Rn {, #offset}] ; адресация со смещением

op{type}{cond} Rt, [Rn, #offset]! ; преиндексирование

op{type}{cond} Rt, [Rn], #offset ; постиндексирование

opD{cond} Rt, Rt2, [Rn {, #offset}] ; адресация со смещением, двойное слово

opD{cond} Rt, Rt2, [Rn, #offset]! ; преиндексирование, двойное слово

opD{cond} Rt, Rt2, [Rn], #offset ; постиндексирование, двойное слово

где *ор* – один из кодов операций:

- LDR – загрузить регистр;
- STR – сохранить регистр.

type – один из суффиксов размера данных:

- B – байт без знака, при загрузке старшие байты устанавливаются в нуль;
- SB – байт со знаком, при загрузке происходит распространение знакового бита в

старшие байты (только LDR);

- H – беззнаковое полуслово, при загрузке старшие байты устанавливаются в нуль;
- SH – полуслово со знаком, при загрузке происходит распространение знакового

бита в старшие байты (только LDR);

- без суффикса – 32-разрядное слово.

cond – необязательный код условия, см. пункт 9.2.7 «Условное исполнение».

Rt – регистр, в который должна производиться загрузка или значение которого должно быть сохранено.

Rn – регистр, содержащий базовый адрес памяти.

Offset – смещение относительно базового адреса *Rn*. В случае, если смещение не указано, оно подразумевается равным нулю.

Rt2 – дополнительный регистр, предназначенный для двухсловных операций чтения или записи.

Описание

LDR – загружает один или два регистра значением из памяти.

STR – сохраняет значение одного или двух регистров в память.

Инструкции с непосредственно заданным смещением могут функционировать в одном из следующих режимов адресации:

Адресация со смещением	Значение смещения добавляется к или вычитается из содержимого регистра <i>Rn</i> . Результат используется в качестве адреса чтения или записи. Значение регистра <i>Rn</i> остается неизменным. Синтаксис задания данного режима: [<i>Rn</i> , # <i>offset</i>]
Адресация с преиндексированием	Значение смещения добавляется к или вычитается из содержимого регистра <i>Rn</i> . Результат используется в качестве адреса чтения или записи, а также записывается обратно в регистр <i>Rn</i> . Синтаксис задания данного режима: [<i>Rn</i> , # <i>offset</i>]!
Адресация с постиндексированием	Содержимое регистра <i>Rn</i> используется в качестве адреса чтения или записи. Значение смещения добавляется к или вычитается из содержимого регистра <i>Rn</i> , после чего записывается обратно в регистр <i>Rn</i> . Синтаксис задания данного режима: [<i>Rn</i>], # <i>offset</i>

Загружаемое или сохраняемое значение может быть байтом, полусловом, словом или двойным словом. Байты и полуслова могут интерпретироваться как числа со знаком или без знака (см. пункт 9.2.5 «Выравнивание адресов»).

Таблица 9-6 показывает диапазоны значений смещения для различных форм адресации.

Таблица 9-6 – Диапазон значений смещения

Тип инструкции	Смещение	Преиндексирование	Постиндексирование
Слово, полуслово, байт	от -255 до 4095	от -255 до 255	от -255 до 255
Двойное слово	Значения, кратные 4, в диапазоне от -1020 до 1020		

Ограничения

Для команд загрузки регистров:

- использовать в качестве Rt регистры PC и SP можно только в командах загрузки слова;
- при загрузке двойных слов регистры Rt и Rt2 не должны совпадать
- в режимах адресации с пре- и постиндексированием регистр Rn не должен совпадать с регистрами Rt или Rt2.

В случае, если в команде загрузки слова в качестве регистра Rt используется счетчик команд PC:

- бит [0] загружаемого значения должен быть равен 1;
- передача управления происходит по адресу, соответствующему значению бита [0] в 0;
- если инструкция является условной, то она должна быть последней инструкцией в блоке IT.

Для команд сохранения регистров:

- использовать в качестве Rt регистры SP можно только в командах записи слова;
- в качестве регистров Rt и Rn нельзя использовать счетчик команд PC;
- в режимах адресации с пре- и постиндексированием регистр Rn не должен совпадать с регистрами Rt или Rt2.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

LDR R8, [R10] ; Загрузка регистра R8 из ячейки по адресу, содержащемуся в R10.

LDRNE R2, [R5, #960]! ; Условная загрузка R2 из слова памяти, расположенного на 960 байт выше адреса в регистре R5, увеличение регистра R5 на 960.

STR R2, [R9, #const-struct] ; const-struct - выражение с постоянным значением, лежащим в диапазоне 0 – 4095.

STRH R3, [R4], #4 ; Записать содержимое R3, интерпретируемое как полуслово, по адресу, содержащемуся в R4, после чего увеличить R4 на 4.

LDRD R8, R9, [R3, #0x20] ; Загрузить R8 словом данных, расположенным на 32 байта выше адреса в R3, загрузить R9 словом данных, расположенным на 36 байта выше адреса в R3.

STRD R0, R1, [R8], #-16 ; Сохранить R0 по адресу, содержащемуся в R8, сохранить R1 по адресу, расположенному на 4 байта выше адреса в R8, уменьшить значение R8 на 16.

9.3.3 LDR и STR, смещение задано в регистре

Загрузка или сохранение регистра в режиме адресации со смещением, заданным в регистре.

Синтаксис

op{type}{cond} Rt, [Rn, Rm {, LSL #n}]

где op – один из кодов операций:

- LDR загрузить регистр.
- STR сохранить регистр.

type – один из суффиксов размера данных:

- B – байт без знака, при загрузке старшие байты устанавливаются в нуль.
- SB – байт со знаком, при загрузке происходит распространение знакового бита в старшие байты (только LDR).
- H – беззнаковое полуслово, при загрузке старшие байты устанавливаются в нуль.
- SH – полуслово со знаком, при загрузке происходит распространение знакового бита в старшие байты (только LDR).
- без суффикса – 32-разрядное слово.

cond – необязательный код условия, см. пункт 9.2.7 «Условное исполнение».

Rt – регистр, в который должна производиться загрузка или значение которого должно быть сохранено.

Rn – регистр, содержащий базовый адрес памяти.

Rm – регистр, содержащий смещение относительно базового адреса.

LSL #n – необязательный параметр сдвига, в диапазоне от 0 до 3.

Описание

LDR – загружает регистра значением из памяти.

STR – сохраняет значение регистра в памяти.

Адрес области памяти, в которую будет производиться обращение, вычисляется на основании значения базового адреса в регистре Rn и смещения. Смещение определяется значением регистра Rm и параметром сдвига влево значения этого регистра.

Считываемое или записываемое значение может иметь размер байта, полуслова или слова. При загрузке данных из памяти байты и полуслова могут интерпретироваться либо как числа со знаком, либо как беззнаковые (см. пункт 9.2.5 «Выравнивание адресов»).

Ограничения

Для данных команд:

- Rn не может быть счетчиком команд PC;
- Rm не может быть SP или PC;
- использовать в качестве Rt регистр SP можно только в командах чтения и записи слова;
- использовать в качестве Rt регистр PC можно только в командах чтения слова.

В случае, если в команде загрузки слова в качестве регистра Rt используется счетчик команд PC:

- бит [0] загружаемого значения должен быть равен 1, передача управления при этом осуществляется по выровненному по границе полуслова адресу;
- если инструкция является условной, то она должна быть последней инструкцией в блоке IT.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

STR R0, [R5, R1] ; Записать значение R0 по адресу, равному сумме R5 и R1.

LDRSB R0, [R5, R1, LSL #1] ; Читать байт по адресу, равному сумме R5 и R1, умноженному на два, распространить значение знакового бита на старшие значащие байты слова, загрузить результат в регистр R0.

STR R0, [R1, R2, LSL #2] ; Сохранить значение регистра R0 по адресу, равному $R1+4 \cdot R2$.

9.3.4 LDR and STR, непривилегированный доступ

Загрузка или сохранение регистра в режиме непривилегированного доступа.

Синтаксис

op{type}T{cond} Rt, [Rn {, #offset}]

где op – один из кодов операций:

- LDR загрузить регистр.
- STR сохранить регистр.

type – один из суффиксов размера данных:

- B – байт без знака, при загрузке старшие байты устанавливаются в нуль.
- SB – байт со знаком, при загрузке происходит распространение знакового бита в старшие байты (только LDR).
- H – беззнаковое полуслово, при загрузке старшие байты устанавливаются в нуль.
- SH – полуслово со знаком, при загрузке происходит распространение знакового бита в старшие байты (только LDR).
- без суффикса – 32-разрядное слово.

cond – необязательный код условия, см. пункт 9.2.7 «Условное исполнение».

Rt – регистр, в который должна производиться загрузка или значение которого должно быть сохранено.

Rn – регистр, содержащий базовый адрес памяти.

offset – смещение относительно базового адреса Rn в диапазоне от 0 до 255. В случае, если смещение не указано, оно подразумевается равным нулю.

Описание

Описываемые инструкции загрузки и сохранения данных функционируют также, как и инструкции доступа к памяти с непосредственно задаваемым смещением, см. пункт 9.3.2 «LDR и STR, непосредственно заданное смещение».

Отличие состоит в том, что рассматриваемые в данном разделе команды обеспечивают исключительно непривилегированный доступ, даже в случае, если они исполняются в привилегированном приложении.

При использовании в непривилегированном приложении какие-либо отличия от команд нормального доступа к памяти с непосредственно задаваемым смещением отсутствуют.

Ограничения

В данных инструкциях:

- Rn не может быть счетчиком команд PC;
- Rt не может быть SP или PC.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

STRBTEQ R4, [R7] ; Условная запись младшего значащего байта в регистр R4 по адресу, хранящемуся в R7, с непривилегированным доступом.

LDRHT R2, [R2, #8] ; Загрузка полуслова из памяти по адресу, равному сумме регистра R2 и 8 в регистр R2, с непривилегированным доступом.

9.3.5 LDR, адресация относительно счетчика команд PC

Загрузка регистра из памяти.

Синтаксис

LDR{type}{cond} Rt, label

LDRD{cond} Rt, Rt2, label ; Load two words

где type – один из суффиксов размера данных:

- B – байт без знака, при загрузке старшие байты устанавливаются в нуль.
- SB – байт со знаком, при загрузке происходит распространение знакового бита в старшие байты (только для LDR).
- H – беззнаковое полуслово, при загрузке старшие байты устанавливаются в нуль.
- SH – полуслово со знаком, при загрузке происходит распространение знакового бита в старшие байты (только для LDR).
- без суффикса – 32-разрядное слово.

cond – необязательный код условия, см. пункт 9.2.7 «Условное исполнение».

Rt – регистр, в который должна производиться загрузка.

Rt2 – второй регистр, в который должна производиться загрузка.

label – относительный адрес, см. пункт 9.2.6 «Адресация относительно счетчика команд PC».

Описание

LDR – загружает регистр значением из памяти с адресом относительно счетчика команд PC, заданным в виде метки.

Считываемое значение может иметь размер байта, полуслова или слова. При загрузке данных из памяти байты и полуслова могут интерпретироваться либо как числа со знаком, либо как беззнаковые (см. пункт 9.2.5 «Выравнивание адресов»).

Метка должна располагаться на ограниченном расстоянии от текущей инструкции. Таблица 9-7 показывает возможные значения смещений между меткой данных и текущим значением счетчика команд. Для использования больших смещений, возможно, придется указать суффикс .W размера команды (см. пункт 9.2.8 «Выбор размера кода инструкции»).

Таблица 9-7 – Диапазон значений смещения

Тип инструкции	Диапазон значений смещения
Слово, полуслово со знаком или без знака, байт со знаком или без знака	от -4095 до 4095
Двойное слово	от -1020 до 1020

Ограничения

В данной инструкции:

- использовать в качестве Rt регистры PC или SP можно только в командах чтения слова;
- нельзя использовать в качестве Rt2 регистры PC и SP;
- при загрузке двойных слов регистры Rt и Rt2 не должны совпадать.

В случае, если в команде загрузки слова в качестве регистра Rt используется счетчик команд PC:

- бит [0] загружаемого значения должен быть равен 1, передача управления при этом осуществляется по выровненному по границе полуслова адресу;
- если инструкция является условной, то она должна быть последней инструкцией в IT-блоке.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

LDR R0, LookUpTable ; Загрузить R0 словом данных по адресу с меткой LookUpTable.

LDRSB R7, localdata ; Загрузить байт данных по адресу с меткой localdata, распространить значение знакового бита в старшие байты слова данных, сохранить результат в R7.

LDM и STM

Загрузка или сохранение множества регистров.

Синтаксис

`op{addr_mode}{cond} Rn{!}, reglist`

где `op` – один из кодов операций:

- LDM загрузить множество регистров.
- STM сохранить множество регистров.
- `addr_mode` – один из режимов адресации:
- IA – с увеличением адреса после каждого доступа. Этот режим используется по умолчанию.
- DB – с уменьшением адреса перед каждым доступом.

`Cond` – необязательный код условия, см. пункт 9.2.7 «Условное исполнение».

`Rn` – регистр, содержащий базовый адрес памяти.

`!` – необязательный суффикс обратной записи значения базового регистра. В случае, если он присутствует в команде, последний адрес, по которому осуществлялся доступ, будет записан обратно в регистр `Rn`.

`reglist` – заключенный в фигурные скобки список из одного или нескольких регистров, которые должны быть записаны или считаны. В списке можно указывать диапазон номеров регистров. Начальный и конечный регистр диапазона разделены знаком «-». Элементы списка (отдельные регистры или диапазоны) разделяются запятыми.

Мнемокоды LDM и LDMFD являются синонимами LDMIA. Наименование LDMFD обусловлено использованием данной команды для организации извлечения данных из стека, растущего вниз, с указателем на последний загруженный элемент (Full Descending stack).

Мнемокод LDMEA является синонимом LDMDB, его наименование обусловлено использованием данной команды для организации извлечения данных из стека, растущего вверх, с указателем на последнюю свободную ячейку (Empty Ascending stack).

Мнемокоды STM и STMEA являются синонимами STMIA. Наименование STMEA обусловлено использованием данной команды для организации записи данных из стека, растущего вверх, с указателем на последнюю свободную ячейку.

Мнемокод STMFD является синонимом STMDB, его наименование обусловлено использованием данной команды для организации извлечения данных из стека, растущего вниз, с указателем на последний загруженный элемент.

Описание

Инструкции LDM загружают регистры, перечисленные в списке `reglist`, значениями слов данных из памяти с базовым адресом, указанным в регистре `Rn`.

Инструкции STM записывают слова данных, содержащиеся в регистрах, перечисленных в списке `reglist`, в память с базовым адресом, указанным в регистре `Rn`.

Команды LDM, LDMIA, LDMFD, STM, STMIA и STMEA для доступа используют адреса памяти в интервале от R_n до $R_n+4\cdot(n-1)$, где n – количество регистров в списке reglist. Доступ осуществляется в порядке увеличения номера регистра, при этом регистр с наименьшим номером соответствует наименьшему адресу памяти, а регистр с наибольшим номером – наибольшему адресу. В случае, если в команде указан суффикс «!», значение $R_n+4\cdot(n-1)$ записывается обратно в регистр R_n .

Команды LDMDB, LDMEA, STMDB и STMFD для доступа используют адреса памяти в интервале от R_n до $R_n-4\cdot(n-1)$, где n – количество регистров в списке reglist. Доступ осуществляется в порядке уменьшения номера регистра, при этом регистр с наибольшим номером соответствует наибольшему адресу памяти, а регистр с наименьшим номером – наименьшему адресу. В случае, если в команде указан суффикс «!», значение $R_n-4\cdot(n-1)$ записывается обратно в регистр R_n .

Инструкции PUSH и POP могут быть выражены через инструкции LDM и STM. Подробности см. в пункте 9.3.6 «PUSH и POP».

Ограничения

В описываемых в разделе командах:

- в качестве регистра R_n нельзя использовать счетчик команд PC;
- список регистров reglist не может содержать указатель стека SP;
- в любой инструкции STM в списке регистров reglist нельзя указывать PC;
- в любой инструкции LDM в reglist нельзя указывать одновременно PC и LR;
- список reglist не может содержать R_n в случае, если указан суффикс «!».

В случае, если инструкция LDM содержит в списке reglist счетчик команд PC:

- бит [0] загружаемого значения должен быть равен 1, передача управления при этом осуществляется по выровненному по границе полуслова адресу;
- если инструкция является условной, то она должна быть последней инструкцией в блоке IT.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

LDM R8,{R0,R2,R9} ; LDMIA – синоним LDM.
STMDB R1!,{R3-R6,R11,R12}.

Примеры неправильного использования

STM R5!,{R5,R4,R9} ; Сохраненное значение R5 является непредсказуемым;
LDM R2, { } ; Список должен содержать хотя бы один регистр.

9.3.6 PUSH и POP

Загружает или считывает регистры в стек или из стека, растущего вниз, с указателем на последний загруженный элемент (full-descending stack).

Синтаксис

PUSH{cond} reglist
POP{cond} reglist

где `cond` – необязательный код условия, см. пункт 9.2.7 «Условное исполнение».

`reglist` – заключенный в фигурные скобки список из одного или нескольких регистров, которые должны быть записаны или считаны. В списке можно указывать диапазон номеров регистров. Начальный и конечный регистр диапазона разделены знаком «-». Элементы списка (отдельные регистры или диапазоны) разделяются запятыми.

Команды `PUSH` и `POP` являются синонимами команд `STMDB` и `LDM (LDMIA)` в которых базовый адрес памяти содержится в регистре указателя стека `SP`, а режим записи обратной записи значения базового регистра включен.

Мнемокоды `PUSH` и `POP` являются предпочтительными вариантами записи.

Описание

`PUSH` – сохраняет регистры в стеке в порядке уменьшения номеров регистров, при этом регистр с большим номером сохраняется в память с большим значением адреса.

`POP` – восстанавливает значения регистров из стека в порядке увеличения номеров регистров, при этом регистр с меньшим номером считывается из памяти с меньшим значением адресом.

Подробности см. в подпункте «`LDM` и `STM`».

Ограничения

В данных инструкциях:

- список регистров `reglist` не должен содержать указатель стека `SP`;
- в инструкции `PUSH` список регистров не должен содержать счетчик команд `PC`;
- в инструкции `POP` список регистров не должен одновременно содержать регистры `PC` и `LR`.

В случае, если инструкция `POP` содержит в списке `reglist` счетчик команд `PC`:

- бит [0] загружаемого значения должен быть равен 1, передача управления при этом осуществляется по выровненному по границе полуслова адресу;
- если инструкция является условной, то она должна быть последней инструкцией в IT-блоке.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

`PUSH {R0,R4-R7}`

`PUSH {R2,LR}`

`POP {R0,R10,PC}`

9.3.7 LDREX и STREX

Эксклюзивное чтение и запись регистров.

Синтаксис

```
LDREX{cond} Rt, [Rn {, #offset}]
STREX{cond} Rd, Rt, [Rn {, #offset}]
LDREXB{cond} Rt, [Rn]
STREXB{cond} Rd, Rt, [Rn]
LDREXH{cond} Rt, [Rn]
STREXH{cond} Rd, Rt, [Rn]
```

где cond – необязательный код условия, см. пункт 9.2.7 «Условное исполнение».

Rd – регистр-приемник, содержащий признак успешного выполнения операции.

Rt – записываемый (считываемый) регистр.

Rn – регистр, содержащий базовый адрес памяти.

offset – необязательный параметр - смещение данных относительно базового адреса.

В случае, если параметр опущен, он предполагается равным нулю.

Описание

Команды LDREX, LDREXB и LDREXH позволяют загрузить соответственно слово, байт или полуслово данных из области памяти с заданным адресом.

Команды STREX, STREXB и STREXH осуществляют попытку записи соответственно слова, байта или полуслова данных в область памяти с заданным адресом.

Адрес, используемый в инструкции эксклюзивной записи должен совпадать с адресом последней выполненной команды эксклюзивного чтения. Кроме того, значение, сохраняемое с помощью инструкции эксклюзивной записи, должно иметь тот же размер данных, что и значение, считанное предшествующей инструкцией эксклюзивного чтения.

В случае, если инструкция эксклюзивной записи успешно выполнила операцию, она записывает в регистр-получатель значение 0. В противном случае она возвращает 1. Запись в регистр-получатель значения 0 гарантирует, что никакие другие процессы не смогут получить доступ к данной ячейке памяти в промежутке времени между выполнением команд эксклюзивного чтения и эксклюзивной записи.

В интересах обеспечения высокой производительности необходимо свести количество операций между командами эксклюзивного чтения и эксклюзивной записи к минимуму.

Результат выполнения операции эксклюзивной записи по адресу, отличному от использованного ранее в инструкции эксклюзивного чтения, непредсказуем.

Ограничения

В рассматриваемых командах:

- нельзя использовать счетчик команд PC;
- нельзя использовать указатель стека SP в качестве регистров Rd и Rt;
- в инструкции STREX регистр Rd должен не совпадать с регистрами Rt и Rn;
- значение смещения offset должно быть кратно 4 и лежать в диапазоне от 0 до

1020.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

MOV R1, #0x1 ; Записать в регистр R1 значение, соответствующее блокировке ресурса try:

```
LDREX R0, [LockAddr] ; Считать значение признака блокировки
CMP R0, #0 ; ресурс свободен?
ITT EQ ; блок IT для инструкций STREXEQ и CMPEQ
STREXEQ R0, R1, [LockAddr] ; пытаемся заблокировать ресурс
CMPEQ R0, #0 ; получилось?
BNE try ; нет – попробовать еще раз
.... ; да – мы заблокировали ресурс.
```

9.3.8 CLREX

Сброс эксклюзивного доступа.

Синтаксис

CLREX{cond}

где cond – необязательный код условия, см. пункт 9.2.7 «Условное исполнение».

Описание

Инструкция CLREX заставляет процессор при выполнении очередной команды STREX, STREXB или STREXH не выполнять операцию записи и вернуть 1 как признак отказа.

Эта возможность используется при обработке исключительных ситуаций, возникших в промежутке между командой эксклюзивного чтения и соответствующей ей команде эксклюзивной записи, когда целесообразно принудительно инициировать отказ записи.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

CLREX.

9.4 Инструкции обработки данных

Таблица 9-8 представляет инструкции обработки данных.

Таблица 9-8 – Команды обработки данных

Мнемокод	Краткое описание	Прим.
ADC	Сложение с учетом переноса	
ADD	Сложение	
ADDW	Сложение	
AND	Логическое И	
ASR	Арифметический сдвиг вправо	
BIC	Сброс бит по маске	
CLZ	Определить количество ведущих нулей	
CMN	Сравнить с противоположным знаком	
CMP	Сравнить	
EOR	Исключающее ИЛИ	
LSL	Логический сдвиг влево	
LSR	Логический сдвиг вправо	
MOV	Загрузка	
MOVT	Загрузка в старшее полуслово	
MOVW	Загрузка 16-битной константы	
MVN	Загрузка инверсного значения	
ORN	Логическое ИЛИ-НЕ	
ORR	Логическое ИЛИ	
RBIT	Обратить порядок бит	
REV	Изменить на обратный порядок байтов в слове	
REV16	Изменить на обратный порядок байтов в полусловах	
REVSH	Изменить на обратный порядок байт в младшем полуслове, произвести распространение знакового бита в старшее полуслово	
ROR	Циклический сдвиг вправо	
RRX	Циклический сдвиг вправо на один бит с учетом переноса	
RSB	Вычитание с противоположным порядком аргументов	
SBC	Вычитание с учетом переноса	
SUB	Вычитание	
SUBW	Вычитание	
TEQ	Проверка равенства	
TST	Проверка значения бит по маске	

9.4.1 ADD, ADC, SUB, SBC и RSB

Сложение, сложение с переносом, вычитание, вычитание с переносом, вычитание с противоположным порядком аргументов.

Синтаксис

op{S}{cond} {Rd,} Rn, Operand2

op{cond} {Rd,} Rn, #imm12 ; только для команд ADD и SUB.

где *ор* – один из кодов операции:

- ADD – сложение;
- ADC – сложение с учетом переноса;
- SUB – вычитание;
- SBC – вычитание с учетом переноса;
- RSB – вычитание с противоположным порядком аргументов;
- S – необязательный суффикс. Если он указан, результат выполнения операции приводит к установке соответствующих флагов, см. пункт 9.2.7 «Условное исполнение».

cond – необязательный суффикс условного исполнения, см. пункт 9.2.7 «Условное исполнение».

Rd – регистр-получатель результата. В случае, если регистр *Rd* не указан, результат записывается в *Rn*.

Rn – регистр, содержащий значение первого операнда.

Operand2 – второй операнд. См. пункт 9.2.3 «Формат второго операнда».

imm12 – любое число в диапазоне от 0 до 4095.

Описание

Команда ADD складывает значение *Operand2* или *imm12* со значением регистра *Rn*.

Команда ADC складывает вместе значения *Rn* и *Operand2*, а также флага переноса.

Команда SUB вычитает значение *Operand2* или *imm12* из значения регистра *Rn*.

Команда SBC вычитает значение *Operand2* из значения регистра *Rn*. Если флаг переноса не установлен, результат дополнительно уменьшается на единицу.

Команда RSB вычитает значение регистра *Rn* из значения *Operand2*. Этот вариант команды полезен, так как существует широкий выбор вариантов построения *Operand2*.

Инструкции ADC и SBC полезны при реализации вычислений с повышенной разрядностью, см. «Арифметика с повышенной разрядностью».

См. также описание команды ADR.

Команда ADDW эквивалентна команде ADD, однако использует 12-разрядный непосредственный операнд *imm12*. Команда SUBW эквивалентна команде SUB, однако использует 12-разрядный непосредственный операнд *imm12*.

Ограничения

Для рассматриваемых инструкций:

- в качестве *Operand2* нельзя использовать SP или PC;
- использовать SP в качестве регистра *Rd* допустимо только в командах ADD и SUB, со следующими дополнительными ограничениями:
 - в качестве *Rn* также должен использоваться SP;
 - сдвиг в *Operand2* должен быть не более 3 бит в режиме LSL;
- указатель стека SP может использоваться в качестве *Rn* только в командах ADD и SUB;
- счетчик команд PC может использоваться в качестве *Rd* только в команде:
 - ADD{cond} PC, PC, Rm

причем:

- не допускается использование суффикса S;
- в качестве Rm не допускается использовать PC и SP;
- если инструкция условная, то она должна быть последней в блоке IT.

– в качестве регистра Rn можно использовать счетчик команд PC только в инструкциях ADD и SUB (за исключением команды ADD{cond} PC, PC, Rm) с дополнительными ограничениями:

- не допускается использование суффикса S;
- второй операнд должен находиться в интервале от 0 до 4095;
- при использовании PC в операциях сложения или вычитания биты [1:0] счетчика команд округляются до 0b00 перед выполнением операции, обеспечивая выравнивание адреса по границе слова;

– при необходимости сформировать адрес инструкции, необходимо скорректировать значение смещения относительно PC. Рекомендуется использовать вместо этого инструкцию ADR, так как в этом случае ассемблер автоматически сгенерирует правильное смещение;

- в случае, если PC используется в качестве Rd в команде

ADD{cond} PC, PC, Rm

бит[0] значения, записываемого в PC, будет проигнорирован, передача управления будет осуществляться по адресу, соответствующему нулевому значению этого бита.

Флаги

В случае, если в команде указан суффикс S, процессор устанавливает флаги N, Z, C и V в соответствии с результатом выполнения операции.

Примеры

ADD R2, R1, R3

SUBS R8, R6, #240 ; установить флаги по результату операции вычитания.

RSB R4, R4, #1280 ; вычесть содержимое регистра R4 из 1280.

ADCHI R11, R0, R3 ; операция выполняется только в случае, если флаг C установлен, а флаг Z сброшен.

Арифметика с повышенной разрядностью

64-разрядное сложение

Следующий пример показывает, как осуществить сложение 64-разрядного целого числа, записанного в паре регистров R2 и R3, с другим 64-разрядным числом, записанным в паре регистров R0 и R1, после чего записывает результат в пару регистров R4 и R5.

ADDS R4, R0, R2 ; сложить младшие значащие слова

ADC R5, R1, R3 ; сложить старшие значащие слова с учетом переноса

96-разрядное вычитание

Данные с повышенной разрядностью не обязательно содержать в смежных регистрах. В примере, приведенном ниже, показан фрагмент кода, осуществляющий вычитание 96-разрядного целого числа, записанного в регистрах R9, R1 и R11, из другого числа, содержащегося в R6, R2 и R8. Результат записывается в регистрах R6, R9 и R2.

SUBS R6, R6, R9 ; вычитание младших значащих слов.

SBCS R9, R2, R1 ; вычитание средних значащих слов с переносом.

SBC R2, R8, R11 ; вычитание старших значащих слов с переносом.

9.4.2 AND, ORR, EOR, BIC и ORN

Логические операции: И, ИЛИ, Исключающее ИЛИ, сброс бит по маске, ИЛИ-НЕ.

Синтаксис

op{S}{cond}{Rd,} Rn, Operand2

где op – один из кодов операции:

- AND – логическое И.
- ORR – логическое ИЛИ.
- EOR – логическое Исключающее ИЛИ.
- BIC – сброс бит по маске.
- ORN – логическое ИЛИ-НЕ.
- S – необязательный суффикс. Если он указан, результат выполнения операции приводит к установке соответствующих флагов, см. пункт 9.2.7 «Условное исполнение».

cond – необязательный суффикс условного исполнения, см. пункт 9.2.7 «Условное исполнение».

Rd – регистр-получатель результата.

Rn – регистр, содержащий значение первого операнда.

Operand2 – второй операнд, см. пункт 9.2.3 «Формат второго операнда».

Описание

Инструкции AND, ORR и EOR осуществляют, соответственно, логические операции И, ИЛИ и исключающего ИЛИ между первым операндом, содержащимся в регистре Rn, и вторым операндом Operand2.

Инструкция BIC выполняет операцию логического И между первым операндом, содержащимся в регистре Rn, и инверсным значением второго операнда Operand2.

Инструкция ORN выполняет операцию логического ИЛИ между первым операндом Rn и инверсным значением второго операнда Operand2

Ограничения

Не допускается использованием указателя стека SP и счетчика команд PC.

Флаги

В случае, если в команде указан суффикс S, процессор:

- устанавливает флаги N и Z в соответствии с результатом выполнения операции;
- может изменить флаг C в ходе вычисления значения второго операнда, см.

пункт 9.2.3 «Формат второго операнда»;

- не влияет на значение флага V.

Примеры

AND R9, R2, #0xFF00

ORREQ R2, R0, R5

ANDS R9, R8, #0x19

EORS R7, R11, #0x18181818

BIC R0, R1, #0xab

ORN R7, R11, R14, ROR #4

ORNS R7, R11, R14, ASR #32.

9.4.3 ASR, LSL, LSR, ROR и RRX

Арифметический сдвиг вправо, логический сдвиг влево, логический сдвиг вправо, циклический сдвиг вправо и циклический сдвиг вправо с переносом.

Синтаксис

op{S}{cond} Rd, Rm, Rs

op{S}{cond} Rd, Rm, #n

RRX{S}{cond} Rd, Rm

где op – один из кодов операции:

- ASR – арифметический сдвиг вправо;
- LSL – логический сдвиг влево;
- LSR – логический сдвиг вправо;
- ROR – циклический сдвиг вправо.

– S – необязательный суффикс. Если он указан, результат выполнения операции приводит к установке соответствующих флагов, см. пункт 9.2.7 «Условное исполнение».

cond – необязательный суффикс условного исполнения, см. пункт 9.2.7 «Условное исполнение».

Rd – регистр-получатель результата.

Rm – регистр, значение которого должно быть подвергнуто сдвигу.

Rs – регистр, содержащий параметр сдвига. Процессор анализирует только младший значащий байт регистра, таким образом, параметр сдвига может принимать значения от 0 до 255.

n – параметр сдвига. Диапазон допустимых значений параметра зависит от инструкции:

- ASR – от 1 до 32;
- LSL – от 0 до 31;
- LSR – от 1 до 32;
- ROR – от 1 до 31.

Команду

LSL{S}{cond} Rd, Rm, #0

рекомендуется записывать в формате

MOV{S}{cond} Rd, Rm.

Описание

Команда ASR, LSL, LSR и ROR сдвигает биты регистра Rm влево или вправо на заданное количество позиций, определяемое константой n или содержимым регистра Rs. Команда RRX осуществляет сдвиг Rm вправо на одну позицию с учетом переноса.

Во всех указанных инструкциях результат записывается в регистр Rd, при этом содержание регистра Rm остается неизменным. Детальное описание операций сдвига представлено в пункте 9.2.4 «Операции сдвига».

Ограничения

Не допускается использованием указателя стека SP и счетчика команд PC.

Флаги

В случае, если в команде указан суффикс S, процессор:

- устанавливает флаги N и Z в соответствии с результатом выполнения операции;
- флаг C устанавливается в значение последнего сдвинутого бита, за исключением случая параметра сдвига, равного нулю. См. пункте 9.2.4 «Операции сдвига».

Примеры

ASR R7, R8, #9 ; Арифметический сдвиг вправо на 9 бит

LSLS R1, R2, #3 ; Логический сдвиг влево на 3 бита с установкой флагов

LSR R4, R5, #6 ; Логический сдвиг вправо на 6 бит

ROR R4, R5, R6 ; Циклический сдвиг вправо на количество бит, указанное
; в младшем байте регистра R6

RRX R4, R5 ; Циклический сдвиг вправо через бит переноса.

9.4.4 CLZ

Определить количество ведущих нулей.

Синтаксис

CLZ{cond} Rd, Rm

где cond – необязательный суффикс условного исполнения, см. пункт 9.2.7 «Условное исполнение».

Rd – регистр-получатель результата.

Rm – регистр операнда.

Описание

Инструкция CLZ выполняет подсчет количества ведущих нулей в двоичной записи значения, записанного в регистре Rm, и возвращает результат в регистр Rd. Результат, равный 32, возвращается в случае, если в регистре Rm нет установленных бит, а результат, равный 0 – в случае, если установлен бит [31].

Ограничения

Не допускается использование указателя стека SP и счетчика команд PC.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

CLZ R4,R9

CLZNE R2,R3.

9.4.5 CMP и CMN

Сравнение и сравнение с противоположным знаком.

Синтаксис

CMP{cond} Rn, Operand2

CMN{cond} Rn, Operand2

где cond – необязательный суффикс условного исполнения, см. пункт 9.2.7 «Условное исполнение».

R – регистр, содержащий первый операнд.

Operand2 – второй операнд. См. пункт 9.2.3 «Формат второго операнда».

Описание

Данные инструкции осуществляют сравнение значений регистра и второго операнда. По результатам сравнения устанавливаются соответствующие флаги, однако сам результат в регистр не записывается.

Команда CMP вычитает из регистра Rn значение второго операнда Operand2. Она аналогична инструкции SUBS, за исключением того, что не сохраняет результат вычитания.

Команда CMN складывает значения регистра Rn и второго операнда Operand2. Она аналогична инструкции ADDS, за исключением того, что не сохраняет результат вычитания.

Ограничения

В данных инструкциях:

- не допускается использованием PC;
- в качестве второго операнда Operand2 нельзя использовать SP.

Флаги

Процессор устанавливает флаги N, Z, C и V в соответствии с результатом сравнения.

Примеры

CMP R2, R9

CMN R0, #6400

CMPGT SP, R7, LSL #2

9.4.6 MOV и MVN

Загрузка в регистр прямого или инверсного значения второго операнда.

Синтаксис

MOV{S}{cond} Rd, Operand2

MOV{cond} Rd, #imm16

MVN{S}{cond} Rd, Operand2

где S – необязательный суффикс. Если он указан, результат выполнения операции приводит к установке соответствующих флагов, см. пункт 9.2.7 «Условное исполнение».

cond – необязательный суффикс условного исполнения, см. пункт 9.2.7 «Условное исполнение».

Rd – регистр-получатель результата.

Operand2 – второй операнд. См. пункт 9.2.3 «Формат второго операнда».

imm16 – любое значение в диапазоне от 0 до 65535.

Описание

Инструкция MOV копирует значение второго операнда Operand2 в регистр Rd. В случае, если Operand2 является регистром с параметром сдвига, отличным от LSL #0, рекомендуется использовать вместо команды MOV соответствующую команду сдвига:

- ASR{S}{cond} Rd, Rm, #n вместо MOV{S}{cond} Rd, Rm, ASR #n;
- LSL{S}{cond} Rd, Rm, #n вместо MOV{S}{cond} Rd, Rm, LSL #n при n != 0;
- LSR{S}{cond} Rd, Rm, #n вместо MOV{S}{cond} Rd, Rm, LSR #n;
- ROR{S}{cond} Rd, Rm, #n вместо MOV{S}{cond} Rd, Rm, ROR #n;
- RRX{S}{cond} Rd, Rm вместо MOV{S}{cond} Rd, Rm, RRX.

Кроме того, допускается использовать дополнительные формы построения второго операнда Operand2 в инструкции MOV как синонимы соответствующих операций сдвига:

- MOV{S}{cond} Rd, Rm, ASR Rs является синонимом ASR{S}{cond} Rd, Rm, Rs;
- MOV{S}{cond} Rd, Rm, LSL Rs является синонимом LSL{S}{cond} Rd, Rm, Rs
- MOV{S}{cond} Rd, Rm, LSR Rs является синонимом LSR{S}{cond} Rd, Rm, Rs
- MOV{S}{cond} Rd, Rm, ROR Rs является синонимом ROR{S}{cond} Rd, Rm, Rs.

См. также описание инструкций ASR, LSL, LSR, ROR и RRX.

Инструкция MVN считывает значение второго операнда Operand2, производит его побитную инверсию, после чего помещает результат в регистр Rd.

Инструкция MOVW функционирует также, как и инструкция MOV, однако в качестве второго операнда в ней можно использовать только непосредственно задаваемое значение imm16.

Ограничения

Регистры SP и PC допускается использовать только с инструкцией MOV, при следующих ограничениях:

- второй операнд должен быть регистром без указания параметра сдвига;
- суффикс S не должен быть указан.

В случае, если в качестве Rd используется счетчик команд PC:

- бит [0] значения, загружаемого в PC, игнорируется;
- передача управления осуществляется по адресу, соответствующему загруженному значению с битом [0], принудительно установленным в 0.

Несмотря на то, что существует возможность использовать инструкцию MOV в качестве команды ветвления, настоятельно рекомендуется использовать для этих целей исключительно инструкции BX и BLX, в интересах обеспечения переносимости программного обеспечения.

Флаги

В случае, если в команде указан суффикс S, процессор:

- устанавливает флаги N и Z в соответствии с результатом выполнения операции;
- может изменить флаг C в ходе вычисления значения второго операнда, см. пункт 9.2.3 «Формат второго операнда»;
- не влияет на значение флага V.

Примеры

MOVS R11, #0x000B ; Записать значение 0x000B в R11, флаги устанавливаются
 MOV R1, #0xFA05 ; Записать значение 0xFA05 в R1, флаги не устанавливаются
 MOVS R10, R12 ; Записать регистр R12 в R10, флаги устанавливаются
 MOV R3, #23 ; Записать значение 23 в R3
 MOV R8, SP ; Записать значение указателя стека в регистр R8
 MVNS R2, #0xF ; Записать значение 0xFFFFFFFF0 (инверсия значения 0x0F)

в регистр R2, установить флаги.

9.4.7 MOVT

Записать в старшее полуслово регистра.

Синтаксис

MOVT{cond} Rd, #imm16

где cond – необязательный суффикс условного исполнения, см. пункт 9.2.7 «Условное исполнение».

Rd – регистр-получатель результата.

imm16 – любое значение в диапазоне от 0 до 65535.

Описание

Инструкция MOVT записывает 16-разрядное непосредственное значение imm16 в старшее полуслово регистра-приемника Rd[31:16]. Младшее полуслово Rd[15:0] остается неизменным.

Комбинация команд MOV и MOVT позволяет загрузить в регистр любую 32-битную константу.

Ограничения

В качестве Rd нельзя использовать указатель стека SP и счетчик команд PC.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

MOVT R3, #0xF123 ; Загрузить 0xF123 в старшее полуслово R3, младшее полуслово и регистр состояния APSR остаются неизменными.

9.4.8 REV, REV16, REVSH и RBIT

Изменение порядка бит или байтов в слове.

Синтаксис

op{cond} Rd, Rn

где op – один из кодов операции:

- REV – изменить на обратный порядок байтов в слове;
- REV16 – изменить на обратный порядок байтов в полуслухах;
- REVSH – изменить на обратный порядок байт в младшем полуслухе, произвести распространение знакового бита в старшее полуслово.
- RBIT – изменить порядок бит в 32-разрядном слухе.

cond – необязательный суффикс условного исполнения, см. пункт 9.2.7 «Условное исполнение».

Rd – регистр-получатель результата.

Rn – регистр, содержащий операнд.

Описание

Инструкции предназначены для изменения формата представления (endianness) данных:

– REV – преобразует 32-разрядное число в формате big-endian в число в формате little-endian и наоборот.

– REV16 – преобразует 32-разрядное число в формате big-endian в число в формате little-endian и наоборот.

– REVSH – выполняет одно из следующих преобразований:

16-разрядное число со знаком в формате big-endian в 32-разрядное число со знаком в формате little-endian;

16-разрядное число со знаком в формате little-endian в 32-bit 32-разрядное число со знаком в формате big-endian.

– RBIT – изменяет на обратный порядок бит в 32-разрядном слухе.

Ограничения

Нельзя использовать указатель стека SP и счетчик команд PC.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

REV R3, R7 ; Обратить порядок следования байтов в R7, записать в R3

REV16 R0, R0 ; Обратить порядок байтов в каждом 16-битном полуслухе R0

REVSH R0, R5 ; Обратить полуслух со знаком

REVHS R3, R7 ; Обратить порядок при условии «больше или равно» (HS)

RBIT R7, R8 ; Обратить порядок бит в R8, записать результат в R7.

9.4.9 TST и TEQ

Проверить значение бит по маске, проверить равенство.

Синтаксис

TST{cond} Rn, Operand2

TEQ{cond} Rn, Operand2

где cond – необязательный суффикс условного исполнения, см. пункт 9.2.7 «Условное исполнение».

Rn – регистр, содержащий первый операнд.

Operand2 – второй операнд. См. пункт 9.2.3 «Формат второго операнда».

Описание

Данные инструкции позволяют проверить значение регистра с учетом значения второго операнда Operand2. По результату устанавливаются флаги, сам результат не сохраняется.

Команда TST выполняет побитную операцию логического И между значениями Rn и Operand2. Она совпадает с инструкцией ANDS, за исключением того, что не сохраняет результат. Для того, чтобы проверить, что заданный бит регистра Rn равен 0 или 1, рекомендуется использовать команду TST со вторым операндом Operand2 в виде константы, в которой соответствующий бит равен 1, а все остальные - равны 0.

Команда TEQ выполняет побитную операцию Иключающее ИЛИ между значениями Rn и Operand2. Она совпадает с инструкцией EORS, за исключением того, что не сохраняет результат. Команда TEQ позволяет проверить равенство двух величин, не меняя значения флагов V и C. Кроме того, эта инструкция полезна для проверки знака числа. После сравнения флаг N является результатом операции Иключающее ИЛИ знаковых разрядов двух операндов.

Ограничения

Нельзя использовать указатель стека SP и счетчик команд PC.

Флаги

В случае, если в команде указан суффикс S, процессор:

- устанавливает флаги N и Z в соответствии с результатом выполнения операции;
- может изменить флаг C в ходе вычисления значения второго операнда, см. пункт 9.2.3 «Формат второго операнда»;
- не влияет на значение флага V.

Примеры

TST R0, #0x3F8 ; Побитное И между R0 и числом 0x3F8, устанавливаются флаги, результат не сохраняется

TEQEQ R10, R9 ; Условное исполнение проверки равенства регистров R10 и R9, устанавливаются флаги, результат не сохраняется.

9.5 Инструкции умножения и деления

В следующей таблице представлена информация о командах умножения и деления.

Таблица 9-9 – Инструкции умножения и деления

Мнемокод	Краткое описание
MLA	Умножение и сложение, 32-битный результат
MLS	Умножение и вычитание, 32-битный результат
MUL	Умножение, 32-разрядный результат
SDIV	Деление чисел со знаком
SMLAL	Умножение чисел со знаком с накоплением (32 • 32 + 64), 64-битный результат
SMULL	Умножение чисел со знаком, 64-битный результат
UDIV	Деление чисел без знака
UMLAL	Умножение чисел без знака с накоплением (32 • 32 + 64), 64-битный результат
UMULL	Умножение чисел без знака, 64-битный результат

9.5.1 MUL, MLA и MLS

Умножение или умножение с накоплением (сложением, вычитанием) с использованием 32-разрядных операндов и выдающее 32-разрядный результат.

Синтаксис

MUL{S}{cond} {Rd,} Rn, Rm ; Умножение

MLA{cond} Rd, Rn, Rm, Ra ; Умножение и сложение

MLS{cond} Rd, Rn, Rm, Ra ; Умножение и вычитание

где S – необязательный суффикс. Если он указан, результат выполнения операции приводит к установке соответствующих флагов, см. пункт 9.2.7 «Условное исполнение».

cond – необязательный суффикс условного исполнения, см. пункт 9.2.7 «Условное исполнение».

Rd – регистр-получатель результата. Если регистр Rd не указан, то в качестве получателя используется регистр Rn.

Rn, Rm – регистры, содержащий значения первого и второго сомножителей.

Ra – регистр, содержащий значение, к которому должно быть прибавлено или вычтено произведение.

Описание

Команда MUL выполняет перемножение значений, содержащихся в регистрах Rn и Rm, после чего сохраняет 32 младших значащих бита произведения в Rd.

Команда MLA перемножает содержимое регистров Rn и Rm, прибавляет к произведению значение Ra, после чего сохраняет 32 младших значащих бита результата в Rd.

Команда **MLS** перемножает содержимое регистров R_n и R_m , вычитает произведение из регистра R_a , после чего сохраняет 32 младших значащих бита результата в R_d .

Результат выполнения операций не зависит от того, используются ли в качестве операндов числа со знаком или без знака.

Ограничения

Нельзя использовать указатель стека SP и счетчик команд PC .

В случае, если инструкция **MUL** используется с суффиксом установки флагов S :

- регистры R_d , R_n и R_m должны находиться в диапазоне от R_0 до R_7 ;
- регистр R_d должен совпадать с R_m ;
- не допускается использование суффикса условного исполнения $cond$.

Флаги

В случае, если в команде указан суффикс S , процессор:

- устанавливает флаги N и Z в соответствии с результатом выполнения операции;
- не влияет на значение флагов C и V .

Примеры

MUL R_{10}, R_2, R_5 ; $R_{10} = R_2 \cdot R_5$

MLA R_{10}, R_2, R_1, R_5 ; $R_{10} = (R_2 \cdot R_1) + R_5$

MULS R_0, R_2, R_2 ; $R_0 = R_2 \cdot R_2$, установить флаги

MULLT R_2, R_3, R_2 ; условное исполнение $R_2 = R_3 \cdot R_2$

MLS R_4, R_5, R_6, R_7 ; $R_4 = R_7 - (R_5 \cdot R_6)$.

9.5.2 UMULL, UMLAL, SMULL и SMLAL

Умножение чисел со знаком или без знака, с возможностью накопления, 32-битные операнды, 64-битный результат.

Синтаксис

$op\{cond\} R_{dLo}, R_{dHi}, R_n, R_m$

где op – один из кодов операции:

- **UMULL** – умножение чисел без знака.
- **UMLAL** – умножение чисел без знака с накоплением.
- **SMULL** – умножение чисел со знаком.
- **SMLAL** – умножение чисел со знаком с накоплением.

$cond$ – необязательный суффикс условного исполнения, см. пункт 9.2.7 «Условное исполнение».

R_{dLo} , R_{dHi} – регистры-получатели младшей и старшей частей результата, соответственно. Для инструкций **UMLAL** и **SMLAL** они также содержат накапливаемое значение.

R_n , R_m – регистры, содержащие значения первого и второго сомножителей.

Описание

Инструкция UMULL перемножает значения регистров Rn и Rm, интерпретируя их как целые числа без знака. Результат умножения размещается в паре регистров RdHi (старшие 32 бита) и RdLo (младшие 32 бита).

Инструкция UMLAL перемножает значения регистров Rn и Rm, интерпретируя их как целые числа без знака. Результат прибавляется к 64-разрядному целому числу без знака, записанному в паре регистров RdHi и RdLo, после чего сохраняется обратно в паре регистров RdHi и RdLo.

Инструкция SMULL перемножает значения регистров Rn и Rm, интерпретируя их как целые числа со знаком, записанные в дополнительном коде. Результат умножения размещается в паре регистров RdHi (старшие 32 бита) и RdLo (младшие 32 бита).

Инструкция SMLAL перемножает значения регистров Rn и Rm, интерпретируя их как целые числа со знаком, записанные в дополнительном коде. Результат прибавляется к 64-разрядному целому числу со знаком, записанному в паре регистров RdHi и RdLo, после чего сохраняется обратно в паре регистров RdHi и RdLo.

Ограничения

Нельзя использовать указатель стека SP и счетчик команд PC.

Пара регистров RdHi и RdLo должна состоять из разных регистров.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

UMULL R0, R4, R5, R6 ; Беззнаковое умножение $(R4, R0) = R5 \cdot R6$

SMLAL R4, R5, R3, R8 ; Операция со знаком $(R5, R4) = (R5, R4) + R3 \cdot R8$.

9.5.3 SDIV и UDIV

Деление чисел со знаком или без знака.

Синтаксис

SDIV{cond} {Rd,} Rn, Rm

UDIV{cond} {Rd,} Rn, Rm

где cond – необязательный суффикс условного исполнения, см. пункт 9.2.7 «Условное исполнение».

Rd – регистр-получатель результата. Если Rd не указан, результат сохраняется в Rn.

Rn – регистр, содержащий значение делимого.

Rm – регистр, содержащий значение делителя.

Описание

Команда SDIV осуществляет деление целого числа со знаком, содержащегося в регистре Rn, на целое число со знаком, содержащееся в регистре Rm.

Команда UDIV осуществляет деление целого числа без знака, содержащегося в регистре Rn, на целое число без знака, содержащееся в регистре Rm.

В случае, если число в R_n не делится нацело на число в R_m , результат округляется в сторону нуля.

Ограничения

Нельзя использовать указатель стека SP и счетчик команд PC .

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

$SDIV\ R0, R2, R4$; Деление чисел со знаком, $R0 = R2/R4$

$UDIV\ R8, R8, R1$; Деление чисел без знака, $R8 = R8/R1$.

9.6 Инструкции преобразования данных с насыщением

В разделе рассмотрены инструкции преобразования данных с насыщением $SSAT$ и $USAT$.

9.6.1 $SSAT$ и $USAT$

Преобразование 32-разрядного числа в n -разрядное со знаком или без знака с насыщением.

Синтаксис

$op\{cond\}\ Rd, \#n, Rm\ \{, shift\ \#s\}$

где op – один из кодов операции:

- $SSAT$ – преобразует число со знаком в число со знаком, лежащее в заданном диапазоне значений, с насыщением.
- $USAT$ – преобразует число со знаком в число без знака, лежащее в заданном диапазоне значений, с насыщением.

$cond$ – необязательный суффикс условного исполнения, см. пункт 9.2.7 «Условное исполнение».

Rd – регистр-получатель результата.

Rm – регистр, содержащий преобразуемое значение.

N – определяет разрядность данных после преобразования с насыщением:

- n находится в диапазоне от 1 до 32 для инструкции $SSAT$
- n находится в диапазоне от 0 до 31 для инструкции $USAT$.

$shift\ \#s$ – необязательный параметр сдвига, применяемый к регистру Rm перед преобразованием с насыщением. Он может принимать следующие значения:

- $ASR\ \#s$, где s находится в диапазоне от 1 до 31;
- $LSL\ \#s$, где s находится в диапазоне от 0 до 31.

Описание

Инструкции преобразуют 32-разрядное число в n-разрядное число со знаком или без знака. Преобразование осуществляется с насыщением.

Команда SSAT подвергает операнд заданной операции сдвига, после чего приводит его к диапазону значений $-2^{(n-1)} \leq x \leq 2^{(n-1)} - 1$ в соответствии со следующими правилами:

- если значение после сдвига меньше $-2^{(n-1)}$, сохраняется результат $-2^{(n-1)}$;
- если значение после сдвига больше $2^{(n-1)} - 1$, сохраняется результат $2^{(n-1)} - 1$;
- в противном случае, результат сохраняется без изменений.

Команда USAT подвергает операнд заданной операции сдвига, после чего приводит его к диапазону значений $0 \leq x \leq 2^n - 1$ в соответствии со следующими правилами:

- если значение после сдвига меньше 0, сохраняется результат 0;
- если значение после сдвига больше $2^n - 1$, сохраняется результат $2^n - 1$;
- в противном случае, результат сохраняется без изменений.

В случае если значение операнда после сдвига отличается от сохраненного результата, возникает ситуация, называемая насыщением, при этом процессор устанавливает в слове состояния приложения APSR флаг Q в 1. В случае если в ходе преобразования данных насыщения не возникло, флаг Q сохраняет свое прежнее значение.

Для того, чтобы сбросить признак насыщения Q в 0, необходимо выполнить команду MSR, см. описание этой команды.

Проверить состояние флага Q можно с помощью команды MRS.

Ограничения

Нельзя использовать указатель стека SP и счетчик команд PC.

Флаги

Данная инструкция не влияет на состояние флагов, за исключением флага Q. Флаг Q устанавливается в 1 в случае, если при преобразовании данных произошло насыщение.

Примеры

SSAT R7, #16, R7, LSL #4 ; Логический сдвиг R7 влево на 4 бита, далее приведение его к 16-разрядному числу со знаком с насыщением, сохранить результат в R7.

USATNE R0, #7, R5 ; Условная операция: преобразовать с насыщением значение R5 к семиразрядному числу без знака, сохранить результат в R0.

9.7 Команды работы с битовыми полями

Таблица 9-10 показывает инструкции, позволяющие манипулировать последовательностями смежных бит данных в регистрах или битовых полях.

Таблица 9-10 – Инструкции упаковки и распаковки данных

Мнемокод команд	Краткое описание
BFC	Запись нуля в битовое поле
BFI	Запись заданного значения битового поля
SBFX	Чтение значения битового поля, интерпретируемого как число со знаком
SXTB	Преобразовать байт со знаком в слово
SXTH	Преобразовать полуслово со знаком в слово
UBFX	Чтение значения битового поля, интерпретируемого как число без знака
UXTB	Преобразовать байт без знака в слово
UXTH	Преобразовать полуслово без знака в слово

9.7.1 BFC и BFI

Сброс в ноль и запись заданного значения битового поля.

Синтаксис

BFC{cond} Rd, #lsb, #width

BFI{cond} Rd, Rn, #lsb, #width

где cond – необязательный суффикс условного исполнения, см. пункт 9.2.7 «Условное исполнение».

Rd – регистр-получатель результата.

Rm – регистр-источник данных.

Lsb – позиция младшего значащего разряда битового поля. Значение lsb должно находиться в интервале от 0 до 31.

Width – ширина битового поля, значение которой должно находиться в интервале от 1 до 32-lsb.

Описание

Инструкция BFC очищает битовое поле, размещенное в регистре Rd, имеющее длину width бит и расположенное, начиная с бита с номером lsb. Остальные биты регистра Rd сохраняются без изменений.

Инструкция BFI копирует битовое поле шириной в width бит, расположенное в регистре Rn, начиная с позиции 0, в битовое поле шириной в width бит, расположенное в регистре Rd, начиная с позиции lsb. Остальные биты регистра Rd сохраняются без изменений.

Ограничения

Нельзя использовать указатель стека SP и счетчик команд PC.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

BFC R4, #8, #12 ; Очистить 12-битовое поле, расположенное с 8-го по 19-й бит R4.

BFI R9, R2, #8, #12 ; Записать в 12-битовое поле, расположенное с 8-го по 19-й бит R9 значение из 12-битового поля, расположенного с 0-го по 11-й бит регистра R2.

9.7.2 SBFX и UBFX

Чтение значения битового поля, интерпретируемого как число со знаком или без знака.

Синтаксис

SBFX{cond} Rd, Rn, #lsb, #width

UBFX{cond} Rd, Rn, #lsb, #width

где cond – необязательный суффикс условного исполнения, см. пункт 9.2.7 «Условное исполнение».

Rd – регистр-получатель результата.

Rn – регистр-источник данных.

Lsb – позиция младшего значащего разряда битового поля. Значение lsb должно находиться в интервале от 0 до 31.

Width – ширина битового поля, значение которой должно находиться в интервале от 1 до 32-lsb.

Описание

Инструкция SBFX считывает значение битового поля из регистра-источника, производит распространение знакового бита в старшие биты 32-разрядного слова, сохраняет результат в регистр-получатель.

Инструкция UBFX считывает значение битового поля из регистра-источника, заполняет нулями старшие биты 32-разрядного слова, сохраняет результат в регистр-получатель.

Ограничения

Нельзя использовать указатель стека SP и счетчик команд PC.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

SBFX R0, R1, #20, #4 ; Извлечь 4 бита (с 20 по 23) из R1, интерпретируя их как число со знаком, записать в R0

UBFX R8, R11, #9, #10 ; Извлечь 10 бит (с 9 по 18) из R11, интерпретируя их как число без знака, записать в R8.

9.7.3 SXT и UXT

Преобразование байта или полуслова в слово с распространением знакового бита или нулей в старшие значащие разряды.

Синтаксис

`SXTextend{cond} {Rd}, Rm {, ROR #n}`

`UXTend{cond} {Rd}, Rm {, ROR #n}`

где Суффикс *extend* может принимать одно из следующих значений:

- В – преобразование 8-битного числа в 32-битное;
- Н – преобразование 16-битного числа в 32-битное.

cond – необязательный суффикс условного исполнения, см. пункт 9.2.7 «Условное исполнение».

Rd – регистр-получатель результата.

Rm – регистр-источник данных.

ROR #n – параметр сдвига, который может принимать одно из значений:

ROR #8 – значение в *Rm* циклически сдвигается вправо на 8 бит;

ROR #16 – значение в *Rm* циклически сдвигается вправо на 16 бит;

ROR #24 – значение в *Rm* циклически сдвигается вправо на 24 бит;

если параметр не указан, сдвиг не производится.

Описание

Команда **SXTB** осуществляет циклический сдвиг содержимого регистра *Rm* вправо на заданное число бит, извлекает из результата младшие восемь бит [7:0], преобразует их в 32-разрядное число со знаком путем копирования знакового разряда [7] в биты [31:8], сохраняет результат в регистре *Rd*.

Команда **UXTB** осуществляет циклический сдвиг содержимого регистра *Rm* вправо на заданное число бит, извлекает из результата младшие восемь бит [7:0], преобразует их в 32-разрядное число без знака путем копирования нуля в биты [31:8], сохраняет результат в регистре *Rd*.

Команда **SXTH** осуществляет циклический сдвиг содержимого регистра *Rm* вправо на заданное число бит, извлекает из результата младшие восемь бит [15:0], преобразует их в 32-разрядное число со знаком путем копирования знакового разряда [15] в биты [31:16], сохраняет результат в регистре *Rd*.

Команда **UXTH** осуществляет циклический сдвиг содержимого регистра *Rm* вправо на заданное число бит, извлекает из результата младшие восемь бит [15:0], преобразует их в 32-разрядное число без знака путем копирования нуля в биты [31:16], сохраняет результат в регистре *Rd*.

Ограничения

Нельзя использовать указатель стека *SP* и счетчик команд *PC*.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

SXTH R4, R6, ROR #16 ; сдвинуть R6 вправо на 16 бит, извлечь из результата младшее полуслово, преобразовать в 32-разрядное число с распространением знака, записать в R4.

UXTB R3, R10 ; извлечь младший байт из R10, преобразовать в 32- разрядное число, старшие байты заполнить нулями, записать результат в R3.

9.8 Инструкции передачи управления

В таблице ниже представлен список инструкций передачи управления.

Таблица 9-11 – Инструкции передачи управления

Мнемокод команды	Краткое описание
B	Переход
BL	Переход со связью
BLX	Косвенный переход со связью
BX	Косвенный переход
CBNZ	Сравнение с нулем и переход по неравенству
CBZ	Сравнение с нулем и переход по равенству
IT	Начало блока условно исполняемых инструкций
TBB	Табличный переход по индексу, смещения – байты
TBN	Табличный переход по индексу, смещения – полусллова

9.8.1 B, BL, BX и BLX

Команды ветвления.

Синтаксис

B{cond} label

BL{cond} label

BX{cond} Rm

BLX{cond} Rm

где:

B – переход по непосредственно заданному адресу;

BL – переход со связью по непосредственно заданному адресу;

BX – косвенный переход по адресу, заданному значением регистра;

BLX – косвенный переход со связью.

Cond– необязательный код условия, см. пункт 9.2.7 «Условное исполнение».

Label – относительный адрес, см. 9.3.5 «LDR, адресация относительно счетчика команд PC».

Rm – регистр, содержащий адрес, на который необходимо передать управления. Бит [0] этого регистра должен быть установлен в 1, однако передача управления будет выполнена по адресу, соответствующему нулевому значению бита [0].

Описание

Все рассматриваемые в данном разделе инструкции осуществляют передачу управления на адрес, заданный меткой, либо содержащийся в регистре Rm. Кроме того:

- команды BL и BLX записывают адрес следующей инструкции в регистр связи LR (R14);
- команды BX и BLX формируют отказ (usage fault) в случае, если bit[0] регистра Rm равен 0.

Инструкция вида B cond label - это единственный тип команды, который может находиться за пределами ИТ-блока. Все остальные условно исполняемые инструкции передачи управления должны располагаться внутри ИТ-блока, а за пределами этого блока должны использоваться только в безусловной форме. Подробности см. в пункте 9.8.1 «ИТ».

Диапазон адресуемых переходов для различных команд ветвления представлен ниже (Таблица 9-12). Для достижения максимального диапазона может потребоваться указать суффикс .W размера инструкции. Подробности см. в пункте 9.2.8 «Выбор размера кода инструкции».

Таблица 9-12 – Диапазон адресуемых переходов для команд ветвления

Инструкция	Диапазон адресации
B label	от -16 Мбайт до +16 Мбайт относительно текущей позиции
B cond label (вне ИТ-блока)	от -1 Мбайт до +1 Мбайт относительно текущей позиции
B cond label (внутри ИТ-блока)	от -16 Мбайт до +16 Мбайт относительно текущей позиции
BL{cond} label	от -16 Мбайт до +16 Мбайт относительно текущей позиции
BX{cond} Rm	любое значение, записанное в регистре
BLX{cond} Rm	любое значение, записанное в регистре

Ограничения

- в команде BLX не допускается использование регистра PC;
- в командах BX и BLX, бит [0] регистра Rm должен быть установлен в 1, при этом передача управления будет, тем не менее, осуществлена по адресу, соответствующему нулевому значению бита [0];
- внутри ИТ-блока любая из инструкций ветвления должна располагаться последней.
- B cond – единственная условно исполняемая команда, которую допустимо использовать за пределами ИТ-блока. Тем не менее, внутри ИТ-блока она обеспечивает более широкий диапазон адресуемых переходов.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

- B loopA ; передача управления на метку loopA
- BLE ng ; условная передача управления на метку ng
- B.W target ; переход на метку target, расположенную в пределах +/- 16Мбайт

BEQ target ; условный переход на метку target
 BEQ.W target ; условный переход на метку target в пределах ± 1 Мбайт
 BL funC ; переход со связью (вызов функции) funC, адрес возврата будет записан в регистре LR
 BX LR ; возврат из функции
 BXNE R0 ; условный переход по адресу, записанному в R0
 BLX R0 ; переход со связью (вызов функции) по адресу, записанному в R0.

9.8.2 CBZ и CBNZ

Сравнение и условная передача управления, по равенству или неравенству нулю.

Синтаксис

CBZ Rn, label
 CBNZ Rn, label

где Rn – регистр, содержащий операнд.

label – метка, на которую должен быть осуществлен переход.

Описание

Инструкции CBZ и CBNZ позволяют осуществить проверку на равенство нулю с условным переходом, при этом не влияя на значения флагов и снижая общее количество инструкций.

Команда CBZ Rn, label не влияет на флаги, а в остальном эквивалентна следующей последовательности инструкций:

CMP Rn, #0
 BEQ label

Команда CBNZ Rn, label не влияет на флаги, а в остальном эквивалентна следующей последовательности инструкций:

CMP Rn, #0
 BNE label

Ограничения

- в качестве Rn допустимо использовать регистры с R0 по R7;
- адрес перехода должен быть расположен после инструкции на расстоянии от 4 до 130 байт;
- данные команды нельзя использовать внутри IT-блока.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

CBZ R5, target ; Условный переход вперед при R5 = 0
 CBNZ R0, target ; Условный переход вперед при R0 != 0.

9.8.3 IT

Начало блока условно исполняемых инструкций.

Синтаксис

$IT\{x\{y\{z\}\}\} \text{ cond}$

где x определяет выбор условия для второй инструкции в блоке IT;
 y определяет выбор условия для третьей инструкции в блоке IT;
 z определяет выбор условия для четвертой инструкции в блоке IT;
 cond определяет условие для первой инструкции в блоке IT.

Суффиксы выбора условия для второй, третьей и четвертой инструкций IT-блока могут принимать одно из следующих значений:

- T – Then. Инструкция выполняется, если условие cond истинно;
- E – Else. Инструкция выполняется, если условие cond ложно.

Существует возможность использовать в блоке IT на месте cond условие AL (всегда истинное). В этом случае все инструкции в блоке IT должны быть безусловными, а суффиксы выбора условия x , y и z должны быть равны T, либо опущены.

Описание

Команда IT делает условными до четырех следующих за ней инструкций. Условия могут либо совпадать, либо быть логически противоположными. Условные инструкции, следующие за командой IT, формируют блок IT.

Мнемокоды команд внутри блока IT, в том числе и команд ветвления, должны включать в себя суффикс условного исполнения $\{\text{cond}\}$.

Ассемблеры некоторых производителей способны автоматически генерировать необходимые инструкции IT, предшествующие условно исполняемым командам, избавляя разработчика от необходимости делать эту работу вручную. Подробности следует уточнить в документации на Ваш ассемблер.

Команда VKPT внутри блока IT всегда выполняется, вне зависимости от истинности или ложности условия.

Обработка исключений внутри блока IT, а также непосредственно после инструкции IT допускается. При этом осуществляется переход на соответствующий обработчик с предварительным сохранением регистра PSR в стеке и необходимой для корректного возврата информации в регистре LR. Возврат из обработчика осуществляется стандартным образом, при этом корректное выполнение блока IT продолжается с прерванной позиции. Это единственный допустимый способ передачи управления внутрь блока IT с помощью команд, модифицирующих счетчик команд PC.

Ограничения

Следующие инструкции нельзя использовать внутри IT-блока:
IT, CBZ и CBNZ, CPSI D и CPSI E.

Кроме того, существуют следующие ограничения при использовании IT-блоков:

- ветвление, а также любая другая команда, модифицирующая счетчик команд PC, должны передавать управление либо за пределы блока IT, либо на последнюю инструкцию блока IT.

- инструкции, модифицирующие счетчик команд:

ADD PC, PC, Rm;

MOV PC, Rm;

B, BL, BX, BLX;

любая инструкция LDM, LDR или POP, приводящая к записи значения в PC;

TBB and TBH.

- не допускается передача управления на инструкцию внутри блока IT, за исключением случая возврата из обработчика исключения;

- все условные инструкции, за исключением B cond, должны находиться внутри блока IT. Команда B cond может быть расположена как внутри, так и вне блока IT, однако внутри IT-блока она обеспечивает более широкий диапазон адресуемых переходов;

- каждая инструкция внутри блока IT должна быть снабжена суффиксом условного исполнения с кодом, либо совпадающим, либо противоположным коду условия блока IT.

Ассемблер конкретного производителя может накладывать дополнительные ограничения, например, возможен запрет на использование директив внутри блока IT.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

ITTE NE ; Следующие три инструкции – условные

ANDNE R0, R0, R1 ; ANDNE не изменяет состояние флагов

ADDNE R2, R2, #1 ; ADDNE изменяет состояние флагов

MOVEQ R2, R3 ; условное копирование

CMP R0, #9 ; преобразование R0 (от 0 до 15) в код ASCII шестнадцатеричного числа ('0'-'9', 'A'-'F')

ITE GT ; следующие две инструкции - условные

ADDGT R1, R0, #55 ; [R0 > 9] преобразуем число 0xA → в код 'A'

ADDLE R1, R0, #48 ; [R0 <= 9] преобразуем число 0x0 → в код '0'

IT GT ; блок IT с одной единственной условной инструкцией

ADDGT R1, R1, #1 ; условное увеличение R1 на единицу

ITTEE EQ ; следующие четыре инструкции - условные

MOVEQ R0, R1 ; условное копирование

ADDEQ R2, R2, #10 ; условное сложение

ANDNE R3, R3, #1 ; условное логическое И

BNE.W dloop ; условный переход. должен быть последним в блоке

IT NE ; следующая инструкция – условная

ADD R0, R0, R1 ; синтаксическая ошибка: не указан код условия в блоке IT.

9.8.4 ТВВ и ТВН

Табличный переход по индексу.

Синтаксис

ТВВ [Rn, Rm]

ТВН [Rn, Rm, LSL #1]

где Rn – регистр, содержащий адрес таблицы длин переходов. Если в качестве регистра Rn используется PC, то первый байт таблицы переходов следует непосредственно после инструкции ТВВ или ТВН.

Rm – регистр, содержащий индекс в таблице переходов. Для таблиц, содержащих полуслова, добавляется операция сдвига LSL #1, что обеспечивает корректную адресацию по смещению в таблице.

Описание

Данные инструкции позволяют выполнить переход вперед относительно текущего значения счетчика команд PC на заданное смещение, выбранное из таблицы смещений, имеющих размер байта (для команды ТВВ) или полуслова (для команды ТВН).

Регистр Rn содержит указатель на начало таблицы, а регистр Rm – индекс требуемого элемента.

Для команды ТВВ смещение вычисляется путем умножения на два значения байта из заданной ячейки таблицы, интерпретируемого как целое число без знака.

Для команды ТВН смещение вычисляется путем умножения на два значения полуслова из заданной ячейки таблицы, интерпретируемого как целое число без знака.

Передача управления по соответствующему смещению осуществляется немедленно после выполнения инструкции ТВВ или ТВН.

Ограничения

- в качестве регистра Rn нельзя использовать SP;
- в качестве регистра Rm нельзя использовать SP и PC;
- при использовании инструкций ТВВ или ТВН внутри IT-блока они должны быть последней командой блока.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

ADR.W R0, BranchTable_Byte

ТВВ [R0, R1] ; R1 – индекс, R0 – базовый адрес таблицы переходов

Case1 ; код для варианта R1 = 0

Case2 ; код для варианта R1 = 1

Case3 ; код для варианта R1 = 2

BranchTable_Byte

DCB 0 ; смещение для Case1

DCB ((Case2-Case1)/2) ; смещение для Case2

DCB ((Case3-Case1)/2) ; смещение для Case3

TBH [PC, R1, LSL #1] ; R1 – индекс, таблица переходов расположена непосредственно после команды TBH

BranchTable_H

DCI ((CaseA - BranchTable_H)/2) ; смещение для CaseA

DCI ((CaseB - BranchTable_H)/2) ; смещение для CaseB

DCI ((CaseC - BranchTable_H)/2) ; смещение для CaseC

CaseA ; код для CaseA

CaseB ; код для CaseB

CaseC ; код для CaseC.

9.9 Прочие инструкции

В таблице ниже представлен список инструкций процессора RISC, не рассмотренных в предыдущих разделах.

Таблица 9-13 – Прочие инструкции

Мнемокод	Краткое описание
BKPT	Точка останова
CPSID	Изменить состояние процессора, запретить прерывания
CPSIE	Изменить состояние процессора, разрешить прерывания
DMB	Барьер синхронизации доступа к памяти данных
DSB	Барьер синхронизации доступа к памяти данных
ISB	Барьер синхронизации доступа к инструкциям
MRS	Загрузка из специального регистра в регистр общего назначения
MSR	Записать регистр общего назначения в специальный регистр
NOP	Нет операции
SEV	Установить признак события
SVC	Вызов супервизора
WFE	Ожидать событие
WFI	Ожидать прерывание

9.9.1 CPS

Изменить состояние процессора.

Синтаксис

CPS*effect* iflags

где *effect* – один из возможных суффиксов:

IE – сбрасывает специальный регистр в 0;

ID – устанавливает специальный регистр в 1.

iflags – последовательность флагов:

i – сбрасывает или устанавливает регистр PRIMASK;

f – сбрасывает или устанавливает регистр FAULTMASK.

Описание

Команда CPS позволяет изменить значение специальных регистров PRIMASK и FAULTMASK. Подробности см. в пункте 8.4.9 «Регистр маски исключений Exception mask».

Ограничения

- команда CPS доступна только из привилегированного приложения, при вызове из непривилегированного приложения она игнорируется;
- команда CPS не допускает условного исполнения и, таким образом, не должна использоваться внутри блока IT.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

CPSID i ; Запретить прерывания и конфигурируемые обработчики отказов
 CPSID f ; Запретить прерывания и все обработчики отказов
 CPSIE i ; Разрешить прерывания и конфигурируемые обработчики отказов
 CPSIE f ; Разрешить прерывания и все обработчики отказов.

9.9.2 DMB

Барьер синхронизации доступа к памяти данных.

Синтаксис

DMB{cond}

где cond – необязательный код условия, см. пункт 9.2.7 «Условное исполнение».

Описание

Команда DMB выполняет функцию барьерной синхронизации доступа к памяти данных. Она гарантирует, что все явные операции доступа к памяти, которые были инициированы перед выполнением инструкции DMB, будут завершены до того, как начнется выполнение любой операции доступа к памяти после этой инструкции.

Команда DMB не влияет на очередность и порядок выполнения инструкций, не выполняющих доступа к памяти.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

DMB ; Барьер синхронизации доступа к памяти данных.

9.9.3 DSB

Барьер синхронизации доступа к памяти данных.

Синтаксис

DSB{cond}

где cond – необязательный код условия, см. пункт 9.2.7 «Условное исполнение».

Описание

Инструкция DSB выполняет функцию барьерной синхронизации доступа к памяти данных. Команды, которые будут следовать в порядке выполнения после DSB, не начнут исполняться до ее завершения. Инструкция DSB завершает свою работу после того, как будут выполнены все инициированные перед ней явные операции доступа к памяти.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

DSB ; Data Synchronisation Barrier.

9.9.4 ISB

Барьер синхронизации доступа к инструкциям.

Синтаксис

ISB{cond}

где cond – необязательный код условия, см. пункт 9.2.7 «Условное исполнение».

Описание

Команда ISB выполняет функцию барьерной синхронизации выполнения команд. Она осуществляет сброс конвейера инструкций процессора, гарантируя таким образом, что все команды, расположенные после инструкции ISB, по окончании ее исполнения будут загружены в конвейер повторно.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

ISB ; Барьер синхронизации доступа к инструкциям.

9.9.5 MRS

Считать содержимое специального регистра в регистр общего назначения.

Синтаксис

MRS{cond} Rd, spec_reg

где cond – необязательный код условия, см. пункт 9.2.7 «Условное исполнение».

Rd – регистр-получатель результата.

spec_reg – один из специальных регистров: APSR, IPSR, EPSR, IEPSR, IAPSR, EAPSR, PSR, MSP, PSP, PRIMASK, BASEPRI, BASEPRI_MAX, FAULTMASK или CONTROL.

Описание

Инструкции MRS совместно с MSR используются для чтения/модификации/записи элементов PSR, например, для сброса флага Q.

В коде, отвечающем за переключение процессов, необходимо обеспечить сохранение состояния приостановленного процесса, и восстановление состояния активизированного процесса. Необходимой составной частью сохраняемой (восстанавливаемой) информации является значение регистра PSR. При этом на этапе сохранения состояния используется команда MRS, а на этапе восстановления – команда MSR.

При использовании команды MRS регистр BASEPRI_MAX является синонимом регистра BASEPRI. См. также описание инструкции MSR.

Ограничения

В качестве регистра-получателя Rd нельзя использовать SP или PC.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

MRS R0, PRIMASK ; Считать значение PRIMASK и записать значение в R0.

9.9.6 MSR

Записать регистр общего назначения в специальный регистр.

Синтаксис

MSR{cond} спец_reg, Rn

где cond – необязательный код условия, см. пункт 9.2.7 «Условное исполнение».

Rn – регистр-источник данных.

спец_reg – один из специальных регистров: APSR, IPSR, EPSR, IEPSR, IAPSR, EAPSR, PSR, MSP, PSP, PRIMASK, BASEPRI, BASEPRI_MAX, FAULTMASK или CONTROL.

Описание

Доступ к специальным регистрам в команде MSR различен для привилегированных и непривилегированных приложений. Непривилегированному приложению доступен только регистр APSR (см. пункт 8.4.6 «Программный регистр состояния приложения APSR»). При этом попытки записи в нераспределенные биты, а также в EPSR игнорируются.

Привилегированное приложение имеет доступ ко всем специальным регистрам.

При записи данных в регистр BASEPRI_MAX инструкция записывает данные в регистр BASEPRI только при выполнении одного из условий:

- Rn не равен нулю и текущее значение BASEPRI равно 0;
- Rn не равен нулю и меньше текущего значения BASEPRI.

См. также описание инструкции MRS.

Ограничения

В качестве регистра-источника данных Rn нельзя использовать SP или PC.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

MSR CONTROL, R1 ; Записать значение регистра R1 в регистр CONTROL

9.9.7 NOP

Нет операции.

Синтаксис

NOP{cond}

где cond – необязательный код условия, см. пункт 9.2.7 «Условное исполнение».

Описание

Инструкция NOP ничего не делает. В частности, эта инструкция в некоторых случаях может быть автоматически исключена из конвейера команд, и таким образом, выполнена за ноль тактов. Команду NOP рекомендуется использовать для заполнения, например, с целью разместить очередную инструкцию по адресу, выровненному по 64-битной границе.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

NOP ; нет операции

9.9.8 SEV

Установить признак события.

Синтаксис

SEV{cond}

где cond – необязательный код условия, см. пункт 9.2.7 «Условное исполнение».

Описание

Инструкция SEV используется для передачи информации о событии всем процессорам в составе многопроцессорной системы. Кроме того, он устанавливает собственный регистр события в 1.

См. также подраздел 36.8 «Управление электропитанием».

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

SEV ; Послать признак события.

9.9.9 SVC

Вызов супервизора.

Синтаксис

SVC{cond} #imm

где cond – необязательный код условия, см. пункт 9.2.7 «Условное исполнение».

Imm – константное выражение, целое число в диапазоне от 0 до 255 (восемьбитное число).

Описание

Инструкция SVC вызывает формирование исключения SVC. Параметр imm игнорируется процессором. При необходимости он может быть получен обработчиком исключения для определения запрошенного приложением варианта обслуживания.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

SVC 0x32 ; Вызов супервизора (обработчик SVC может извлечь параметр по сохраненному в стеке, адресу PC приложения).

9.9.10 WFE

Ожидать событие.

Синтаксис

WFE{cond}

где cond – необязательный код условия, см. пункт 9.2.7 «Условное исполнение».

Описание

В случае, если регистр события равен 0, выполнение команды WFE приводит к приостановке исполнения команд до тех пор, пока не произойдет одно из следующих событий:

- исключение, не запрещенное путем установки маски или текущим уровнем приоритета;
- перевод исключения в состояние ожидания обслуживания при установленном в 1 бите SEVONPEND регистра управления системой SCR;
- получение запроса на переход в режим отладки, в случае, если отладка разрешена;
- получение сигнала о событии от периферийного устройства или от другого процессора (по команде SEV) в многопроцессорной системе.

В случае, если регистр события равен 1, команда WFE сбрасывает его в 0, после чего завершает свое функционирование без приостановки процессора.

Более подробная информация отражена в подразделе 36.8 «Управление электропитанием».

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

WFE; Ожидание события.

9.9.11 WFI

Ожидание прерывание.

Синтаксис

WFI{cond}

где cond – необязательный код условия, см. пункт 9.2.7 «Условное исполнение».

Описание

Команда WFI приостанавливает процессор до тех пор, пока не произойдет одно из следующих событий:

- исключение;
- запрос на перевод в режим отладки, вне зависимости от того, разрешен или запрещен этот режим.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

WFI ; Ожидание прерывания.

10 Системный таймер SysTick

Процессор имеет 24-разрядный системный таймер, SysTick, который считает вниз от загруженного в него значения до нуля; перезагрузка (возврат в начало) значения в регистр LOAD происходит по следующему фронту синхросигнала, затем счет продолжается по последующему фронту.

Когда процессор остановлен для отладки, таймер не декрементируется.

10.1 Описание регистров системного таймера SysTick

Таблица 10-1 – Описание регистров системного таймера SysTick

Адрес	Название	Тип	Доступ	Значение после сброса	Описание
0xE000E000	SysTick				Системный таймер SYSTICK
0x000	CTRL	RW	привилегированный	0x00000000	SysTick->CTRL
0x004	LOAD	RW	привилегированный	0x00000000	SysTick->LOAD
0x008	VAL	RW	привилегированный	0x00000000	SysTick->VAL
0x00C	CALIB	RO	привилегированный	0x00000000	SysTick->CAL

10.1.1 SysTick->CTRL

Регистр CTRL разрешает основные функции системного таймера.

Назначение бит:

Таблица 10-2 – Регистр контроля и статуса CTRL

Номер	31...17	16	15...3	2	1	0
Доступ	U	RO	U	R/W	R/W	R/W
Сброс		0		0	0	0
	-	COUNTFLAG	-	CLKSOURCE	TICKINT	ENABLE

COUNTFLAG

Возвращает 1, если таймер досчитал до нуля с последнего момента чтения.

CLKSOURCE

Указывает источник синхросигнала:

0 – LSI;

1 – HCLK

TICKINT

Разрешает запрос на прерывание от системного таймера:

0 – таймер досчитает до нуля и прерывание не возникнет;

1 – таймер досчитывает до нуля и возникает запрос на прерывание.

Программное обеспечение может использовать бит COUNTFLAG, чтобы определить, досчитал таймер до нуля или нет.

ENABLE

Разрешает работу таймера:

0 – работа таймера запрещена;

1 – работа таймера разрешена.

Когда ENABLE установлен в единицу, таймер загружает значение RELOAD из регистра LOAD и затем начинает декрементироваться. По достижению значения 0 таймер устанавливает бит COUNTFLAG и в зависимости от TCKINT генерирует запрос на прерывание. Затем загружается значение RELOAD и продолжается счет.

10.1.2 SysTick->LOAD

Регистр LOAD устанавливает стартовое значение, загружаемое в регистр VAL.

Таблица 10-3 – Регистр перегружаемого значения LOAD

Номер	31...24	23...0
Доступ		
Сброс		
	-	RELOAD

RELOAD

Значение, загружаемое в регистр VAL, когда таймер разрешен и когда достигается значение нуля.

Расчет значения RELOAD

Значение RELOAD может быть любым в диапазоне 0x00000001-0x00FFFFFF. Значение 0 допустимо, но не оказывает эффекта, потому что запрос на прерывание и активизация бита COUNTFLAG происходит только при переходе таймера из состояния 1 в 0.

Расчет значения RELOAD происходит в соответствии с использованием таймера:

– Для формирования мультикороткого таймера с периодом N процессорных циклов применяется значение RELOAD, равное N-1. Например, если требуется прерывание каждые 100 циклов, то устанавливается значение RELOAD, равное 99;

– Для формирования одиночного прерывания после задержки в N тактов процессора используется значение N. Например, если требуется прерывание после 400 тактов процессора, то устанавливается RELOAD, равное 400.

10.1.3 SysTick->VAL

Регистр VAL содержит текущее значение системного таймера.

Таблица 10-4 – Регистр текущего значения таймера VAL

Номер	31...24	23...0
Доступ		
Сброс		
	-	CURRENT

CURRENT

Чтение возвращает текущее значение системного таймера.

Запись любого значения очищает регистр в ноль, и также очищает бит COUNTFLAG регистра CTRL.

10.1.4 SysTick->CAL

Регистр CALIB показывает калибровочное значение системного таймера. Калибровка системного таймера не реализована

Таблица 10-5 – Регистр калибровочного значения таймера CAL

Номер	31	30	29...24	23...0
Доступ				
Сброс				
	NOREF	SKEW	-	TENMS

NOREF

Читается как ноль.

SKEW

Читается как ноль.

TENMS

Читается как 0

10.2 Советы и особенности при применении системного таймера

Системный таймер работает от процессорного синхросигнала. Если синхросигнал останавливается в режиме пониженного энергопотребления, то системный таймер останавливается.

Гарантируйте, чтобы программа использовала доступ к регистрам системного таймера, выровненный по словам.

11 Модуль защиты памяти

Этот раздел описывает модуль защиты памяти (MPU).

MPU делит карту памяти на регионы, и определяет положение, размер, разрешение на доступ и атрибуты памяти для каждого из них. Поддерживается:

- независимая установка атрибута для каждого региона;
- наложение (перекрывание) регионов;
- экспортирование атрибутов памяти в систему.

Атрибуты памяти влияют на доступ к памяти в регионе. В ядре RISC определено:

- восемь независимых регионов, 0–7;
- фоновый регион.

Если регионы памяти перекрываются, на доступ к памяти влияют атрибуты региона с большим номером. Например, атрибуты региона 7 получают первенство над атрибутами любых других регионов, перекрывающихся с 7.

Фоновый регион имеет такие же атрибуты доступа к памяти, как и default карта памяти, но доступен только через привилегированные инструкции программы.

Карта памяти RISC унифицированная. Это означает, что атрибуты доступа к инструкциям и данным одинаковые.

Если происходит программный запрос в запрещенную область памяти MPU, процессор генерирует ошибку управления памятью. Это вызывает прерывание по ошибке и может вызвать прерывание процессов в переменном окружении OS.

В переменном окружении OS, ядро может обновлять настройки MPU региона динамически, основываясь на выполняемых процессах. Обычно встроенные OS используют MPU для защиты памяти.

Конфигурация MPU регионов основывается на типе памяти, см. подпункт 6.4.1.1 «Регионы памяти, типы и атрибуты RISC».

Таблица 11-1 показывает возможные атрибуты MPU регионов. Здесь включены такие атрибуты памяти, как *shareable* и кэшируемость, которые не существенны во многих реализациях микросхем.

Таблица 11-1 – Обзор атрибутов памяти

Тип памяти	Атрибут shareable	Другие атрибуты	Описание
Строго упорядоченная	-	-	Весь доступ к строго упорядоченной памяти осуществляется под программным управлением. Все строго упорядоченные регионы могут быть общими
Устройство	Общая	-	Общая периферийная память для нескольких процессоров
	Не общая	-	Периферийная память только для одного процессора
Обычная	Общая		Обычная общая память для нескольких процессоров
	Не общая		Обычная память только для одного процессора

11.1 Описание регистров MPU

Применяются следующие MPU регистры для определения регионов и их атрибутов.

Таблица 11-2 – Обзор регистров MPU

Адрес	Обозначение	Тип	Доступ	Значение после сброса	Описание
0xE000ED90	MPU				Модуль защиты памяти MPU
0x000	TYPE	RO	привилегированный	0x00000800	MPU->TYPE
0x004	CTRL	RW	привилегированный	0x00000000	MPU->CTRL
0x008	RNR	RW	привилегированный	0x00000000	MPU->RNR
0x00C	RBAR	RW	привилегированный	0x00000000	MPU->RBAR
0x010	RASR	RW	привилегированный	0x00000000	MPU->RASR
0x014	RBAR_A1	RW	привилегированный	0x00000000	Обозначение RBAR
0x018	RASR_A1	RW	привилегированный	0x00000000	Обозначение RASR
0x01C	RBAR_A2	RW	привилегированный	0x00000000	Обозначение RBAR
0x020	RASR_A2	RW	привилегированный	0x00000000	Обозначение RASR
0x24	RBAR_A3	RW	привилегированный	0x00000000	Обозначение RBAR
0x28	RASR_A3	RW	привилегированный	0x00000000	Обозначение RASR

11.1.1 MPU->TYPE

Регистр TYPE показывает, присутствует или нет MPU, и как много регионов поддерживается.

Таблица 11-3 – Регистр TYPE

Номер	31...24	23...16	15...8	7...1	0
Доступ					
Сброс					
	-	IREGION	DREGION	-	SEPARATE

IREGION

Указывает количество поддерживаемых MPU регионов инструкций.

Всегда содержит 0x00. Карта памяти MPU унифицированная и описывается полем DREGION.

DREGION

Указывает количество поддерживаемых MPU регионов данных.

0x08 – Восемь MPU регионов.

SEPARATE

Указывает, поддерживается унифицированная или отдельная карта памяти для инструкций и данных:

0 – унифицированная.

11.1.2 MPU->CTRL

Регистр CTRL:

- разрешает MPU;
- разрешает default карту памяти как фоновый регион;
- разрешает применение MPU, при возникновении аппаратной ошибки, немаскируемое прерывание (NMI), FAULTMASK вызываемый обработчик.

Таблица 11-4 – Регистр CTRL

Номер	31...4	3	2	1	0
Доступ					
Сброс					
	-	PRIVDEFENA		HFNMIEA	ENABLE

PRIVDEFENA

Разрешает привилегированный программный доступ к default карте памяти:

0 – если MPU разрешен, запрещение применяется к default карте памяти. Любой доступ к памяти, не покрываемой разрешенным регионом, вызывает ошибку;

1 – если MPU разрешен, разрешает применение default карты памяти как фонового региона для привилегированного программного доступа.

Когда разрешено, фоновый регион считается как номер региона -1. Любой регион, который определен и разрешен, имеет приоритет выше этой default памяти.

Если MPU запрещен, то процессор игнорирует этот бит.

HFNMIEA

Разрешает операции MPU во время возникновения аппаратной ошибки, NMI, и FAULTMASK обработчик.

Если MPU разрешен:

0 – MPU запрещен во время возникновения аппаратной ошибки, NMI, FAULTMASK обработчик, несмотря на значения бита ENABLE;

1 – MPU разрешен во время возникновения аппаратной ошибки, NMI, FAULTMASK обработчик.

Если MPU запрещен и этот бит устанавливается в единицу, то поведение непредсказуемо.

ENABLE

Разрешает MPU:

0 – MPU запрещен;

1 – MPU разрешен.

Если ENABLE и PRIVDEFENA одновременно установлены в «1»:

Любой неадресованный доступ привилегированным программным обеспечением к разрешенному региону, ведет себя как определено default картой памяти. Любой неадресованный доступ непривилегированным программным обеспечением к разрешенному региону вызывает ошибку управления памятью.

XN и строго упорядоченные правила всегда применяются к управляющему системному пространству несмотря на значение бита ENABLE.

Когда ENABLE установлен в «1», по крайней мере, один регион карты памяти должен быть разрешен для системных функций, за исключением PRIVDEFENA установлен в «1». Если PRIVDEFENA установлен в «1» и нет разрешенных регионов, тогда только привилегированное программное обеспечение может исполняться.

Когда ENABLE установлен в «0», система использует default карту памяти. Это аналогично памяти с атрибутами, как если бы MPU не применялся. К default карте памяти доступ осуществляется как с помощью привилегированного, так и непривилегированного программного обеспечения.

Когда MPU разрешен, доступ к системному пространству управления и таблице векторов всегда разрешен. К другим областям доступ базируется на регионах и состоянии бита PRIVDEFENA.

За исключением случая HFNMIENA установленного в 1, MPU не разрешает процессору выполнять обработчики прерываний с приоритетом -1 или -2. Эти приоритеты допустимы только когда обрабатывается прерывание аппаратной ошибки или NMI, или когда FAULTMASK разрешен. Установка бита HFNMIENA в «1» разрешает действовать этим двум приоритетам.

11.1.3 MPU->RNR

Регистр RNR выбирает, на какой регион памяти ссылаются регистры RBAR и RASR.

Таблица 11-5 – Регистр номера региона RNR

Номер	31...8	7...0
Доступ		
Сброс		
	-	REGION

REGION

Указывает MPU регион, на который ссылаются регистры RBAR и RASR.

MPU поддерживает 8 регионов памяти, поэтому разрешенное значение для этого поля от 0 до 7.

Обычно вы записываете требуемое значение номера региона в этот регистр перед обращением в RBAR и RASR. Однако вы можете изменить номер региона записью в RBAR с установленным в единицу битом VALID. Эта запись обновляет значение поля REGION.

11.1.4 MPU->RBAR

Регистр RBAR определяет базовый адрес MPU региона, выбранного RNR, вы можете изменить значение RNR. Запись RBAR с битом VALID установленным в «1» изменяет текущий номер региона и обновляет RNR.

Таблица 11-6 – Регистр базового адреса региона RBAR

Номер	31...N	N-1...6	5	4	3...0
Доступ					
Сброс					
	ADDR	-	VALID		REGION

ADDR

Поле базового адреса региона. Значение N зависит от размера региона. Для более подробной информации см. «Поле ADDR».

VALID

Бит верности номера региона MPU:

Запись:

0 – RNR не изменяется, и процессор:

- обновляет базовый адрес для региона, определенного в RNR;
- игнорирует значение поля REGION.

1 – процессор:

- обновляет значение RNR на значение из поля REGION;
- обновляет базовый адрес региона, определенного в поле REGION.

Всегда читается как ноль.

REGION

Поле MPU региона:

- поведение при записи описано выше (см. описание бита VALID);
- при чтении возвращает текущий номер региона, который определен в регистре RNR.

Поле ADDR

Поле ADDR это [31:N] бит регистра RBAR. Размер региона определяется полем SIZE в регистре RASR, как

$$N = \text{Log}_2 (\text{Размер региона в байтах}),$$

Если размер региона сконфигурирован равным 4 ГБ, в RASR, то значение поля ADDR неверно. В этом случае, регион занимает всю карту памяти, и базовый адрес его равен 0x00000000.

Базовый адрес выравнивается под размер региона. Например, 64 КБ регион должен быть кратно 64 КБ, например, 0x00010000 или 0x00020000.

11.1.5 MPU->RASR

Регистр RASR определяет размер и атрибуты памяти MPU региона, выбранного RNR, а также разрешает регион и любые подрегионы.

RASR доступен в режиме слова или полуслова:

- старшее значащее полуслово содержит атрибуты региона;
- младшее значащее полуслово содержит размер региона и биты разрешения региона и подрегионов.

Таблица 11-7 – Назначение бит регистра RASR

Номер	31	30	29	28	27	26	24	23	22	21	19
Доступ											
Сброс											
	-		XN	-		AP		-		TEX	

Номер	18	17	16	15	8	7	6	5	1	0
Доступ										
Сброс										
	S	C	B	SRD		-		SIZE		ENABLE

XN

Бит запрещения доступа инструкций:

- 0 – выборка инструкций разрешена;
- 1 – выборка инструкций запрещена.

AP

Поле разрешения доступа, см. Таблица 11-11 – Кодирование привилегий доступа в поле AP.

TEX, C, B

Атрибуты доступа к памяти, см. Таблица 11-9 – Кодирование бит разрешения доступа.

S

Бит общего доступа, см. Таблица 11-8 – Пример значений поля SIZE.

SRD

Бит запрещения подрегиона. Для каждого бита в этом поле:

- 0 – соответствующий подрегион разрешен;
- 1 – соответствующий подрегион запрещен.

Для более подробной информации см. «Подрегионы».

Регион размером 128 байт и менее не поддерживает подрегионы. Когда записываются атрибуты для такого региона, записывайте поле SRD равным 0x00.

SIZE

Определяет размер MPU региона. Минимальное разрешенное значение 3(b00010), для более подробной информации см. «Значения поля SIZE».

ENABLE

Бит разрешения региона.

Для более подробной информации о разрешении доступа, см. «Атрибуты разрешения доступа MPU».

Значения поля SIZE

Поле SIZE определяет размер памяти MPU региона выбранного регистром RNR следующим образом:

$$(\text{Region size in bytes}) = 2(\text{SIZE} + 1)$$

Наименьший разрешенный размер региона 32 байт, соответствует значению SIZE, равному 4. Таблица 11-8 представляет примеры значений SIZE, соответствующие размеру региона и значению N регистра RBAR.

Таблица 11-8 – Пример значений поля SIZE

Значение SIZE	Размер региона	Значение N*	Комментарий
b00100 (4)	32 байт	5	Минимальный разрешенный размер
b01001 (9)	1 Кбайт	10	-
b10011 (19)	1 Мбайт	20	-
b11101 (29)	1 Гбайт	30	-
b11111 (31)	4 Гбайт	b01100	Максимальный разрешенный размер
* Содержится в RBAR, см. пункт 11.1.4 «MPU->RBAR»			

11.1.6 Атрибуты разрешения доступа MPU

Раздел описывает атрибуты разрешения доступа. Биты разрешения доступа TEX, C, B, S, AP и XN регистра RASR контролируют доступ к соответствующему региону памяти. Если происходит доступ к области памяти без разрешения доступа, то MPU генерирует ошибку доступа.

Таблица 11-9 – Кодирование бит разрешения доступа TEX, C, B, S

TEX	C	B	S	Тип памяти	Возможность общего доступа	Другие атрибуты
b000	0	0	x*	Строго упорядоченная	Общий доступ	
		1	x*	Устройство	Общий доступ	
	1	0	0	Обычная	Не общий доступ	Внешний и внутренний кэш, синхронное обновление памяти.
			1		Общий доступ	Запись без кэширования
		1	0	Обычная	Не общий доступ	Внешний и внутренний кэш, отложенное обновление памяти.
			1		Общий доступ	Запись без кэширования

TEX	C	B	S	Тип памяти	Возможность общего доступа	Другие атрибуты
b001	0	0	0	Обычная	Не общий доступ	
			1		Общий доступ	
		1	x*	Зарезервировано		-
	1	0	x*	Реализация определяется атрибутами		-
			1	0	Обычная	Не общий доступ
		1				Общий доступ
b010	0	0	x*	Устройство	Не общий доступ	Индивидуальное устройство
			1			
	1	x*	x*	Зарезервировано		-
b1BB	A	A	0	Обычное	Не общий доступ	
			1		общий доступ	
* MPU игнорирует значение этих бит						

Таблица 11-10 поясняет кодирование режима кэша атрибутом TEX в диапазоне значений атрибута от 4 до 7.

Таблица 11-10 – Кодирование режима кэша атрибутом TEX

Значение AA или BB при TEX=1xx	Соответствующий режим кэша
00	Не кэшируемая
01	Отложенное обновление, запись и чтение пакетные
10	Синхронное обновление, запись без кэширования
11	Отложенное обновление, запись без кэширования

Таблица 11-11 поясняет кодирование бит AP, определяющих разрешение на доступ для привилегированного и непривилегированного программного обеспечения (ПО).

Таблица 11-11 – Кодирование привилегий доступа в поле AP

AP[2:0]	Привилегирован- ный доступ	Непривилегирован- ный доступ	Описание
000	нет доступа	нет доступа	Любой доступ приводит к ошибке доступа
001	RW	нет доступа	Доступ только для привилегированного ПО
010	RW	RO	Запись непривилегированным ПО приводит к ошибке доступа
011	RW	RW	Полный доступ
100	непредсказуемо	непредсказуемо	Зарезервировано
101	RO	нет доступа	Чтение только привилегированным ПО

AP[2:0]	Привилегированный доступ	Непривилегированный доступ	Описание
110	RO	RO	Только чтение и привилегированным, и непривилегированным ПО
111	RO	RO	Только чтение и привилегированным, и непривилегированным ПО

11.1.7 Несоответствие MP

Когда происходит нарушение разрешения доступа MPU, процессор генерирует ошибку управления памятью, см. раздел 36 «Прерывания и исключения RISC». Регистр MMFSR указывает причину ошибки. Для более подробной информации см. пункт «Поле MMFSR».

11.1.8 Обновление MPU региона

Атрибуты для MPU региона обновляют через регистры RNR, RBAR и RASR. Вы можете программировать каждый регистр независимо или использовать для программирования возможность множественной записи всех этих регистров. Вы можете использовать обозначения RBAR и RASR, чтобы запрограммировать до 4 регионов одновременно, используя инструкцию STM.

Обновление MPU региона через отдельные регистры

Простой код для одного региона:

```
; R1 = номер региона
; R2 = размер/разрешение
; R3 = атрибуты
; R4 = адрес
LDR R0,=MPU_RNR    ; 0xE000ED98, регистр номера региона MPU
STR R1, [R0, #0x0]  ; номер региона
STR R4, [R0, #0x4]   ; базовый адрес региона
STRH R2, [R0, #0x8]  ; размер региона и разрешение
STRH R3, [R0, #0xA]  ; атрибуты региона
```

Запрещение региона перед записью новых настроек в MPU, если этот регион перед этим был разрешен. Например:

```
; R1 = номер региона
; R2 = размер/разрешение
; R3 = атрибуты
; R4 = адрес
LDR R0,=MPU_RNR    ; 0xE000ED98, регистр номера региона MPU
STR R1, [R0, #0x0]  ; номер региона
BIC R2, R2, #1      ; запрещение
STRH R2, [R0, #0x8] ; размер региона и разрешение
STR R4, [R0, #0x4]   ; базовый адрес региона
STRH R3, [R0, #0xA] ; атрибуты региона
ORR R2, #1          ; разрешение
STRH R2, [R0, #0x8] ; размер региона и разрешение.
```

Программное обеспечение должно применить barrier инструкции:

- если перед установкой MPU будет невыполненная пересылка в память, такая как буферная запись, то это может повлиять на изменение настроек MPU;
- после установки MPU, если это включает пересылку в память, должны использоваться новые настройки MPU.

Однако не требуются barrier инструкции памяти, если процесс установки MPU начинается с помощью входа в обработчик прерывания, или сопровождается возвращением из прерывания, потому что вход и выход из прерывания сопровождается механизмом barrier для памяти.

Программному обеспечению не требуются barrier инструкции памяти во время установки MPU, потому что этот доступ осуществляется через PPB, который строго упорядоченный регион памяти.

Например, если вы хотите, чтобы все изменения доступа к памяти имели место непосредственно после программной последовательности, используйте инструкции DSB и ISB. Инструкции DSB требуются после изменения настроек MPU, в конце переключения контекста. Инструкции ISB требуются, если код, который программирует MPU регион или регионы вызывается с использованием инструкций перехода (branch) или вызова подпрограммы (call). Если программная последовательность вызывается инструкцией выхода из прерывания (return), или прерыванием, то ISB не требуется.

Обновление MPU региона через множественную запись регистров

Вы можете программировать напрямую, используя запись множества регистров, в зависимости от того, как распределена информация.

; R1 = номер региона

; R2 = адрес

; R3 = размер, атрибуты

LDR R0, =MPU_RNR ; 0xE000ED98, регистр номера региона MPU

STR R1, [R0, #0x0] ; номер региона

STR R2, [R0, #0x4] ; базовый адрес региона

STR R3, [R0, #0x8] ; атрибут региона, размер и разрешение.

Оптимизация при использовании STM инструкции:

; R1 = номер региона

; R2 = адрес

; R3 = размер, атрибуты

LDR R0, =MPU_RNR ; 0xE000ED98, регистр номера региона MPU

STM R0, {R1-R3} ; номер региона, адрес, атрибут, размер и разрешение.

Вы можете использовать два слова для предварительной упаковки информации. Это значит, что RBAR содержит требуемый номер региона и имеет бит VALID, установленный в единицу, см. пункт 11.1.4 «MPU->RBAR». Это применимо, если данные упакованы статически, например, в начальном загрузчике:

; R1 = адрес и номер региона;

; R2 = размер и атрибуты;

LDR R0, =MPU_RBAR ; 0xE000ED9C, регистр базового адреса MPU

STR R1, [R0, #0x0] ; базовый адрес и номер региона, совмещенные с битом VALID, установленным в 1
 STR R2, [R0, #0x4] ; атрибут региона, размер и разрешение.
 Оптимизация при использовании STM инструкции:
 ; R1 = адрес и номер региона
 ; R2 = размер и атрибуты
 LDR R0, MPU_RBAR ; 0xE000ED9C, регистр базового адреса MPU
 STM R0, {R1-R2} ; базовый адрес региона, номер региона и бит VALID, и атрибут региона, размер и разрешение.

Подрегионы

Регионы величиной в 256 байт или более делятся на восемь равных подрегионов. Установите соответствующий бит в поле SRD регистра RASR для запрещения подрегиона, см. пункт 11.1.5 «MPU->RASR». Младший значащий бит SRD контролирует первый подрегион, и старший значащий бит контролирует последний подрегион. Запрещение подрегиона означает, что другой регион перекрывает запрещенную область. Если другой разрешенный регион не перекрывает запрещенный регион, то MPU вырабатывает ошибку.

Регионы размером 32, 64 и 128 не поддерживают подрегионы, с этими регионами вы должны установить поле SRD равным 0x00, иначе поведение MPU непредсказуемо.

Пример применения SRD

Два региона с одинаковым базовым адресом перекрываются. Регион размером 128 Кбайт и регион размером 512 Кбайт. Убедитесь, что атрибуты для региона один установлены для первых 128 Кбайт, установите SRD поле для региона два в значение b00000011 для запрещения первых двух подрегионов, как показывает рисунок 11-1.

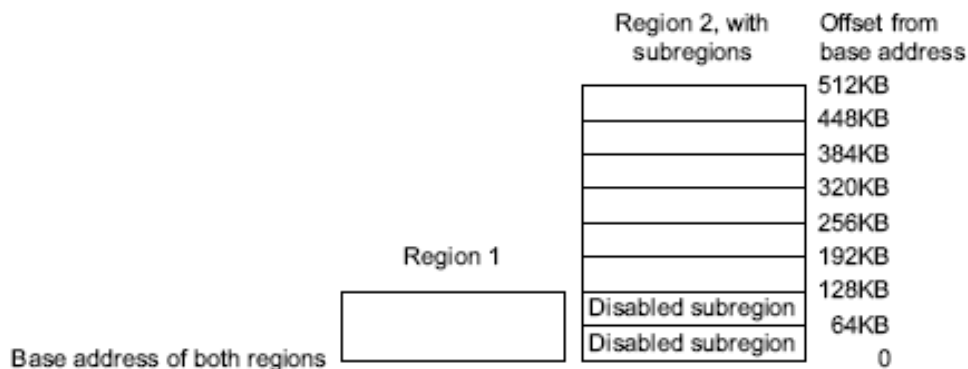


Рисунок 11-1 – Применение SRD

11.2 Советы и особенности применения MPU

Во избежание непредвиденных ситуаций, запретите прерывания перед обновлением атрибутов региона, к которому может осуществляться доступ в обработчике прерываний.

Убедитесь, что программное обеспечение использует корректный доступ, соответствующий размеру регистров MPU:

- за исключением RASR, необходимо использовать доступ по словам;
- для RASR может использоваться доступ по байтам, полусловам или словам.

Процессор не поддерживает невыровненный доступ к регистрам MPU.

Если MPU перенастраивается, то запретите неиспользуемые регионы для предотвращения любых предыдущих настроек регионов от их влияния на новые настройки.

Конфигурация MPU для микросхемы

Обычно, микроконтроллерные системы имеют только один процессор и не имеют кэша. В таких системах MPU программируется следующим образом:

Таблица 11-12 – Атрибуты регионов памяти для микросхем

Регион памяти	TEX	C	B	S	Типа памяти и атрибут
Флеш-память	b000	1	0	0	Обычная память, не общий доступ, сквозная запись
Внутренняя SRAM	b000	1	0	1	Обычная память, общий доступ, сквозная запись
Внешняя SRAM	b000	1	1	1	Обычная память, общий доступ, обратная запись, выделенная запись
Периферия	b000	0	1	1	Память устройства, общий доступ

В большинстве микроконтроллерных приложений, установка атрибутов общего доступа и кэширования не влияет на поведение системы. Однако применение этих настроек для MPU регионов может сделать код приложений более переносимым. Это имеет большую важность в обычных ситуациях. В специальных системах, таких как многопроцессорные или с отдельным DMA устройством, атрибуты общего доступа очень важны. В этих случаях обращайтесь к рекомендациям производителей устройств памяти.

12 Сигналы тактовой частоты

Микросхема имеет два встроенных генератора, два внешних осциллятора и специализированный блок формирования тактовой синхронизации микросхемы.

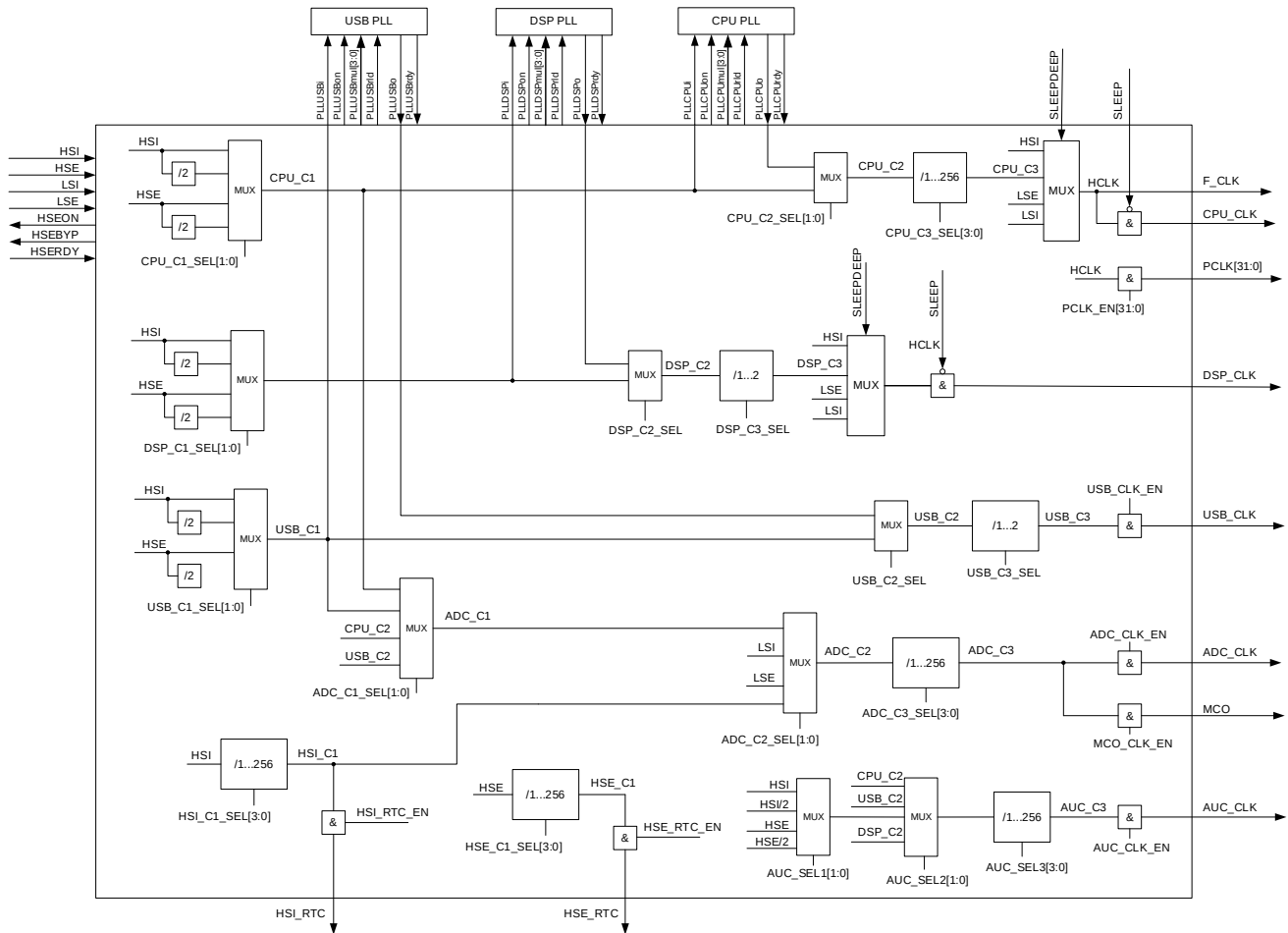


Рисунок 12-1 – Структурная блок-схема формирования тактовой частоты

Управление тактовыми частотами ведется через периферийный блок RST_CLK. При включении питания микросхемы запускается на частоте HSI генератора. Выдача тактовых сигналов синхронизации для всех периферийных блоков кроме RST_CLK отключена. Для начала работы с нужным периферийным блоком необходимо включить его тактовую частоту в регистре PER_CLOCK. Некоторые контроллеры интерфейсов (UART, USB, Таймеры) могут работать на частотах отличных от частоты процессорного ядра, по этому в соответствующих регистрах (UART_CLOCK, USB_CLOCK, TIM_CLOCK) могут быть заданы их скорости работы. Для изменения тактовой частоты ядра можно перейти на другой генератор и/или воспользоваться блоком умножения тактовой частоты. Для корректной смены тактовой частоты сначала должны быть сформированы необходимые тактовые частоты и за тем осуществлено переключение на них на соответствующих мультиплексорах управляемых регистрами CPU_CLOCK и USB_CLOCK.

Для переключения с одного источника частоты на другой, нужно, чтобы оба источника оставались включенными.

При смене тактовой частоты процессорного ядра следует:

1. Настроить число тактов паузы для доступа к Flash-памяти, задаваемое полем Delay в регистре MDR_EEPROM->EEPROM_CMD.
2. Настроить режим работы встроенного регулятора напряжения DU_{CC}, задаваемый полем LOW регистра MDR_BKP->REG_0E совместно с дополнительной стабилизирующей нагрузкой для DU_{CC}, задаваемой полем SelectRI регистра MDR_BKP->REG_0E.

Если настраиваемая частота процессорного ядра выше текущей, то данные настройки нужно проводить до смены частоты, в противном случае – после её смены.

12.1 Встроенный RC-генератор HSI

Генератор HSI вырабатывает тактовую частоту f_{0_HSI} . Генератор автоматически запускается при появлении питания U_{CC} и при выходе в нормальный режим работы вырабатывает сигнал HSIRDY в регистре батарейного домена BKP_REG_0F. Первоначально процессорное ядро запускается на тактовой частоте HSI. При дальнейшей работе генератор HSI может быть отключен при помощи сигнала HSION в регистре BKP_REG_0F. Также генератор может быть подстроен при помощи сигнала HSITRIM в регистре BKP_REG_0F.

12.2 Встроенный RC-генератор LSI

Генератор LSI вырабатывает тактовую частоту f_{0_LSI} . Генератор автоматически запускается при появлении питания U_{CC} и при выходе в нормальный режим работы вырабатывает сигнал LSIRDY в регистре BKP_REG_0F. Первоначально тактовая частота генератор LSI используется для формирования дополнительной задержки t_{trig}. При дальнейшей работе генератор LSI может быть отключен при помощи сигнала LSION в регистре BKP_REG_0F.

12.3 Внешний осциллятор HSE

Осциллятор HSE предназначен для выработки тактовой частоты 2...16 МГц с помощью внешнего резонатора. Осциллятор запускается при появлении питания U_{CC} и сигнала разрешения HSEON в регистре HS_CONTROL. При выходе в нормальный режим работы вырабатывает сигнал HSERDY в регистре CLOCK_STATUS. Также осциллятор может работать в режиме HSEBYP, когда входная тактовая частота с входа OSC_IN проходит напрямую на выход HSE. Выход OSC_OUT находится в этом режиме в третьем состоянии.

12.4 Внешний осциллятор LSE

Осциллятор LSE предназначен для выработки тактовой частоты 32 КГц с помощью внешнего резонатора. Осциллятор запускается при появлении питания U_{CCBD} и сигнала разрешения LSEON в регистре BKP_REG_0F. При выходе в нормальный режим работы вырабатывает сигнал LSERDY в регистре BKP_REG_0F. Также осциллятор может работать в режиме LSEBYP, когда входная тактовая частота с входа OSC_IN32 проходит напрямую на выход LSE. Выход OSC_OUT32 находится в этом режиме третьем состоянии. Так как генератор LSE питается от напряжения питания U_{CCBD} и его

регистр управления BKP_REG_0F расположен в батарейном домене, то генератор может продолжать работать при пропадании основного питания U_{CC} . Генератор LSE используется для работы часов реального времени.

12.5 Встроенный блок умножения системной тактовой частоты

Блок умножения позволяет провести умножение входной тактовой частоты на коэффициент от 2 до 16, задаваемых на входе PLLCPUMUL[3:0] в регистре PLL_CONTROL. При выходе блока умножителя тактовой частоты в расчетный режим вырабатывается сигнал PLLCPURDY в регистре CLOCK_STATUS. Блок включается с помощью сигнала PLLCPUON в регистре PLL_CONTROL. Выходная частота используется как основная частота процессора и периферии.

12.6 Встроенный блок умножения системной тактовой частоты DSP

Блок умножения позволяет провести умножение входной тактовой частоты на коэффициент от 2 до 16, задаваемых на входе PLLDSPMUL[3:0] в регистре PLL_CONTROL. При выходе блока умножителя тактовой частоты в расчетный режим вырабатывается сигнал PLLDSPRDY в регистре CLOCK_STATUS. Блок включается с помощью сигнала PLLDSPON в регистре PLL_CONTROL. Выходная частота используется как основная частота подсистемы DSP.

12.7 Встроенный блок умножения тактовой частоты USB

Блок умножения позволяет провести умножение входной тактовой частоты на коэффициент от 2 до 16, задаваемых на входе PLLUSBMUL[3:0] в регистре PLL_CONTROL. Выходная частота блока умножителя должна составлять 48 МГц. При выходе блока умножителя тактовой частоты в расчетный режим вырабатывается сигнал PLLUSBRDY в регистре CLOCK_STATUS. Блок включается с помощью сигнала PLLUSBON в регистре PLL_CONTROL. Выходная частота используется как основная частота протокольной части USB интерфейса.

12.8 Описание регистров блока контроллера тактовой частоты

Таблица 12-1 – Описание регистров блока контроллера тактовой частоты

Базовый Адрес	Название	Описание
0x4002_0000	MDR_RST_CLK	Контроллер тактовой частоты
Смещение		
0x00	CLOCK_STATUS	MDR_RST_CLK->CLOCK_STATUS Регистр состояния блока управления тактовой частотой
0x04	PLL_CONTROL	MDR_RST_CLK->PLL_CONTROL Регистр управления блоками умножения частоты
0x08	HS_CONTROL	MDR_RST_CLK->HS_CONTROL Регистр управления высокочастотным генератором и осциллятором
0x0C	CPU_CLOCK	MDR_RST_CLK->CPU_CLOCK

Базовый Адрес	Название	Описание
		Регистр управления тактовой частотой процессорного ядра
0x10	USB_CLOCK	MDR_RST_CLK->USB_CLOCK Регистр управления тактовой частотой контроллера USB
0x14	ADC_MCO_CLOCK	MDR_RST_CLK->ADC_MCO_CLOCK Регистр управления тактовой частотой АЦП
0x18	RTC_CLOCK	MDR_RST_CLK->RTC_CLOCK Регистр управления формированием высокочастотных тактовых сигналов блока RTC
0x1C	PER_CLOCK	MDR_RST_CLK->PER_CLOCK Регистр управления тактовой частотой периферийных блоков
0x20	-	Зарезервировано
0x24	TIM_CLOCK	MDR_RST_CLK->TIM_CLOCK Регистр управления тактовой частотой TIMER
0x28	UART_CLOCK	MDR_RST_CLK->UART_CLOCK Регистр управления тактовой частотой UART
0x2C	SSP_CLOCK	MDR_RST_CLK->SSP_CLOCK Регистр управления тактовой частотой SSP
0x30	DSP_CLOCK	Регистр управления тактовой частотой контроллера DSP
0x34	SSP_CLOCK2	Регистр управления тактовой частотой SSP
0x38	DSP_CONTROL_STATUS	Регистр управления DSP См. Контроллер блока DSP

12.8.1 MDR_RST_CLK->CLOCK_STATUS

Таблица 12-2 – Регистр CLOCK_STATUS

Номер	31...4	3	2	1	0
Доступ	U	RO	RO	RO	RO
Сброс	0	0	0	0	0
	-	PLL DSP RDY	HSE RDY	PLL CPU RDY	PLL USB RDY

Таблица 12-3 – Описание бит регистра CLOCK_STATUS

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...4	-	Зарезервировано
3	PLL DSP RDY	Флаг выхода в рабочий режим DSP PLL: 0 – PLL не запущена или не стабильна; 1 – PLL запущена и стабильна
2	HSE RDY	Флаг выхода в рабочий режим осциллятора HSE: 0 – осциллятор не запущен или не стабилен; 1 – осциллятор запущен и стабилен

1	PLL CPU RDY	Флаг выхода в рабочий режим CPU PLL: 0 – PLL не запущена или не стабильна; 1 – PLL запущена и стабильна
0	PLL USB RDY	Флаг выхода в рабочий режим USB PLL: 0 – PLL не запущена или не стабильна; 1 – PLL запущена и стабильна

12.8.2 MDR_RST_CLK->PLL_CONTROL

Таблица 12-4 – Регистр PLL_CONTROL

Номер	31...24	23...20	19,18	17	16	15...12
Доступ	U	R/W	U	R/W	R/W	U
Сброс	0		0	0		
	-	PLL DSP MUL[3:0]	-	PLL DSP RLD	PLL DSP ON	-

Номер	11...8	7...4	3	2	1	0
Доступ	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0
	PLL CPU MUL[3:0]	PLL USB MUL[3:0]	PLL CPU RLD	PLL CPU ON	PLL USB RLD	PLL USB ON

Таблица 12-5 – Описание бит регистра PLL_CONTROL

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...24	-	Зарезервировано
23...20	PLL DSP MUL[3:0]	Коэффициент умножения для DSP PLL: $PLLDSP_0 = PLLDSP_i \cdot (PLLDSPMUL + 1)$
19...18	-	Зарезервировано
17	PLL DSP RLD	Бит перезапуска DSP PLL При смене коэффициента умножения в рабочем режиме необходимо задать равным 1, а после этого сбросить в 0
16	PLL DSP ON	Бит включения DSP PLL 0 – PLL выключена 1 – PLL включена
15...12	-	Зарезервировано
11...8	PLL CPU MUL[3:0]	Коэффициент умножения для CPU PLL: $PLLCPU_0 = PLLCPU_i \cdot (PLLCPUMUL + 1)$
7...4	PLL USB MUL[3:0]	Коэффициент умножения для USB PLL: $PLLUSB_0 = PLLUSB_i \cdot (PLLUSBMUL + 1)$

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
3	PLL CPU RLD	Бит перезапуска CPU PLL. При смене коэффициента умножения в рабочем режиме необходимо задать равным 1, а после этого сбросить в 0
2	PLL CPU ON	Бит включения CPU PLL: 0 – PLL выключена; 1 – PLL включена
1	PLL USB RLD	Бит перезапуска USB PLL. При смене коэффициента умножения в рабочем режиме необходимо задать равным 1, а после этого сбросить в 0
0	PLL USB ON	Бит включения USB PLL: 0 – PLL выключена; 1 – PLL включена

12.8.3 MDR_RST_CLK->HS_CONTROL

Таблица 12-6 – Регистр HS_CONTROL

Номер	31...2	1	0
Доступ	U	R/W	R/W
Сброс	0	0	0
	-	HSE BYP	HSE ON

Таблица 12-7 – Описание бит регистра HS_CONTROL

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...2	-	Зарезервировано
1	HSE BYP	Бит управления HSE осциллятором : 0 – режим осциллятора; 1 – режим внешнего генератора
0	HSE ON	Бит управления HSE осциллятором: 0 – выключен; 1 – включен

12.8.4 MDR_RST_CLK->CPU_CLOCK

Таблица 12-8 – Регистр CPU_CLOCK

Номер	31...10	9...8	7...4	3	2	1...0
Доступ	U	R/W	R/W	U	R/W	R/W
Сброс	0	0	0	0	0	0
	-	HCLK SEL[1:0]	CPU C3 SEL[3:0]	-	CPU C2 SEL	CPU C1 SEL[1:0]

Таблица 12-9 – Описание бит регистра CPU_CLOCK

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...10	-	Зарезервировано
9...8	HCLK SEL[1:0]	Биты выбора источника для HCLK: 00 – HSI; 01 – CPU_C3; 10 – LSE; 11 – LSI
7...4	CPU C3 SEL[3:0]	Биты выбора делителя для CPU_C3: 0xxx – CPU_C3 = CPU_C2; 1000 – CPU_C3 = CPU_C2 / 2; 1001 – CPU_C3 = CPU_C2 / 4; 1010 – CPU_C3 = CPU_C2 / 8; ... 1111 – CPU_C3 = CPU_C2 / 256
3	-	Зарезервировано
2	CPU C2 SEL	Биты выбора источника для CPU_C2: 0 – CPU_C1; 1 – PLLCPUo
1...0	CPU C1 SEL[1:0]	Биты выбора источника для CPU_C1: 00 – HSI; 01 – HSI/2; 10 – HSE; 11 – HSE/2

12.8.5 MDR_RST_CLK->USB_CLOCK

Таблица 12-10 – Регистр USB_CLOCK

Номер	31...9	8	7...5	4	3	2	1...0
Доступ	U	R/W	U	R/W	U	R/W	R/W
Сброс	0	0	0	0	0	0	0
	-	USB CLK EN	-	USB C3 SEL	-	USB C2 SEL	USB C1 SEL[1:0]

Таблица 12-11 – Описание бит регистра USB_CLOCK

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...9	-	Зарезервировано
8	USB CLK EN	Бит разрешения тактирования USB: 0 – нет тактовой частоты; 1 – есть тактовая частота

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
7...5	-	Зарезервировано
4	USB C3 SEL	Биты выбора делителя для USB_C3: 0 – USB_C3 = USB_C2; 1 – USB_C3 = USB_C2 / 2
3	-	Зарезервировано
2	USB C2 SEL	Биты выбора источника для USB_C2: 0 – USB_C1; 1 – PLLUSBo
1...0	USB C1 SEL[1:0]	Биты выбора источника для USB_C1: 00 – HSI; 01 – HSI/2; 10 – HSE; 11 – HSE/2

12.8.6 MDR_RST_CLK->ADC_MCO_CLOCK

Таблица 12-12 – Регистр ADC_MCO_CLOCK

Номер	31...14	13	12	11...8	7...6	5...4	3...2	1...0
Доступ	U	R/W	U	R/W	U	R/W	U	R/W
Сброс	0	0	0	0	0	0	0	0
	-	ADC CLK EN	-	ADC C3 SEL[3:0]	-	ADC C2 SEL[1:0]	-	ADC C1 SEL[1:0]

Таблица 12-13 – Описание бит регистра ADC_MCO_CLOCK

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31	AUC_CLK_EN	Бит разрешения выдачи тактовой частоты AUC CLK: 0 – запрещен; 1 – разрешен
30..28	-	Зарезервировано
27..24	AUC_SEL3	Биты выбора делителя для AUC_C3: 0xxx – AUC_C3 = AUC_C2; 1000 – AUC_C3 = AUC_C2 / 2; 1001 – AUC_C3 = AUC_C2 / 4; 1010 – AUC_C3 = AUC_C2 / 8; ... 1111 – AUC_C3 = AUC_C2 / 256
23..22	-	Зарезервировано
21..20	AUC_SEL2	Биты выбора источника для AUC_C2: 00 – AUC_C1; 01 – CPU_C2; 10 – USB_C2; 11 – DSP_C2
19..18	-	Зарезервировано

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
17..16	AUC_SEL1	Биты выбора источника для AUC_C1: 00 – HSI; 01 – HSI/2; 10 – HSE; 11 – HSE/2
15..14	-	Зарезервировано
13	ADC CLK EN	Бит разрешения выдачи тактовой частоты ADC CLK: 0 – запрещен; 1 – разрешен
12	-	Зарезервировано
11...8	ADC C3 SEL[3:0]	Биты выбора делителя для ADC_C3: 0xxx – ADC_C3 = ADC_C2; 1000 – ADC_C3 = ADC_C2 / 2; 1001 – ADC_C3 = ADC_C2 / 4; 1010 – ADC_C3 = ADC_C2 / 8; ... 1111 – ADC_C3 = ADC_C2 / 256
7...6	-	Зарезервировано
5...4	ADC C2 SEL[1:0]	Биты выбора источника для ADC_C1: 00 – LSE; 01 – LSI; 10 – ADC_C1; 11 – HSI_C1
3...2	-	Зарезервировано
1...0	ADC C1 SEL[1:0]	Биты выбора источника для ADC_C1: 00 – CPU_C1; 01 – USB_C1; 10 – CPU_C2; 11 – USB_C2

12.8.7 MDR_RST_CLK->RTC_CLOCK

Таблица 12-14 – Регистр RTC_CLOCK

Номер	31...10	9	8	7...4	3...0
Доступ	U	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0
	-	HSI RTC EN	HSE RTC EN	HSI C1 SEL[1:0]	HSE C1 SEL[1:0]

Таблица 12-15 – Описание бит регистра RTC_CLOCK

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...10	-	Зарезервировано
9	HSI RTC EN	Бит разрешения выдачи тактовой частоты HSI_RTC: 0 – запрещен; 1 – разрешен

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
8	HSE RTC EN	Бит разрешения выдачи тактовой частоты HSE_RTC: 0 – запрещен; 1 – разрешен
7...4	HSI C1 SEL[3:0]	Биты выбора делителя для HSI_C1: 0xxx – HSI_C1 = HSI; 1000 – HSI_C1 = HSI / 2; 1001 – HSI_C1 = HSI / 4; 1010 – HSI_C1 = HSI / 8; ... 1111 – HSI_C1 = HSI / 256
3...0	HSE C1 SEL[3:0]	Биты выбора делителя для HSE_C1: 0xxx – HSE_C1 = HSE; 1000 – HSE_C1 = HSE / 2; 1001 – HSE_C1 = HSE / 4; 1010 – HSE_C1 = HSE / 8; ... 1111 – HSE_C1 = HSE / 256

12.8.8 MDR_RST_CLK->PER_CLOCK

Таблица 12-16 – Регистр PER_CLOCK

Номер	31...0
Доступ	R/W
Сброс	0
	PCLK_EN[31:0]

Таблица 12-17 – Описание бит регистра PER_CLOCK

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...0	PCLK EN[31:0]	Биты разрешения тактирования периферийных блоков: 0 – запрещено; 1 – разрешено. PCLK[0] – SSP3; PCLK[1] – SSP4; PCLK[2] – USB; PCLK[3] – EEPROM_CNTRL; PCLK[4] – RST_CLK; PCLK[5] – DMA; PCLK[6] – UART1; PCLK[7] – UART2; PCLK[8] – SSP1; PCLK[9] – SDIO; PCLK[10] – I2C1; PCLK[11] – POWER; PCLK[12] – WWDG;

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
		PCLK[13] – IWDG; PCLK[14] – TIMER1; PCLK[15] – TIMER2; PCLK[16] – TIMER3; PCLK[17] – ADC; PCLK[18] – DAC; PCLK[19] – COMP; PCLK[20] – SSP2; PCLK[21] – PORTA; PCLK[22] – PORTB; PCLK[23] – PORTC; PCLK[24] – PORTD; PCLK[25] – PORTE; PCLK[26] – UART3; PCLK[27] – BKP; PCLK[28] – AUDIO CODEC; PCLK[29] – PORTF; PCLK[30] – EXT_BUS_CNTRL; PCLK[31] – CRYPTO

12.8.9 MDR_RST_CLK->TIM_CLOCK

Таблица 12-18 – Регистр TIM_CLOCK

Номер	31...27	26	25	24	23...16	15...8	7...0
Доступ	U	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0	0
	-	TIM3 CLK EN	TIM2 CLK EN	TIM1 CLK EN	TIM3 BRG [7:0]	TIM2 BRG [7:0]	TIM1 BRG [7:0]

Таблица 12-19 – Описание бит регистра TIM_CLOCK

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...27	-	Зарезервировано
26	TIM3 CLK EN	Разрешение тактовой частоты на TIM3: 0 – нет частоты; 1 – есть частота
25	TIM2 CLK EN	Разрешение тактовой частоты на TIM2: 0 – нет частоты; 1 – есть частота
24	TIM1 CLK EN	Разрешение тактовой частоты на TIM1: 0 – нет частоты; 1 – есть частота

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
23..16	TIM3 BRG [7:0]	Делитель тактовой частоты TIM3: xxxxxx000 – TIM3_CLK == HCLK; xxxxxx001 – TIM3_CLK == HCLK/2; xxxxxx010 – TIM3_CLK == HCLK/4; ... xxxxxx111 – TIM3_CLK == HCLK/128
15...8	TIM2 BRG [7:0]	Делитель тактовой частоты TIM2: xxxxxx000 – TIM2_CLK == HCLK; xxxxxx001 – TIM2_CLK == HCLK/2; xxxxxx010 – TIM2_CLK == HCLK/4; ... xxxxxx111 – TIM2_CLK == HCLK/128
7...0	TIM1 BRG [7:0]	Делитель тактовой частоты TIM1: xxxxxx000 – TIM1_CLK == HCLK; xxxxxx001 – TIM1_CLK == HCLK/2; xxxxxx010 – TIM1_CLK == HCLK/4; ... xxxxxx111 – TIM1_CLK == HCLK/128

12.8.10 MDR_RST_CLK->UART_CLOCK

Таблица 12-20 – Регистр UART_CLOCK

Номер	31...27	26	25	24	23...16	15...8	7...0
Доступ	U	R/W	R/W	R/W	U	R/W	R/W
Сброс	0	0	0	0	0	0	0
	-	UART3 CLK EN	UART2 CLK EN	UART1 CLK EN	UART3 BRG [7:0]	UART 2 BRG [7:0]	UART 1 BRG [7:0]

Таблица 12-21 – Описание бит регистра UART_CLOCK

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...27	-	Зарезервировано
26	UART3 CLK EN	Разрешение тактовой частоты на UART3: 0 – нет частоты; 1 – есть частота
25	UART2 CLK EN	Разрешение тактовой частоты на UART2: 0 – нет частоты; 1 – есть частота
24	UART1 CLK EN	Разрешение тактовой частоты на UART 1: 0 – нет частоты; 1 – есть частота
23...16	UART3 BRG	Делитель тактовой частоты UART 3: xxxxxx000 – UART 3_CLK == HCLK;

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
	[7:0]	xxxxx001 – UART 3_CLK == HCLK/2; xxxxx010 – UART 3_CLK == HCLK/4; ... xxxxx111 – UART 3_CLK == HCLK/128
15...8	UART2 BRG [7:0]	Делитель тактовой частоты UART 2: xxxxx000 – UART 2_CLK == HCLK; xxxxx001 – UART 2_CLK == HCLK/2; xxxxx010 – UART 2_CLK == HCLK/4; ... xxxxx111 – UART 2_CLK == HCLK/128
7...0	UART1 BRG [7:0]	Делитель тактовой частоты UART1: xxxxx000 – UART 1_CLK == HCLK; xxxxx001 – UART 1_CLK == HCLK/2; xxxxx010 – UART 1_CLK == HCLK/4; ... xxxxx111 – UART 1_CLK == HCLK/128

12.8.11 MDR_RST_CLK->SSP_CLOCK

Таблица 12-22 – Регистр SSP_CLOCK

Номер	31...27	26	25	24	23...16	15...8	7...0
Доступ	U	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0	0
	-	SSP3 CLK EN	SSP2 CLK EN	SSP 1 CLK EN	SSP 3 BRG [7:0]	SSP 2 BRG [7:0]	SSP 1 BRG [7:0]

Таблица 12-23 – Описание бит регистра SSP_CLOCK

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...27	-	Зарезервировано
26	SSP3 CLK EN	Разрешение тактовой частоты на SSP 3: 0 – нет частоты; 1 – есть частота
25	SSP2 CLK EN	Разрешение тактовой частоты на SSP 2: 0 – нет частоты; 1 – есть частота
24	SSP1 CLK EN	Разрешение тактовой частоты на SSP 1: 0 – нет частоты; 1 – есть частота
23...16	SSP3 BRG [7:0]	Делитель тактовой частоты SSP 3: xxxxx000 – SSP 3_CLK == HCLK; xxxxx001 – SSP 3_CLK == HCLK/2; xxxxx010 – SSP 3_CLK == HCLK/4; ... xxxxx111 – SSP 3_CLK == HCLK/128
15...8	SSP2	Делитель тактовой частоты SSP 2:

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
	BRG [7:0]	xxxxx000 – SSP 2_CLK == HCLK; xxxxx001 – SSP 2_CLK == HCLK/2; xxxxx010 – SSP 2_CLK == HCLK/4; ... xxxxx111 – SSP 2_CLK == HCLK/128
7...0	SSP1 BRG [7:0]	Делитель тактовой частоты SSP 1: xxxxx000 – SSP 1_CLK == HCLK; xxxxx001 – SSP 1_CLK == HCLK/2; xxxxx010 – SSP 1_CLK == HCLK/4; ... xxxxx111 – SSP 1_CLK == HCLK/128

12.8.12 MDR_RST_CLK->DSP_CLOCK

Таблица 12-24 – Регистр DSP_CLOCK

Номер	31...9	8	7...5	4	3	2	1...0
Доступ	U	R/W	U	R/W	U	R/W	R/W
Сброс	0		0	0	0	0	0
	-	DSP CLK EN	-	DSP C3 SEL	-	DSP C2 SEL	DSP C1 SEL

Таблица 12-25 – Описание бит регистра DSP_CLOCK

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...26	-	Зарезервировано
8	DSP CLK EN	Бит разрешения тактирования DSP: 0 – нет тактовой частоты; 1 – есть тактовая частота
7...5	-	Зарезервировано
4	DSP C3 SEL	Биты выбора делителя для DSP_C3: $DSP\_C3 = DSP\_C2 / (DSP\_C3\_SEL + 1)$
3	-	Зарезервировано
2	DSP C2 SEL	Биты выбора источника для DSP_C2: 0 – DSP_C1; 1 – PLLDSPo
1...0	DSP C1 SEL	Биты выбора источника для DSP_C1: 00 – HSI; 01 – HSI/2; 10 – HSE; 11 – HSE/2

12.8.13 MDR_RST_CLK->SSP_CLOCK2

Таблица 12-26 – Регистр SSP_CLOCK2

Номер	31...25	24	25...8	7...0
Доступ	U	R/W	U	R/W
Сброс	0	0	0	0
	-	SSP4 CLK EN	-	SSP 4 BRG [7:0]

Таблица 12-27 – Описание бит регистра SSP_CLOCK2

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...25	-	Зарезервировано
24	SSP4 CLK EN	Разрешение тактовой частоты на SSP 4: 0 – нет частоты; 1 – есть частота
23...8	-	Зарезервировано
7...0	SSP4 BRG [7:0]	Делитель тактовой частоты SSP 4: xxxxxx000 – SSP 4_CLK == HCLK; xxxxxx001 – SSP 4_CLK == HCLK/2; xxxxxx010 – SSP 4_CLK == HCLK/4; ... xxxxxx111 – SSP 4_CLK == HCLK/128

Часы реального времени позволяют организовать механизм отсчета времени в кристалле, в том числе при отключении основного источника питания. Включение часов реального времени осуществляется битом RTC_EN. В качестве источника тактовой частоты часов реального времени RTC_CLK может выступать генератор LSI или LSE, а также генератор HSI (частота HSI_RTC) или HSE (частота HSE_RTC) с дополнительным делителем до 256. Частоты HSI_RTC и HSE_RTC формируются в блоке управления тактовыми частотами MDR_RST_CLK и могут быть выбраны только при наличии питания U_{CCD}. Частота LSI может быть выбрана при наличии питания U_{CC}, LSE – при наличии U_{CC} или U_{CCB}. Выбор между источниками осуществляется битами RTC_SEL. При возможном отключении основного источника питания U_{CC} в качестве источника тактовой частоты RTC_CLK должен использоваться осциллятор LSE, так как он также имеет питание U_{CCBD}. Биты управления осциллятором LSE расположены в батарейном домене и таким образом при отключении основного питания они не сбрасываются.

Для отсчета секунд в часах реального времени применяется 20-битный предварительный делитель входной тактовой частоты RTC_CLK, на выходе которого формируется тактовый сигнал RTC_CLKd. Регистр RTC_DIV выступает в качестве счетчика предварительного делителя, который тактируется на частоте RTC_CLK. Регистр RTC_PRL задает коэффициент деления предварительного делителя, при этом счетчик RTC_DIV инкрементируется в интервале от 0 до RTC_PRL. Коэффициент деления в регистре RTC_PRL должен быть задан таким образом, чтобы частота RTC_CLKd составляла 1 Гц.

Для калибровки тактовой частоты RTC_CLK используются биты RTC_CAL[7:0]. Значение RTC_CAL[7:0] определяет, какое число тактов RTC_CLK из 2^{20} будет замаскировано с помощью дополнительного счетчика RTC_20. Таким образом, с помощью бит RTC_CAL[7:0] производится замедление хода часов. Для ускорения хода часов необходимо задать меньшее, чем требуется, значение регистра RTC_PRL, а затем произвести замедление с помощью бит RTC_CAL[7:0]. Изменение значения бит RTC_CAL[7:0] может быть осуществлено в ходе работы часов реального времени.

Регистр RTC_CNT предназначен для отсчета времени в секундах и работает на выходной частоте предварительного делителя RTC_CLKd. Регистр RTC_ALRM предназначен для задания времени, при превышении которого вырабатывается флаг прерывания и пробуждения процессора ALRF. Таким образом, бит STANDBY, отключающий внутренний регулятор напряжения, автоматически сбрасывается при превышении RTC_CNT значения RTC_ALRM.

Бит STANDBY также может быть сброшен с помощью вывода WAKEUP.

13.2 Регистры аварийного сохранения

Батарейный домен имеет 16 встроенных 32-разрядных регистров аварийного сохранения. 16-й и 15-й регистры служат для хранения бит управления батарейным доменом, оставшиеся 14 регистров могут быть использованы разработчиком программы.

13.3 Описание регистров блока батарейного домена

Таблица 13-1 – Описание регистров блока батарейного домена

Базовый Адрес	Название	Описание
0x400D_8000	MDR_BKP	Контроллер батарейного домена и часов реального времени
Смещение		
0x00	REG_[00...0D]	Регистр MDR_BKP->REG_[00...0D] аварийного сохранения 0
...		
0x38	REG_0E	Регистр MDR_BKP->REG_0E аварийного сохранения 14
0x3C	REG_0F	Регистр MDR_BKP->REG_0F аварийного сохранения 15 и управления блоками RTC, LSE, LSI и HSI
0x40	RTC_CNT	Регистр MDR_BKP->RTC_CNT основного счетчика часов реального времени
0x44	RTC_DIV	Регистр MDR_BKP->RTC_DIV счетчика предварительного делителя

Базовый Адрес	Название	Описание
0x48	RTC_PRL	Регистр MDR_BKP->RTC_PRL коэффициента деления предварительного делителя
0x4C	RTC_ALRM	Регистр MDR_BKP->RTC_ALRM значения для сравнения с основным счетчиком RTC_CNT и выработки сигнала ALRF
0x50	RTC_CS	Регистр MDR_BKP->RTC_CS управления и состояния флагов часов реального времени

13.3.1 MDR_BKP->REG_[00...0D]

Здесь приведено описание следующих регистров:

MDR_BKP->REG_00;
MDR_BKP->REG_01;
MDR_BKP->REG_02;
MDR_BKP->REG_03;
MDR_BKP->REG_04;
MDR_BKP->REG_05;
MDR_BKP->REG_06;
MDR_BKP->REG_07;
MDR_BKP->REG_08;
MDR_BKP->REG_09;
MDR_BKP->REG_0A;
MDR_BKP->REG_0B;
MDR_BKP->REG_0C;
MDR_BKP->REG_0D.

Таблица 13-2 – Регистры REG_[0D...00]

Номер	31...0
Доступ	R/W
Сброс	U
	BKP REG[31:0]

Таблица 13-3 – Описание битов регистров REG_[0D...00]

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...0	BKP REG[31:0]	Регистр аварийного сохранения

13.3.2 MDR_BKP->REG_0E

Таблица 13-4 – Регистр REG_0E

Номер	31...15	14...12	11	10...8	7	6	5...3	2...0
Доступ	U	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	0	U	1	0	1	0	0	0
	-	MODE [2:0]	FPOR	Trim [2:0]	JTAG_B	JTAG_A	SelectRI [2:0]	LOW [2:0]

Таблица 13-5 – Описание бит регистра REG_0E

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...15	-	Зарезервировано
14...12	MODE[2:0]	Режим работы микросхемы, определенный при включении питания по выводам MODE[2:0] (PF[6:4]): 000 – микроконтроллер, с отладкой через JTAG_B; 001 – микроконтроллер, с отладкой через JTAG_A; 010 – микропроцессор, с отладкой через JTAG_B; 011 – микропроцессор, с отладкой через JTAG_B; 100 – зарезервировано; 101 – зарезервировано; 110 – UART-загрузчик без отладки; 111 – зарезервировано (режим тестирования кристалла). Подробнее см. в разделе 6.5 «Загрузочное ПЗУ и режимы работы микросхемы». При смене режима работы в данном регистре, изменение вступит в силу после сброса процессора через RESET, при сбросе по просадке питания режим будет определяться выводами MODE[2:0]
11	FPOR	Флаг срабатывания POR. Устанавливается в 1 загрузочным ПЗУ при первоначальном включении по появлению питания U_{CC}, при сбросе по питанию устанавливается в 0. Служит для анализа загрузочным ПЗУ, что сейчас идет выполнение программы после системного или программного сброса, либо после сброса по питанию
10...8	Trim[2:0]	Коэффициент настройки опорного напряжения встроенного регулятора напряжения U_{CCD}. С помощью Trim осуществляется подстройка напряжения U_{CCD}: 000 – $U_{CCD} + 0,10 \text{ В}$ – значение по умолчанию; 001 – $U_{CCD} + 0,06 \text{ В}$; 010 – $U_{CCD} + 0,04 \text{ В}$; 011 – $U_{CCD} + 0,01 \text{ В}$; 100 – $U_{CCD} - 0,01 \text{ В}$; 101 – $U_{CCD} - 0,04 \text{ В}$; 110 – $U_{CCD} - 0,06 \text{ В}$; 111 – $U_{CCD} - 0,10 \text{ В}$
7	JTAG B	Разрешение работы порта JTAG B: 0 – запрещен; 1 – разрешен. При одновременной установке JTAG B и JTAG A выбирается режим отбраковочного тестирования микросхемы, при этом микросхема перестает работать в функциональном режиме
6	JTAG A	Разрешение работы порта JTAG A: 0 – запрещен; 1 – разрешен

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
5...3	SelectRI[2:0]	Выбор дополнительной стабилизирующей нагрузки для встроенного регулятора напряжения U_{CCD} : 000 – ~ 6 кОм (дополнительный ток потребления 300 мкА); 001 – ~ 270 кОм (дополнительный ток потребления 6,6 мкА); 010 – ~ 90 кОм (дополнительный ток потребления 20 мкА); 011 – ~ 24 кОм (дополнительный ток потребления 80 мкА); 100 – ~ 900 кОм (собственное потребление 2 мкА); 101 – ~ 2 кОм (дополнительный ток потребления 900 мкА); 110 – ~ 400 Ом (дополнительный ток потребления 4,4 мА); 111 – ~ 100 Ом (дополнительный ток потребления 19 мА)
2...0	LOW[2:0]	Выбор режима работы встроенного регулятора напряжения U_{CCD} . Значение LOW должно совпадать со значением SelectRI и выставляться в зависимости от тактовой частоты микросхемы: 000 – частота до 10 МГц; 001 – частота до 200 КГц; 010 – частота до 500 КГц; 011 – частота до 1 МГц; 100 – при выключении всех генераторов; 101 – частота до 40 МГц; 110 – частота до 80 МГц; 111 – частота более 80 МГц

13.3.3 MDR_BKP->REG_0F

Таблица 13-6 – Регистр REG_0F

Номер	15	14	13	12...5	4	3, 2	1	0
Доступ	R/W	U	RO	R/W	R/W	R/W	R/W	R/W
Сброс	1	0	0	0	0	0	0	0
	LSI ON	-	LSE RDY	RTC CAL[7:0]	RTC EN	RTC SEL[1:0]	LSE BYP	LSE ON

Номер	31	30	29...24	23	22	21	20...16
Доступ	R/W	R/W	R/W	RO	R/W	RO	R/W
Сброс	0	0	100000	1	1	1	10000
	RTC RESET	STANDBY	HSI TRIM [5:0]	HSI RDY	HSI ON	LSI RDY	LSI TRIM [4:0]

Таблица 13-7 – Описание битов регистра REG_0F

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31	RTC RESET	Сброс часов реального времени: 0 – часы не сбрасываются; 1 – часы сбрасываются

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
30	STANDBY	Режим отключения регулятора U_{CCD} : 0 – регулятор включен и выдает напряжение; 1 – регулятор выключен. Триггер сбрасывается флагом ALRF или по низкому уровню на выводе WAKEUP
29...24	HSI TRIM[5:0]	Коэффициент подстройки частоты генератора HSI. Смотрите диаграмму зависимости (Рисунок 13-3)
23	HSI RDY	Флаг выхода генератора HSI в рабочий режим: 0 – генератор не запущен или не вышел в режим; 1 – генератор работает в рабочем режиме
22	HSI ON	Бит управления генератором HSI: 0 – генератор выключен; 1 – генератор включен
21	LSI RDY	Флаг выхода генератора LSI в рабочий режим: 0 – генератор не запущен или не вышел в режим; 1 – генератор находится в рабочем режиме
20...16	LSI TRIM[4:0]	Коэффициент подстройки частоты генератора LSI. Смотрите диаграмму зависимости (Рисунок 13-2)
15	LSI ON	Бит управления генератором LSI: 0 – генератор выключен; 1 – генератор включен
14	-	Зарезервировано
13	LSE RDY	Флаг выхода генератора LSE в рабочий режим: 0 – генератор не запущен или не вышел в режим; 1 – генератор работает в рабочем режиме
12...5	RTC CAL[7:0]	Коэффициент подстройки тактовой частоты часов реального времени RTC_CLK. Из каждых 2^{20} тактов RTC_CLK будет замаскировано RTC_CAL тактов: 00000000 – 0 тактов 00000001 – 1 такт 11111111 – 255 тактов Таким образом, если исходная частота равна 32768,00000 Гц, то при RTC_CAL = 0 частота RTC_CLK = 32768,00000 Гц; при RTC_CAL = 1 частота RTC_CLK = 32767,96875 Гц; ... при RTC_CAL = 255 частота RTC_CLK = 32760,03125 Гц В общем случае тактовая частота RTC_CLK после подстройки равна: $RTC\_CLK = CLK - RTC\_CAL \cdot CLK/2^{20}$, где CLK – исходная тактовая частота часов реального времени в Гц.
4	RTC EN	Бит разрешения работы часов реального времени: 0 – работа запрещена; 1 – работа разрешена

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
3, 2	RTC SEL[1:0]	Биты выбора источника тактовой синхронизации часов реального времени RTC_CLK: 00 – LSI; 01 – LSE; 10 – HSI_RTC (формируется в блоке MDR_RST_CLK); 11 – HSE_RTC (формируется в блоке MDR_RST_CLK)
1	LSE BYP	Бит управления генератором LSE: 0 – режим осциллятора; 1 – режим работы на проход (внешний генератор)
0	LSE ON	Флаг выхода генератора LSI в рабочий режим: 0 – генератор не запущен или не вышел в режим; 1 – генератор работает в рабочем режиме

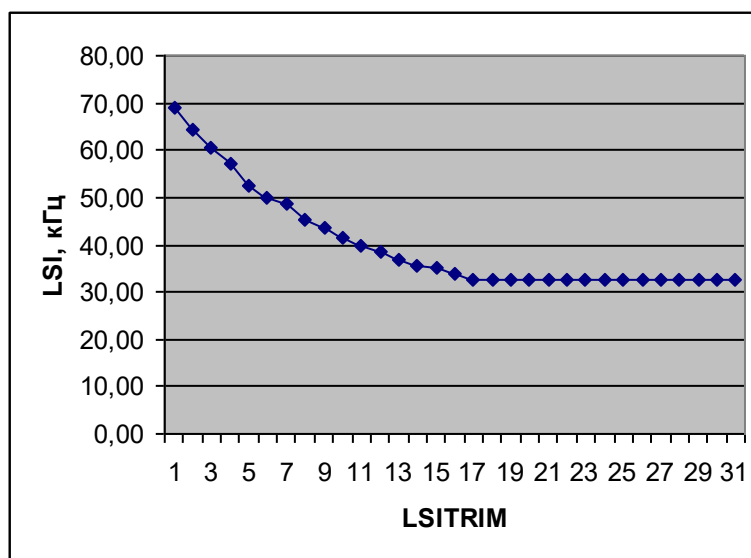


Рисунок 13-2 – Зависимость частоты LSI от значения LSITRIM

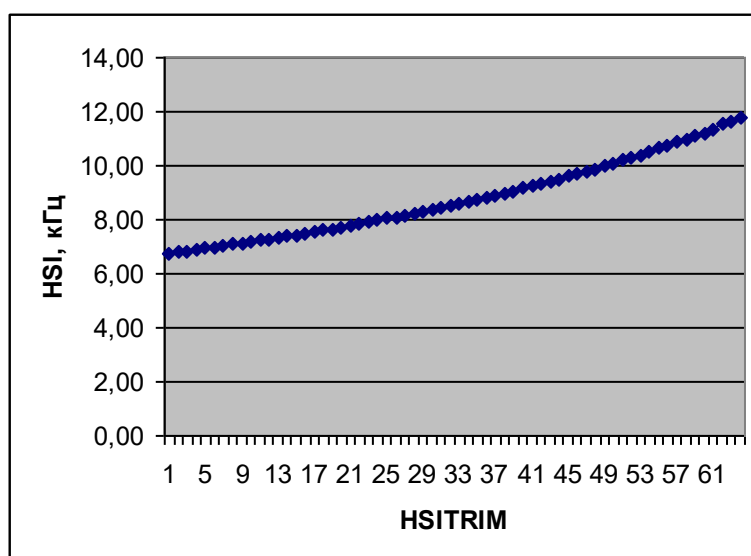


Рисунок 13-3 – Зависимость частоты HSI от значения HSITRIM

13.3.4 MDR_BKP->RTC_CNT

Таблица 13-8 – Регистр RTC_CNT

Номер	31...0
Доступ	R/W
Сброс	0
	RTC CNT[31:0]

Таблица 13-9 – Описание битов регистра RTC_CNT

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...0	RTC CNT[31:0]	Значение основного счетчика часов реального времени

13.3.5 MDR_BKP->RTC_DIV

Таблица 13-10 – Регистр RTC_DIV

Номер	31...20	19...0
Доступ	U	R/W
Сброс	0	0
	-	RTC DIV[19:0]

Таблица 13-11 – Описание битов регистра RTC_DIV

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...20	-	Зарезервировано
19...0	RTC DIV[19:0]	Значение счетчика предварительного делителя часов реального времени

13.3.6 MDR_BKP->RTC_PRL

Таблица 13-12 – Регистр RTC_PRL

Номер	31...20	19...0
Доступ	U	R/W
Сброс	0	0
	-	RTC PRL[19:0]

Таблица 13-13 – Описание битов регистра RTC_PRL

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...20	-	Зарезервировано
19...0	RTC PRL[19:0]	Коэффициент деления тактовой частоты RTC_CLK: $RTC\_CLKd = RTC\_CLK / (RTC\_PRL + 1)$

13.3.7 MDR_BKP->RTC_ALARM

Таблица 13-14 – Регистр RTC_ALARM

Номер	31...0
Доступ	R/W
Сброс	0
	RTC ALRM[31:0]

Таблица 13-15 – Описание битов регистра RTC_ALARM

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...0	RTC ALRM[31:0]	Значение для сравнения с основным счетчиком RTC_CNT. Сигнал ALRF вырабатывается в момент превышения основным счетчиком RTC_CNT значения RTC_ALARM (RTC_CNT == (RTC_ALARM+1))

13.3.8 MDR_BKP->RTC_CS

Таблица 13-16 – Регистр RTC_CS

Номер	31...7	6	5	4	3	2	1	0
Доступ	U	RO	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0	0	0
	-	WEC	ALRF_IE	SECF_IE	OWF_IE	ALRF	SECF	OWF

Таблица 13-17 – Описание битов регистра RTC_CS

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
30...7	-	Зарезервировано
6	WEC	Флаг выполнения записи в регистры RTC: 0 – можно записывать в регистры RTC; 1 – запись в регистры запрещена, идет запись в регистры RTC
5	ALRF_IE	Флаг разрешения прерывания по событию ALRF: 0 – прерывание запрещено; 1 – прерывание разрешено
4	SECF_IE	Флаг разрешения прерывания по событию SECF: 0 – прерывание запрещено; 1 – прерывание разрешено
3	OWF_IE	Флаг разрешения прерывания по событию OWF: 0 – прерывание запрещено; 1 – прерывание разрешено
2	ALRF	Флаг превышения основным счетчиком RTC_CNT значения RTC_ALARM: 0 – нет события превышения; 1 – было событие превышения. Флаг устанавливается при RTC_CNT == (RTC_ALARM+1), сброс флага осуществляется записью 1

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
1	SECF	Флаг изменения значения основного счетчика RTC_CNT (инкремент счетчика RTC_CNT выполняется один раз в секунду): 0 – нет изменения RTC_CNT; 1 – есть изменение RTC_CNT. Сброс флага осуществляется записью 1
0	OWF	Флаг переполнения основного счетчика RTC_CNT: 0 – нет переполнения; 1 – было переполнение Сброс флага осуществляется записью 1

14 Порты ввода-вывода

Микросхема имеет шесть портов ввода/вывода. Порты 16-ти разрядные и их выводы мультиплексируются между различными функциональными блоками, управление для каждого вывода отдельное. Для того, что бы выводы порта перешли под управление того или иного периферийного блока необходимо задать для нужных выводов выполняемую функцию и настройки.

При работе в режиме отладки не допускается изменение функций выводов, совмещенных с выводами JTAG, путем записи «1» в соответствующие биты регистров RXTX, SET и OE. Это может привести к блокировке интерфейса отладки.

Таблица 14-1 – Порты ввода-вывода

Вывод	Аналоговая функция ANALOG_EN=0	Цифровая функция								
		Порт IO MODE[1:0]=00 ANALOG_EN=1	Основная MODE[1:0]=01 ANALOG_EN=1	Альтернативная MODE[1:0]=10 ANALOG_EN=1		Переопределенная MODE[1:0]=11 ANALOG_EN=1				
Порт A										
PA0	-		PA0	DATA0	1	SDIO_CLK	2	TMR1_CH1	3	
PA1	-		PA1	DATA1		SDIO_CMD		TMR1_CH1n		
PA2	-		PA2	DATA2		SDIO_DATA0		TMR1_CH2		
PA3	-		PA3	DATA3		SDIO_DATA1		TMR1_CH2n		
PA4	-		PA4	DATA4		SDIO_DATA2		TMR1_CH3		
PA5	-		PA5	DATA5		SDIO_DATA3		TMR1_CH3n		
PA6	-		PA6	DATA6		UART1_RXD		10		TMR1_CH4
PA7	-		PA7	DATA7	UART1_TXD	TMR1_CH4n				
PA8	-		PA8	DATA8	SSP2_CLK	13	BSP1_RTCLK	18		
PA9	-		PA9	DATA9	SSP2_FSS		BSP1_RTFR			
PA10	-		PA10	DATA10	SSP2_TXD		BSP1_TX			
PA11	-		PA11	DATA11	SSP2_RXD		BSP1_RX			
PA12	-		PA12	DATA12	SSP1_RXD	16	TMR1_ETR	3		
PA13	-		PA13	DATA13	SSP1_TXD		TMR1_BLK			
PA14	-		PA14	DATA14	SSP1_CLK		TMR2_CH1	15		
PA15	-		PA15	DATA15	SSP1_FSS		EXT_INT1/XF	9		
Порт B										
PB0	-		PB0	JA_TDO	DATA16	1	UART2_TXD	141	TMR2_CH1n	15
PB1	-		PB1	JA_TMS	DATA17		TMR1_ETR	3	TMR2_CH2	
PB2	-		PB2	JA_TCK	DATA18		TMR1_BLK		TMR2_CH2n	
PB3	-		PB3	JA_TDI	DATA19		UART2_RXD	14	TMR2_CH3	
PB4	-		PB4	JA_TRST	DATA20	TMR1_CH1	3	TMR2_CH3n		
PB5	-		PB5		DATA21	SSP3_TXD	4	BSP1_TX	18	
PB6	-		PB6		DATA22	SSP3_FSS		BSP1_TFR		
PB7	-		PB7		DATA23	SSP3_RXD		BSP1_RX		
PB8	-		PB8		DATA24	SSP3_CLK		BSP1_TCLK		
PB9	-		PB9		DATA25	TMR1_CH1n	3	TMR2_ETR	15	
PB10	-		PB10		DATA26	TMR1_CH2		TMR2_BLK		
PB11	-		PB11		DATA27	TMR1_CH2n		EXT_INT2		9
PB12	-		PB12		DATA28	SSP2_CLK	13	BSP1_RCLK	18	
PB13	-		PB13		DATA29	SSP2_FSS		BSP1_RFR		
PB14	-		PB14		DATA30	SSP2_TXD		BSP1_TX		
PB15	-		PB15		DATA31	SSP2_RXD		BSP1_RX		

Вывод	Аналоговая функция ANALOG_EN=0	Цифровая функция								
		Порт IO MODE[1:0]=00 ANALOG_EN=1	Основная MODE[1:0]=01 ANALOG_EN=1	Альтернативная MODE[1:0]=10 ANALOG_EN=1		Переопределенная MODE[1:0]=11 ANALOG_EN=1				
Порт C										
PC0	-		PC0	COMP_OUT	7	SSP4_FSS	17	SDIO_CLK	2	
PC1	-		PC1	OE	1	UART1_TXD	10	SDIO_CMD		
PC2	-		PC2	WE		UART1_RXD	17	SDIO_DATA0		
PC3	-		PC3	BE0		SSP4_RXD		SDIO_DATA1		
PC4	-		PC4	BE1			SSP4_TXD	3	SDIO_DATA2	14
PC5	-		PC5	BE2	SSP4_CLK		SDIO_DATA3			
PC6	-		PC6	BE3	TMR1_CH3		UART2_TXD		4	UART2_RXD
PC7	-		PC7	CLOCK	TMR1_CH3n	4	BSP2_TX	19		
PC8	-		PC8	SDIO_CLK	2		SSP3_TXD		17	BSP2_RTFR
PC9	-		PC9	SDIO_CMD			SSP3_FSS			BSP2_RX
PC10	-		PC10	SDIO_DATA0		SSP3_RXD	BSP2_RTCLK			
PC11	-		PC11	SDIO_DATA1		SSP3_CLK	11	BSP3_RTFR	20	
PC12	-		PC12	SDIO_DATA2	SSP4_FSS	BSP3_RTCLK				
PC13	-		PC13	SDIO_DATA3	SSP4_CLK	BSP3_TX				
PC14	-		PC14	I2C1_SCK	11	SSP4_TXD		BSP3_RX		
PC15	-		PC15	I2C_SDA		SSP4_RXD				
Порт D										
PD0	ADC0_REF+	5	PD0	JB_TMS	3	I2C1_SCK	11	BSP3_RX	20	
PD1	ADC1_REF-		PD1	JB_TCK		I2C1_SDA		BSP3_TX		
PD2	ADC2		PD2	BUSY1	1	SSP1_TXD	16	BSP2_TX	19	
PD3	ADC3		PD3	JB_TDI	3	SSP1_FSS		BSP2_TFR		
PD4	ADC4		PD4	JB_TDO		SSP1_RXD		BSP2_RX		
PD5	ADC5		PD5	CLE	1	SSP1_CLK	3	BSP2_TCKL	20	
PD6	ADC6		PD6	ALE	3	TMR1_CH4		BSP3_TCLK		
PD7	ADC7		PD7			UART1_SIROUT		BSP3_TFR		
PD8	ADC8		PD8			UART1_SIRIN	15	BSP3_TX		
PD9	ADC9		PD9		TMR1_CH3	BSP3_RFR				
PD10	ADC10		PD10		TMR2_BLK	BSP3_RX				
PD11	ADC11		PD11		TMR2_ETR	13	BSP3_RCLK	19		
PD12	ADC12		PD12		SSP2_FSS		BSP2_RFR			
PD13	ADC13		PD13	UART3_TXD	SSP2_RXD		BSP2_RX			
PD14	ADC14		PD14	UART3_RXD	SSP2_CLK	BSP2_RCKL				
PD15	ADC15		PD15	EXT_INT1	9	SSP2_TXD	BSP2_TX			
Порт E										
PE0	DAC2_OUT	6	PE0	ADDR16	1	SSP4_FSS	17	BSP1_RTFR	18	
PE1	DAC2_REF		PE1	ADDR17		SSP4_CLK		BSP1_RTCLK		
PE2	COMP_IN1		PE2	ADDR18		SSP4_TXD		BSP1_TX		
PE3	COMP_IN2	7	PE3	ADDR19		SSP4_RXD	13	BSP1_RX	10	
PE4	COMP_REF+		PE4	ADDR20		SSP2_TXD		UART1_TXD		
PE5	COMP_REF-		PE5	ADDR21		SSP2_FSS		UART1_RXD		
PE6	OSC_IN32	8	PE6	ADDR22		SSP2_RXD	15	EXT_INT3	9	
PE7	OSC_OUT32		PE7	ADDR23		SSP2_CLK		TMR2_CH4	15	
PE8	COMP_IN3	7	PE8	ADDR24		TMR2_CH4	14	TMR2_CH4n		12
PE9	DAC1_OUT	6	PE9	ADDR25		TMR2_CH4n		TMR3_CH1		
PE10	DAC1_REF		PE10	ADDR26		UART2_TXD	16	TMR3_ETR	19	
PE11	-	PE11	ADDR27	UART2_RXD		TMR3_BLK				
PE12	-	PE12	ADDR28		SSP1_RXD	BSP2_RTFR				
PE13	-	PE13	ADDR29		SSP1_TXD	BSP2_RTCLK				
PE14	-	PE14	ADDR30		SSP1_CLK	BSP2_TX				
PE15	-	PE15	ADDR31		SSP1_FSS	BSP2_RX				

Вывод	Аналоговая функция ANALOG_EN=0	Цифровая функция							
		Порт IO MODE[1:0]=00 ANALOG_EN=1	Основная MODE[1:0]=01 ANALOG_EN=1	Альтернативная MODE[1:0]=10 ANALOG_EN=1		Переопределенная MODE[1:0]=11 ANALOG_EN=1			
Порт F									
PF0	-		PF0	ADDR0	1	UART3_RXD	18	TMR3_CH1n	12
PF1	-		PF1	ADDR1		UART3_TXD		TMR3_CH2	
PF2	-		PF2	ADDR2	SSP4_RXD	17	BSP3_RX	20	
PF3	-		PF3	ADDR3	SSP4_TXD		BSP3_TX		
PF4	-		PF4	ADDR4	SSP4_CLK		BSP3_TCKL		
PF5	-		PF5	ADDR5	SSP4_FSS	BSP3_TFR			
PF6	-		PF6	ADDR6	TMR2_CH3n	15	TMR3_CH2n	12	
PF7	-		PF7	ADDR7	TMR2_CH3		TMR3_CH3		
PF8	-		PF8	ADDR8	TMR2_CH2n		TMR3_CH3n		
PF9	-		PF9	ADDR9	TMR2_CH2		TMR3_CH4		
PF10	-		PF10	ADDR10	TMR2_CH1n		TMR3_CH4n		
PF11	-		PF11	ADDR11	TMR2_CH1		EXT_INT4		9
PF12	-		PF12	ADDR12	SSP3_TXD		4		BSP3_TX
PF13	-		PF13	ADDR13	SSP3_FSS	BSP3_RFR			
PF14	-		PF14	ADDR14	SSP3_RXD	BSP3_RX			
PF15	-	PF15	ADDR15	SSP3_CLK	BSP3_RCKL				

Примечания

- 1) Выводы управляются системной шиной EXT_BUS.
- 2) Выводы управляются контроллером интерфейса SDIO.
- 3) Выводы управляются Таймером 1.
- 4) Выводы управляются контроллером интерфейса SSP3.
- 5) Выводы используются АЦП.
- 6) Выводы используются ЦАП.
- 7) Выводы используются Компаратором.
- 8) Выводы используются генератором LSE.
- 9) Выводы используются контроллером прерываний.
- 10) Выводы управляются контроллером интерфейса UART1.
- 11) Выводы управляются контроллером интерфейса I2C.
- 12) Выводы управляются Таймером 3.
- 13) Выводы управляются контроллером интерфейса SSP2.
- 14) Выводы управляются контроллером интерфейса UART2.
- 15) Выводы управляются Таймером 2.
- 16) Выводы управляются контроллером интерфейса SSP1.
- 17) Выводы управляются контроллером интерфейса SSP4.
- 18) Выводы управляются контроллером интерфейса McBSP1.
- 19) Выводы управляются контроллером интерфейса McBSP2.
- 20) Выводы управляются контроллером интерфейса McBSP3.

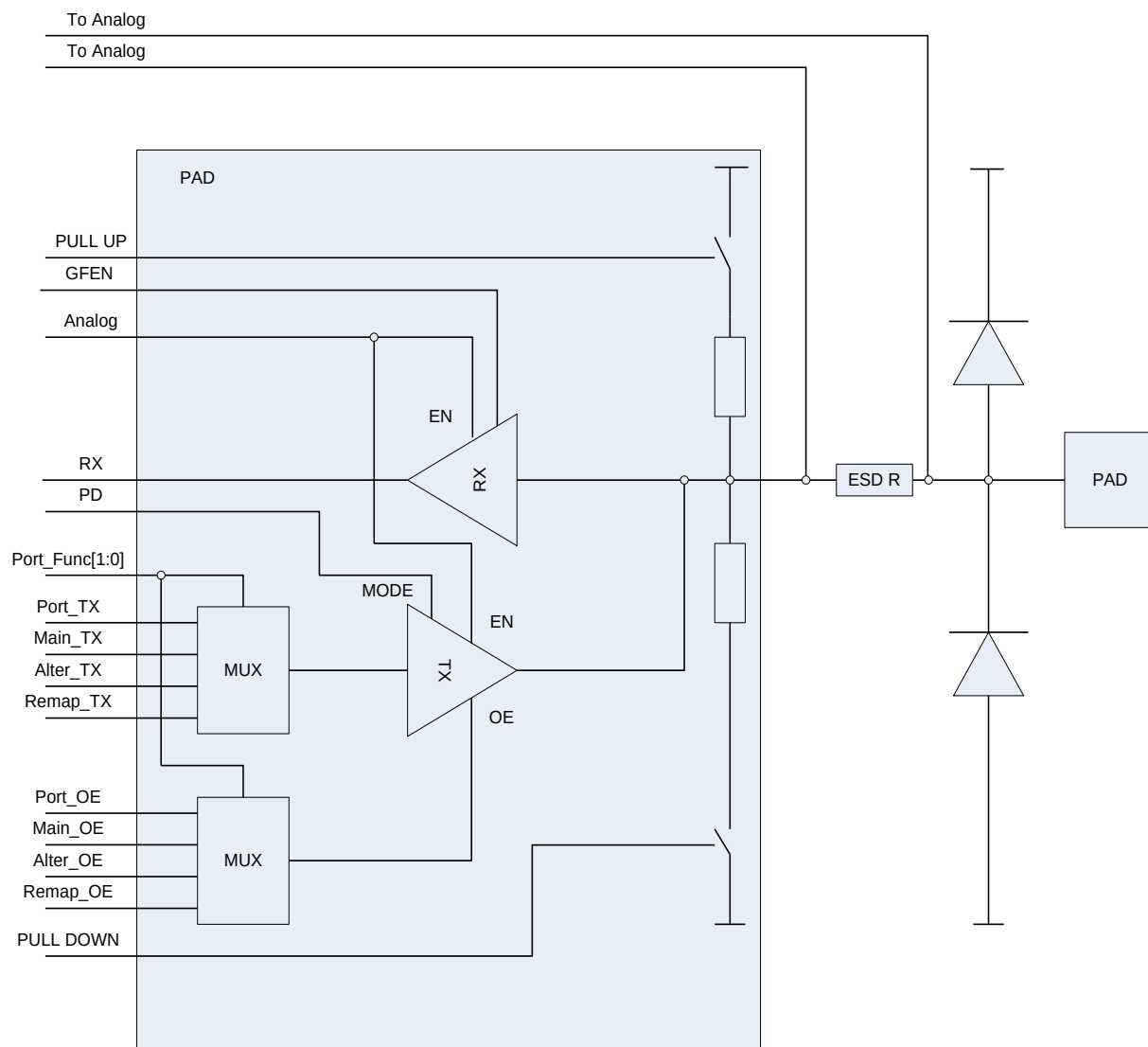


Рисунок 14-1 – Структурная схема порта ввода/вывода

14.1 Описание регистров портов ввода-вывода

Таблица 14-2 – Описание регистров портов ввода-вывода

Базовый Адрес	Название	Описание
0x400A_8000	MDR_PORTA	Порт А
0x400B_0000	MDR_PORTB	Порт В
0x400B_8000	MDR_PORTC	Порт С
0x400C_0000	MDR_PORTD	Порт D
0x400C_8000	MDR_PORTE	Порт E
0x400E_8000	MDR_PORTF	Порт F
Смещение		
0x00	RXTX[15:0]	MDR_PORTx->RXTX Данные порта
0x04	OE[15:0]	MDR_PORTx->OE Направление порта
0x08	FUNC[31:0]	MDR_PORTx->FUNC Режим работы порта

Базовый Адрес	Название	Описание
0x0C	ANALOG[15:0]	MDR_PORTx->ANALOG Аналоговый режим работы порта
0x10	PULL[31:0]	MDR_PORTx->PULL Подтяжка порта
0x14	PD[31:0]	MDR_PORTx->PD Режим работы выходного драйвера
0x18	PWR[31:0]	MDR_PORTx->PWR Режим мощности передатчика
0x1C	GFEN[15:0]	MDR_PORTx->GFEN Режим работы входного фильтра
0x20	SET[15:0]	MDR_PORTx->SET Установка данных выходного порта. Записываемые данные объединяются по «ИЛИ» с регистром выходных данных порта
0x24	CLR[15:0]	MDR_PORTx->CLR Установка данных выходного порта. Инверсия записываемых данных объединяется по «И» с регистром выходных данных порта
0x28	RD[15:0]	MDR_PORTx->RD Чтение регистра выходных данных порта

14.1.1 MDR_PORTx->RXTX

Таблица 14-3 – Регистр RXTX

Номер	31...16	15...0
Доступ	U	R/W
Сброс	0	0
	-	PORT RXTX[15:0]

Таблица 14-4 – Описание бит регистра RXTX

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...16	-	Зарезервировано
15...0	PORT RXTX[15:0]	Режим работы контроллера. Данные для выдачи на выходы порта и для чтения. Биты читаются как 1, если соответствующие им выходы настроены в аналоговый режим работы

14.1.2 MDR_PORTx->OE

Таблица 14-5 – Регистр OE

Номер	31...16	15...0
Доступ	U	R/W
Сброс	0	0
	-	PORT OE[15:0]

Таблица 14-6 – Описание бит регистра OE

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...16	-	Зарезервировано
15...0	PORT OE[15:0]	Режим работы контроллера. Направление передачи данных на выводах порта: 1 – выход; 0 – вход

14.1.3 MDR_PORTx->FUNC

Таблица 14-7 – Регистр FUNC

Номер	31	30	...	3	2	1	0
Доступ	R/W	R/W	...	R/W	R/W	R/W	R/W
Сброс	0	0	...	0	0	0	0
	MODE15[1:0]		...	MODE1[1:0]		MODE0[1:0]	

Таблица 14-8 – Описание бит регистра FUNC

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...2	MODEx	Аналогично MODE0 для остальных бит порта
1...0	MODE0[1:0]	Режим работы вывода порта: 00 – порт; 01 – основная функция; 10 – альтернативная функция 11 – переопределенная функция

14.1.4 MDR_PORTx->ANALOG

Таблица 14-9 – Регистр ANALOG

Номер	31...16	15...0
Доступ	U	R/W
Сброс	0	0
	-	ANALOG EN[15:0]

Таблица 14-10 – Описание бит регистра ANALOG

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...16	-	Зарезервировано
15...0	ANALOG EN[15:0]	Режим работы контроллера: 0 – аналоговый; 1 – цифровой

14.1.5 MDR_PORTx->PULL

Таблица 14-11 – Регистр PULL

Номер	31...16	15...0
Доступ	R/W	R/W
Сброс	0	0
	PULL UP[15:0]	PULL DOWN[15:0]

Таблица 14-12 – Описание бит регистра PULL

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...16	PULL UP15:0]	Режим работы контроллера. Разрешение подтяжки вверх: 0 – подтяжка в питание выключена; 1 – подтяжка в питание включена (есть подтяжка ~50 кОм)
15...0	PULL DOWN[15:0]	Режим работы контроллера. Разрешение подтяжки вниз: 0 – подтяжка в ноль выключена; 1 – подтяжка в ноль включена (есть подтяжка ~ 50 кОм)

14.1.6 MDR_PORTx->PD

Таблица 14-13 – Регистр PD

Номер	31...16	15...0
Доступ	R/W	R/W
Сброс	0	0
	PORT SHM[15:0]	PORT PD[15:0]

Таблица 14-14 – Описание бит регистра PD

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...16	PORT SHM[15:0]	Режим работы контроллера. Режим работы входа: 0 – триггер Шмитта выключен, гистерезис 200 мВ; 1 – триггер Шмитта включен, гистерезис 400 мВ
15...0	PORT PD[15:0]	Режим работы контроллера. Режим работы выхода: 0 – управляемый драйвер; 1 – открытый сток

14.1.7 MDR_PORTx->PWR

Таблица 14-15 – Регистр PWR

Номер	31	30	...	3	2	1	0
Доступ	R/W	R/W	...	R/W	R/W	R/W	R/W
Сброс	0	0	...	0	0	0	0
	PWR15[1:0]		...	PWR1[1:0]		PWR0[1:0]	

Таблица 14-16 – Описание бит регистра PORTx_PWR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...2	PWRx	Аналогично PWR0 для остальных бит порта
1...0	PWR0[1:0]	Режим работы вывода порта: 00 – зарезервировано (передатчик отключен); 01 – медленный фронт; 10 – быстрый фронт; 11 – максимально быстрый фронт

14.1.8 MDR_PORTx->GFEN

Таблица 14-17 – Регистр GFEN

Номер	31...16	15...0
Доступ	R/W	R/W
Сброс	0	0
	-	GFEN[15:0]

Таблица 14-18 – Описание бит регистра GFEN

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...16	-	Зарезервировано
15...0	GFEN[15:0]	Режим работы входного фильтра: 0 – фильтр выключен; 1 – фильтр включен

14.1.9 MDR_PORTx->SET

Таблица 14-19 – Регистр SET

Номер	31...16	15...0
Доступ	U	W
Сброс	0	0
	-	SET[15:0]

Таблица 14-20 – Описание бит регистра SET

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...16	-	Зарезервировано
15...0	SET[15:0]	Установка данных выходного порта. Записываемые данные объединяются по «ИЛИ» с регистром выходных данных порта. При чтении читается регистр выходных данных порта

14.1.10 MDR_PORTx->CLR

Таблица 14-21 – Регистр CLR

Номер	31...16	15...0
Доступ	U	W
Сброс	0	0
	-	CLR[15:0]

Таблица 14-22 – Описание бит регистра CLR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...16	-	Зарезервировано
15...0	CLR[15:0]	Установка данных выходного порта. Инверсия записываемых данных объединяется по «И» с регистром выходных данных порта

14.1.11 MDR_PORTx->RD

Таблица 14-23 – Регистр RD

Номер	31...16	15...0
Доступ	U	R
Сброс	0	0
	-	RD[15:0]

Таблица 14-24 – Описание бит регистра RD

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...16	-	Зарезервировано
15...0	RD[15:0]	Чтение регистра выходных данных порта

15 Детектор напряжения питания MDR_POWER

Блок детектора напряжения питания PVD предназначен для контроля питания U_{CC} и U_{CCB} при работе микросхемы. Блок PVD позволяет сравнивать внешние уровни напряжения с внутренними опорными уровнями и в случае превышения или снижения ниже опорного уровня выработать сигнал или прерывание для программной обработки.

Уровень опорного напряжения для сравнения с U_{CC} задается битами PLS[2:0] в регистре PVDCS, для сравнения с U_{CCB} задается битами PLBS[1:0] в регистре PVDCS. В соответствии с уровнями напряжения формируются флаги PVD и PBVD. Данные флаги выставляются при возникновении события и сбрасываются программно.

В микросхемах ревизии 1 (по адресу 0x000003FC загрузочного ПЗУ хранится значение 0x03400FDF) при $U_{CCB} < U_{CC}$ блок определения уровня U_{CCB} не функционирует.

Таблица 15-1 – Типовые уровни напряжений детектора питания

Параметр	Не менее	Типовое	Не более
Входное напряжение, U_{CC} , В	2,0	-	3,6
Входное напряжение, U_{CCB} , В	1,8	-	3,6
Уровень срабатывания PVD от U_{CC} , при PLS = «000», В		2,0	
Уровень срабатывания PVD от U_{CC} , при PLS = «001», В		2,2	
Уровень срабатывания PVD от U_{CC} , при PLS = «010», В		2,4	
Уровень срабатывания PVD от U_{CC} , при PLS = «011», В		2,6	
Уровень срабатывания PVD от U_{CC} , при PLS = «100», В		2,8	
Уровень срабатывания PVD от U_{CC} , при PLS = «101», В		3,0	
Уровень срабатывания PVD от U_{CC} , при PLS = «110», В		3,2	
Уровень срабатывания PVD от U_{CC} , при PLS = «111», В		3,4	
Уровень срабатывания PBVD от U_{CCB} , при PBLBS = «00», В		1,8	
Уровень срабатывания PBVD от U_{CCB} , при PBLBS = «01», В		2,2	
Уровень срабатывания PBVD от U_{CCB} , при PBLBS = «10», В		2,6	
Уровень срабатывания PBVD от U_{CCB} , при PBLBS = «11», В		3,0	

Таблица 15-2 – Описание регистров блока PVD

Базовый Адрес	Название	Описание
0x4005_8000	MDR_POWER	Датчик подсистемы питания
Смещение		
0x00	PVDCS [12:0]	Регистр MDR_POWER->PVDCS управления и состояния датчика питания

15.1 MDR_POWER->PVDCS

Таблица 15-3 – Регистр PVDCS

Номер	9	8	7	6	5...3	2...1	0
Доступ	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	000	00	0
	IEPVD	IEPVBD	PVD	PVBD	PLS [2:0]	PBLBS [1:0]	PVD EN

Номер	31...12	11	10
Доступ	U	R/W	R/W
Сброс	0	0	0
	-	INV	INVB

Таблица 15-4 – Описание бит регистра PVDCS

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...12	-	Зарезервировано
11	INV	Флаг инверсии выхода от датчика PVD: 0 – нет инверсии; 1 – инверсия. Если флаг не инвертируется, то флаг выставляется при превышении заданного уровня, если инвертируется, то при снижении ниже заданного уровня
10	INVB	Флаг инверсии выхода от датчика PVBD: 0 – нет инверсии; 1 – инверсия. Если флаг не инвертируется, то флаг выставляется при превышении заданного уровня, если инвертируется, то при снижении ниже заданного уровня
9	IEPVD	Флаг разрешения прерывания от датчика PVD: 0 – прерывание запрещено; 1 – прерывание разрешено. Очищается записью 0, если при очистке, датчик продолжает выдавать сигнал, то флаг не будет очищен
8	IEPVBD	Флаг разрешения прерывания от датчика PVBD: 0 – прерывание запрещено; 1 – прерывание разрешено. Очищается записью 0, если при очистке, датчик продолжает выдавать сигнал, то флаг не будет очищен
7	PVD	Результат сравнения напряжения основного питания Устанавливается событием, очищается записью 0. Если событие продолжается запись игнорируется: 0 – напряжение питания меньше чем уровень задаваемый PLS; 1 – напряжение питания больше чем уровень задаваемый PLS. Примечание – Сброс флага необходимо проводить с подтверждением – сбрасывать дважды
6	PVBD	Результат сравнения напряжения батарейного питания Устанавливается событием, очищается записью 0. Если событие продолжается запись игнорируется: 0 – напряжение питания меньше чем уровень задаваемый PBLs; 1 – напряжение питания больше чем уровень задаваемый PBLs. Примечание – Сброс флага необходимо проводить с подтверждением – сбрасывать дважды

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
5...3	PLS[2:0]	Уровень напряжения для сравнения с напряжением основного питания: 000 – 2,0 В; 001 – 2,2 В; 010 – 2,4 В; 011 – 2,6 В; 100 – 2,8 В; 101 – 3,0 В; 110 – 3,2 В; 111 – 3,4 В
2...1	PBLS[1:0]	Уровень напряжения для сравнения с напряжением батарейного питания: 00 – 1,8 В; 01 – 2,2 В; 10 – 2,6 В; 11 – 3,0 В
0	PVDEN	Бит разрешения работы блока датчика напряжения питания: 0 – датчик отключен; 1 – датчик включен

16 Внешняя системная шина MDR_EBC

Внешняя системная шина позволяет работать с внешними микросхемами памяти и периферийными устройствами. Области адресного пространства микросхемы отведены для работы с внешней системной шиной.

Таблица 16-1 – Адресные диапазоны внешней системной шины

Адресный диапазон	Размер	Описание
0x1000_0000 – 0x1FFF_FFFF	256 Мбайт	Область памяти секции CODE отображаемая на внешнюю системную шину с доступом через I Code и D code шины. В режиме микропроцессора из этой области начинается выполняться программа
0x5000_0000 – 0xDFFF_FFFF	2,256 Гбайт	Область памяти секции PERIPHERAL и EXTERNAL BUS отображаемая на внешнюю системную шину с доступом через S Bus. К этой области имеет доступ DMA контроллер

Контроллер внешней системной шины во всех режимах не формирует сигналов выборки чипа CE. При работе с внешними статическими ОЗУ, ПЗУ и периферийными устройствами в качестве сигнала выборки чипа можно использовать старшие линии шины адреса, не используемые для непосредственной адресации, либо использовать программно управляемые выходы портов для формирования сигналов CE.

16.1 Работа с внешними статическими ОЗУ, ПЗУ и периферийными устройствами

Для работы контроллера внешней системной шины с внешними микросхемами статического ОЗУ, ПЗУ или внешними периферийными устройствами необходимо задать режим работы через регистр EXT_BUS_CONROL. Бит RAM разрешает работу с внешними ОЗУ, бит ROM разрешает только чтение внешних ОЗУ или ПЗУ. В зависимости от скорости работы ядра микросхемы и внешних устройств необходимо задать времена транзакции на внешней системной шине через биты WAIT_STATE[3:0]. После этого все обращения в область памяти, отображаемой на внешнюю системную шину, будут транслироваться на выходы внешней системной шины ADDR, DATA и сигналы управления OE, WE, BE[3:0] и сигнал синхронизации CLOCK.

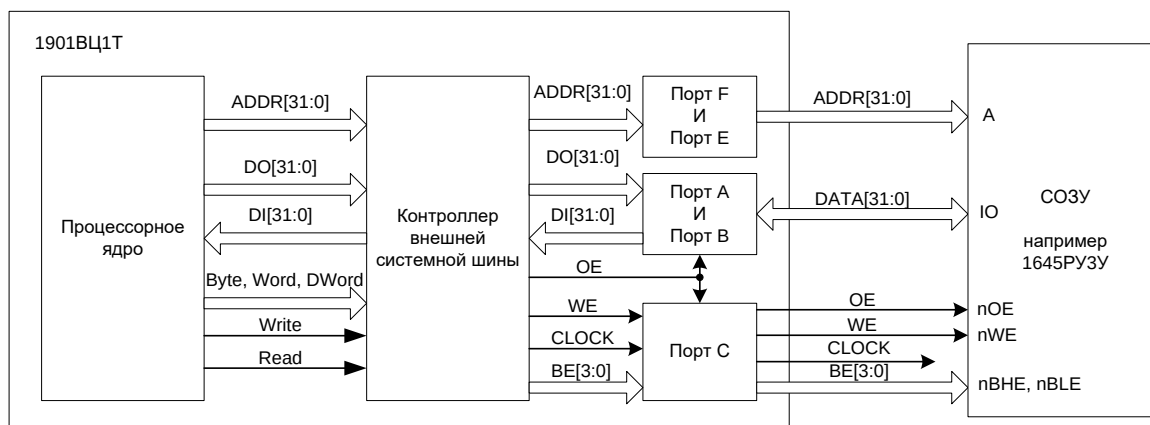


Рисунок 16-1 – Обмен по внешней системной шине

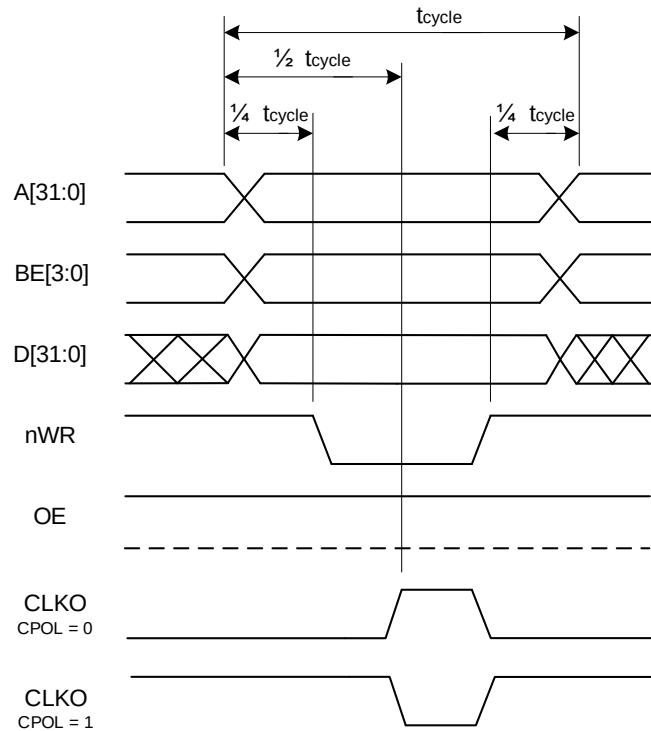


Рисунок 16-2 – Диаграмма записи

Время цикла записи t_{cycle} задается битами $WAIT\_STATE[3:0]$. Активный уровень сигналов WE , OE , $BE[3:0]$ низкий. Если сигнал $CLOCK$ не требуется, он может не использоваться.

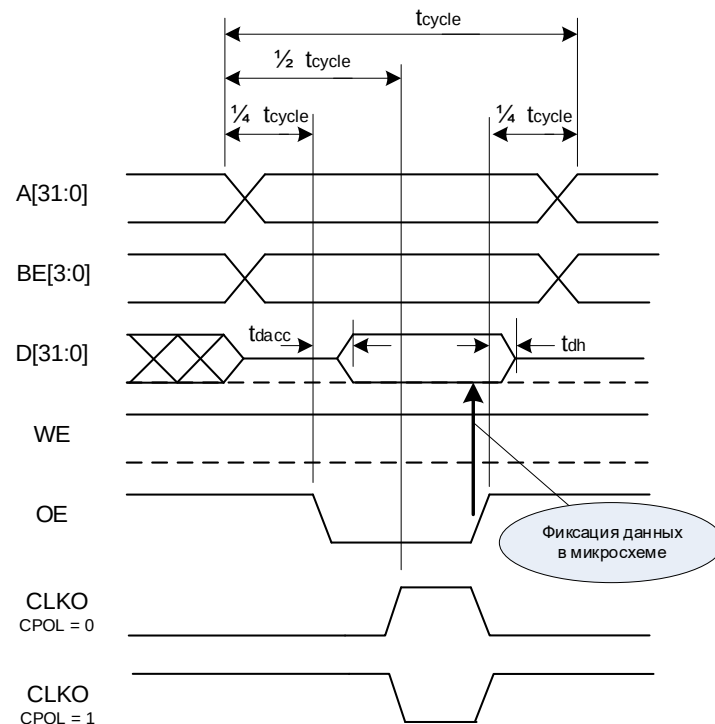


Рисунок 16-3 – Диаграмма чтения

При чтении по внешней системной шине необходимо выбрать такую длительность времени t_{cycle} , чтобы выполнялось время скорости доступа к памяти. Время t_{dh} для микросхемы равно нулю.

Для уменьшения задержек в работе контроллера внешней системной шины направление шины данных вне цикла чтения/записи (неактивное состояние) зависит от последнего обращения:

- если выполнялась запись, после цикла записи шина данных продолжает работать на выход, значение D[31:0] неопределено;
- если выполнялось чтение, после цикла чтения шина данных продолжает работать на вход.

Шина адреса вне цикла чтения/записи работает на выход, значение A[31:0] = 0xFFFF_FFFF.

Таблица 16-2 – Длительность фаз обращения в тактах процессора

WAIT_STATE	Предустановка адреса и данных перед сигналом WE или OE	Длительность WE или OE	Удержание адреса и данных после сигнала WE или OE
0	1	1	0
1	1	1	1
2	1	1	1
3	1	2	1
4	2	2	1
5	2	3	1
6	2	3	2
7	2	4	2
8	3	4	2
9	3	5	2
10	3	5	3
11	3	6	3
12	4	6	3
13	4	7	3
14	4	7	4
15	4	8	4

16.2 Работа с внешней NAND FLASH-памятью

Для работы контроллера внешней системной шины с внешними NAND Flash микросхемами памяти необходимо задать режим работы через регистр EXT_BUS_CONROL. Бит NAND разрешает работу с внешними NAND Flash микросхемами. В зависимости от скорости работы ядра микросхемы и внешних устройств необходимо задать времена выполнения различных этапов работы NAND FLASH-памяти через регистр NAND_CYCLES. После этого обращения в область памяти, отображаемой на внешнюю системную шину, будут перекодироваться в командные, адресные и обмена данными циклы обращения с NAND Flash через выводы внешней системной шины DATA[7:0], ALE, CLE, BUSY1 и BUSY2.

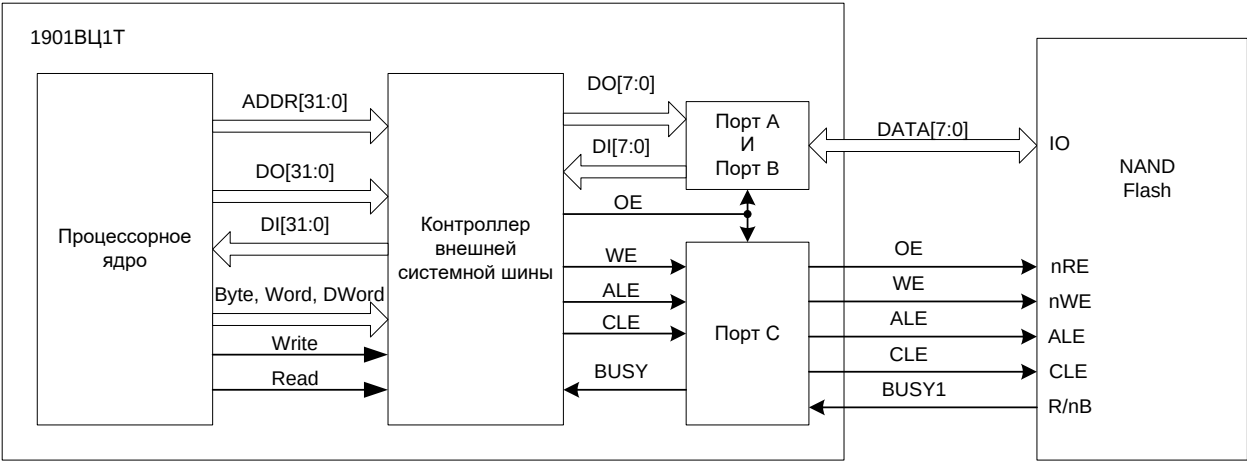


Рисунок 16-4 – Подключение внешней NAND Flash

Контроллер имеет два сигнала BUSY1 и BUSY2 для возможности подключения двух независимых микросхем NAND Flash. Оба сигнала объединяются по логическому И внутри контроллера и формируют общий сигнал BUSY. Если использование второго сигнала BUSY не требуется, то достаточно не задавать соответствующую функцию вывода порта D (BUSY1 – PD2 (основная функция) и BUSY1 – PD15 (альтернативная функция)).

При работе с NAND FLASH-памятью тип выполняемой операции кодируется адресом обращения, а данные и адрес передаются данными при записи и чтении памяти. Формат кодирования адреса обращения представлен ниже (Таблица 16-3).

Таблица 16-3 – Формат кодирования адреса обращения

Адрес обращения	Фаза команды	Фаза данных
ADDR[31:24]	Не имеет значения, но должно попадать в адресные диапазоны внешней системной шины: 0x10...0x1F; 0x30...0x3F; 0x50...0xCF	
ADDR[23:21]	ADR_CYCLES[2:0] 000 – 0 циклов; 001 – 1 цикл; ... 111 – 7 циклов	Не имеет значения
ADDR[20]	Выполнение завершающей команды: 0 – не выполнять; 1 – выполнять	
ADDR[19]	Всегда 0	Всегда 1
ADDR[18:11]	Код завершающей команды: ECMD[7:0] 0x10/0x11 – Page Program; 0xD0 – Block Erase	
ADDR[10:3]	Код начальной команды: SCMD[7:0] 0x00/0x01 – Read1; 0x50 – Read2;	Не имеет значения

Адрес обращения	Фаза команды	Фаза данных
	0x90 – Read ID; 0xFF – Reset; 0x80 – Page Program; 0x60 – Block Erase; 0x70 – Read Status	
ADDR[2:0]	Не имеет значения	

Более подробная информация о командах NAND FLASH-памяти представлена в документации на этот тип микросхем.

Пример работы с NAND FLASH-памятью

```
// =====
// Инициализация контроллера внешней системной шины для работы с NAND Flash
// =====
NAND_CYCLES = 0x02A63466;
// время t_rr = 2 цикла HCLK или 20 нс при частоте HCLK 100 МГц
// время t_alea = 10 циклов
// время t_whr = 6 циклов
// время t_wr = 3 цикла
// время t_rea = 4 цикла
// время t_wc = 6 циклов
// время t_rc = 6 циклов

EXT_BUS_CONTROL = 0x00000004;
// NAND = 1;

// =====
// Чтение ID микросхемы
// =====

unsigned char IDH;
unsigned char IDL;

// Фаза команды
*((volatile unsigned char *) (0x77200480)) = 0x00;
// ADR_CYCLE = 1
// SCMD = 0x90 (READ)
// Address 1 cycle = 0x00

// Фаза данных
IDL = *((volatile unsigned char *) (0x77080000));
IDH = *((volatile unsigned char *) (0x77080000));
```

```
// =====
// Стирание блока памяти
// =====

// Фаза команды
*((volatile unsigned char *)(0x70768300))=0x11;
*((volatile unsigned char *)(0x70768301))=0x22;
*((volatile unsigned char *)(0x70768302))=0x33;
// ADR_CYCLE = 3
// выполнять завершающую команду
// ECMD= 0xD0
// SCMD = 0x60
// Address 1 cycle = 0x11
// Address 2 cycle = 0x22
// Address 1 cycle = 0x33
while (EXT_BUS_CONTROL!=0x080 ) {};
// Ждем R/nB

// Фаза команды
*((volatile unsigned char *)(0x70000380+addon))=0x00;
// ADR_CYCLE = 0
// SCMD = 0x70
// Фаза данных
IDL = *((volatile unsigned char *)(0x77080000));
If (IDL & 0x01==0x01) Error ();
// Если бит IO0==1, то стирание не выполнено

// =====
// Запись страницы
// =====

// Фаза команды
*((volatile unsigned char *)(0x70800400))=0x11;
*((volatile unsigned char *)(0x70800400))=0x22;
*((volatile unsigned char *)(0x70800400))=0x33;
*((volatile unsigned char *)(0x70800400))=0x44;
// ADR_CYCLE = 4
// SCMD = 0x80

// Фаза данных
*((volatile unsigned char *)(0x70088000+addon))=0xBB;
*((volatile unsigned char *)(0x70088000+addon))=0xCC;
*((volatile unsigned char *)(0x70088000+addon))=0xDD;
// не выполнять завершающую команду
// ECMD= 0x10
...
*((volatile unsigned char *)(0x70188000+addon))=0xEE;
// не выполнять завершающую команду
```

```
// ECMD= 0x10
// Данные 0 – 0xBB, 1 – 0xCC,... N – 0xEE
// N от 1 до 528
while (EXT_BUS_CONTROL!=0x080 ) {};
// Ждем R/nB

// Фаза команды
*((volatile unsigned char *)(0x70000380+addon))=0x00;
// ADR_CYCLE = 0
// SCMD = 0x70
// Фаза данных
IDL = *((volatile unsigned char *)(0x77080000));
If (IDL & 0x01==0x01) Error ();
// Если бит IO0==1, то запись не выполнена
// =====
// Чтение страницы
// =====

// Фаза команды
*((volatile unsigned char *)(0x70800000))=0x11;
*((volatile unsigned char *)(0x70800000))=0x22;
*((volatile unsigned char *)(0x70800000))=0x33;
*((volatile unsigned char *)(0x70800000))=0x44;
// ADR_CYCLE = 4
// SCMD = 0x00
while (EXT_BUS_CONTROL!=0x080 ) {};
// Ждем R/nB

// Фаза данных
IDL=*((volatile unsigned char *)(0x70080000));
IDH=*((volatile unsigned char *)(0x70080000));
If (IDL != 0xBB || IDH != 0xCC) Error ();
// Если считали не то, что записали, то ошибка
```

16.3 Описание регистров блока контроллера внешней системной шины

Таблица 16-4 – Описание регистров блока контроллера внешней системной шины

Базовый Адрес	Название	Описание
0x400F_0000	MDR_EBC	Контроллер внешней системной шины
Смещение		
0x50	NAND_CYCLES	Регистр MDR_EBC->NAND_CYCLES управления работой с NAND_Flash
0x54	CONTROL	Регистр MDR_EBC->CONTROL управления внешней системной шиной

16.3.1 MDR_EBC->NAND_CYCLES

Таблица 16-5 – Регистр NAND_CYCLES

Номер	31-28	27-24	23-20	19-16	15-12	11-8	7-4	3-0
Доступ	U	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Спрос		0	0	0	0	0	0	0
	-	t_rr	t_alea	t_whr	t_wp	t_rea	t_wc	t_rc

Таблица 16-6 – Описание бит регистра NAND_CYCLES

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...28		Зарезервировано
27...24	t_rr[3:0]	Время от снятия busy до операции чтения: 0000 – 0 HCLK циклов; 0001 – 1 HCLK цикл; 1111 – 15 HCLK циклов; Типовое значение для памяти NAND Flash составляет 20 нс
23...20	t_alea[3:0]	Время доступа к регистрам ID. Аналогично t_rr. Типовое значение для памяти NAND Flash составляет 100 нс
19...16	t_whr[3:0]	Время доступа к регистру статуса. Аналогично t_rr. Типовое значение для памяти NAND Flash составляет 60 нс
15...12	t_wp[3:0]	Время доступа по записи. Аналогично t_rr. Типовое значение для памяти NAND Flash составляет 25 нс
11...8	t_rea[3:0]	Время доступа по чтению. Аналогично t_rr. Типовое значение для памяти NAND Flash составляет 35 нс
7...4	t_wc[3:0]	Время цикла записи. Аналогично t_rr. Типовое значение для памяти NAND Flash составляет 60 нс
3...0	t_rc[3:0]	Время цикла чтения. Аналогично t_rr. Типовое значение для памяти NAND Flash составляет 60 нс

16.3.2 MDR_EBC->CONTROL

Таблица 16-7 – Регистр CONTROL

Номер	31...16	15	14	13	12	11...8	7	6...4	3	2	1	0
Доступ	U	R/W	R/W	R/W	R/W	U	RO	U	R/W	R/W	R/W	R/W
Спрос	0	0	0	0	0	0	1	0	0	0	0	0
	-	WAIT_STATE[3:0]				-	BUSY	-	CPOL	NAND	RAM	ROM

Таблица 16-8 – Описание бит регистра CONTROL

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...16	-	Зарезервировано
15...12	WAIT STATE[3:0]	Количество тактов шины АНВ, необходимых для стандартного цикла записи/чтения. Сигналы OE/WE устанавливаются в момент времени $\frac{1}{4}$ WAIT_STATE, снимаются в момент времени $\frac{3}{4}$ WAIT_STATE: 0000 – 3 такта HCLK; 0001 – 4 такта HCLK; ... 1111 – 17 тактов HCLK
11...8	-	Зарезервировано
7	BUSY	Сигнал занятости NAND FLASH-памяти: 1 – операция завершена; 0 – операция не завершена
6..4	-	Зарезервировано
3	CPOL	Бит задания полярности сигнала CLOCK: 0 – положительная полярность; 1 – отрицательная полярность
2	NAND	Бит глобального разрешения памяти NAND: 1 – выбрана NAND; 0 – NAND не выбрана. Одновременная установка нескольких бит 3..0 недопустима, в этом случае запрещается работа со всей памятью
1	RAM	Бит глобального разрешения памяти RAM: 1 – выбрана RAM; 0 – RAM не выбрана
0	ROM	Бит глобального разрешения памяти ROM: 1 – выбрана ROM; 0 – ROM не выбрана

17 Контроллер прямого доступа в память MDR_DMA

Контроллер прямого доступа в память RISC реализован для ускоренного переноса данных между блоками памяти микросхемы, периферийными устройствами и внешними микросхемами памяти.

Данный контроллер имеет доступ к ОЗУ RISC части, всем периферийным модулям RISC и DSP, памяти программ и памяти данных DSP, контроллеру выхода на внешнюю шину. Контроллер не имеет доступа к ROM и FLASH-памяти RISC, регистрам ядра DSP (Таблица 17-1). Контроллер DMA содержит 32 аппаратных/программных запроса. Аппаратные запросы реализованы от большинства периферийных устройств RISC и подсистем DSP (таблица 21–3). Каждый канал может быть сконфигурирован для обработки аппаратных или программных запросов.

Несмотря на то, что ядро DSP не имеет доступа к регистрам DMA RISC, в системе реализован специальный регистр, предоставляющий возможность задавать запросы при помощи ядра DSP. Подробнее об этом см. раздел 18 «Организация управления подсистемой DSP».

Таблица 17-1 – Таблица доступа к секторам памяти со стороны DMA RISC

Адрес DMA RISC	Сектор	Тип со стороны DMA (RISC)
0x0000_0000	BOOT ROM	NA
0x0800_0000	EEPROM	NA
0x1000_0000	EXTERNAL BUS	WR
0x2000_0000	SYSTEM RAM	WR
0x2200_0000	SYSTEM RAM Bit Band Region	NA
0x3000_0000	Регистры ядра DSP	NA
0x3000_0040	Регистры периферийных модулей DSP	WR
0x3000_0100	Память данных DSP	WR
0x3002_0000	Память программ DSP	WR
0x3000_0040	Регистры периферийных модулей RISC	WR
0x5000_0000	EXTERNAL BUS	WR
Обозначения: NA – нет доступа; RO – только чтение; WR – полный доступ (чтение/запись)		

17.1 Основные свойства контроллера DMA

Основные свойства и отличительные особенности:

- 32 канала DMA;
- каждый канал DMA имеет свои сигналы управления передачей данных;
- каждый канал DMA имеет программируемый уровень приоритета;
- каждый уровень приоритета обрабатывается, исходя из уровня приоритета, определяемого номером канала DMA;

- поддержка различного типа передачи данных:
 - память – память;
 - память – периферия;
 - периферия – память;
- поддержка различных типов DMA циклов;
- поддержка передачи данных различной разрядности;
- каждому каналу DMA доступна первичная и альтернативная структура управляющих данных канала;
 - все управляющие данные канала хранятся в системной памяти;
 - разрядность данных приемника равна разрядности данных передатчика;
 - количество передач в одном цикле DMA может программироваться от 1 до 1024;
- инкремент адреса передачи может быть больше чем разрядность данных.

17.2 Термины и определения

При описании контроллера используются следующие термины:

Таблица 17-2 – Термины и определения

Альтернативная	Альтернативная структура управляющих данных канала. Вы можете установить соответствующий регистр для изменения типа структуры данных (см. раздел «Структура управляющих данных канала»)
C	Идентификатор номера канала прямого доступа. Например: C=1 – канал DMA 1 C=23 – канал DMA 23
Канал	Возможны конфигурации контроллера с числом каналов до 32. Каждый канал содержит независимые сигналы управления передачей данных, которые могут инициировать передачу данных по каналу DMA
Управляющие данные канала	Структура данных находится в системной памяти. Вы можете программировать эту структуру данных так, что контроллер может выполнять передачу данных по каналу DMA в желаемом режиме. Контроллер должен иметь доступ к области системной памяти, где находится эта информация. Примечание – Любое упоминание в документе структуры данных означает управляющие данные канала
Цикл DMA	Все передачи DMA, которые контроллер должен выполнить для передачи N пакетов данных
Передача DMA	Акция пересылки одного байта, полуслова или слова. Общее количество передач DMA, которые контроллер выполняет для канала
Пинг-понг	Режим работы для выбранного канала, при котором контроллер получает начальный запрос и затем выполняет цикл DMA, используя первичную или альтернативную структуру данных.

	После завершения этого цикла DMA контроллер начинает выполнять новый цикл DMA, используя другую структуру данных. Контроллер сигнализирует об окончании каждого цикла DMA, позволяя главному процессору перенастраивать неактивную структуру данных. Контроллер продолжает переключаться от первичной к альтернативной структуре данных и обратно, до тех пор, пока он не прочитает «неправильную» структуру данных или пока он не завершит цикл без переключения к другой структуре
Первичная	Первичная структура управляющих данных канала. Контроллер использует эту структуру данных, если соответствующий разряд в регистре <code>chnl_pri_alt_set</code> установлен в 0
R	Степень числа 2, устанавливающее число передач DMA, которые могут произойти перед сменой арбитража. Количество передач DMA программируется в диапазоне от 1 до 1024 двоичными шагами от 2 в степени 0 до 2 в степени 10
Исполнение с изменением конфигурации	Режим работы для выбранного канала, при котором контроллер получает запрос от периферии и выполняет 4 DMA передачи, используя первичную структуру управляющих данных, которые настраивают альтернативную структуру управляющих данных. После чего контроллер начинает цикл DMA, используя альтернативную структуру данных. После того, как цикл закончится, если периферия устанавливает новый запрос на обслуживание, контроллер выполняет снова 4 DMA передачи, используя первичную структуру управляющих данных, которые опять перенастраивают альтернативную структуру управляющих данных. После чего контроллер начинает цикл DMA, используя альтернативную структуру данных. Контроллер будет продолжать работать вышеописанным способом до тех пор, пока не прочитает неправильную структуру данных или процессор не установит альтернативную структуру данных для обычного цикла. Контроллер устанавливает флаг <code>dma_done</code>, если окончание подобного режима работы происходит после выполнения обычного цикла

17.3 Функциональное описание

Ниже (Рисунок 17-1) показана упрощенная структурная схема контроллера.

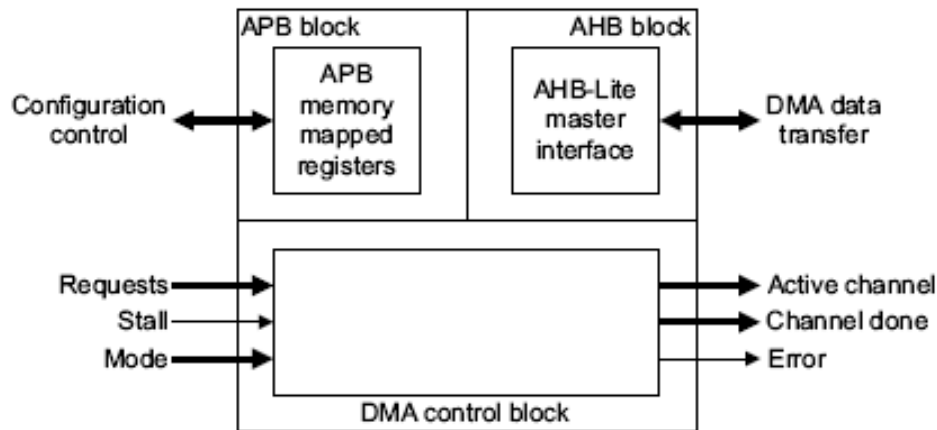


Рисунок 17-1 – Структурная схема контроллера

Контроллер состоит из следующих основных функциональных блоков:

- блок, подключенный к шине APB;
- блок, подключенный к шине AHB;
- управляющий блок DMA.

17.3.1 Распределение каналов DMA

Таблица 17-3 – Распределение каналов DMA

Номер канала	Источник req	Источник sreq	Описание
0	UART1 TX	UART1 TX	Запрос от UART1 по передаче
1	UART1 RX	UART1 RX	Запрос от UART1 по приему
2	UART2 TX	UART2 TX	Запрос от UART2 по передаче
3	UART2 RX	UART2 RX	Запрос от UART2 по приему
4	SSP1 TX	SSP1 TX	Запрос от SSP1 по передаче
5	SSP1 RX	SSP1 RX	Запрос от SSP1 по приему
6	SSP2 TX	SSP2 TX	Запрос от SSP2 по передаче
7	SSP2 RX	SSP2 RX	Запрос от SSP2 по приему
8	-	ADC1	Запрос от АЦП1
9	DSP0	ADC2	Запросы от ядра DSP/Запрос от АЦП2
10	DSP1	TIMER1	Запросы от ядра DSP/Запрос от Timer1
11	-	TIMER2	Запрос от Timer2
12	-	TIMER3	Запрос от Timer3
13	-	SDIO	Запрос от SDIO
14	DSP2	DSP0	Запросы от ядра DSP
15	DSP3	DSP1	Запросы от ядра DSP
16	SSP3 TX	SSP3 TX	Запрос от SSP3 по передаче
17	SSP3 RX	SSP3 RX	Запрос от SSP3 по приему
18	SSP4 TX	SSP4 TX	Запрос от SSP4 по передаче
19	SSP4 RX	SSP4 RX	Запрос от SSP4 по приему
20	UART3 TX	UART3 TX	Запрос от UART3 по передаче
21	UART3 RX	UART3 RX	Запрос от UART3 по приему

Номер канала	Источник req	Источник sreq	Описание
22	A_ADC(DSP)	-	Запрос от АЦП Аудиокодека(DSP)
23	A_DAC(DSP)	-	Запрос от ЦАП Аудиокодека(DSP)
24	Crypto(DSP)	-	Запрос от Криптомодуля(DSP)
25	TIMER (DSP)	TIMER (DSP)	Запрос от таймера DSP части
26	McBSP1 X	McBSP1 X	Запрос от McBSP1(DSP) по передаче
27	McBSP1 R	McBSP1 R	Запрос от McBSP1(DSP) по приему
28	McBSP2 X	McBSP2 X	Запрос от McBSP2(DSP) по передаче
29	McBSP2 R	McBSP2 R	Запрос от McBSP2(DSP) по приему
30	McBSP3 X	McBSP3 X	Запрос от McBSP3(DSP) по передаче
31	McBSP3 R	McBSP3 R	Запрос от McBSP3(DSP) по приему

17.3.2 Блок, подключенный к шине APB

Блок содержит набор регистров, позволяющих настраивать контроллер, используя ведомый интерфейс APB. Регистры занимают адресное пространство емкостью 4 Кбайт.

17.3.3 Блок, подключенный к шине АНВ

Контроллер содержит один блок типа «ведущий» шины DMA Bus, который позволяет, используя 32-разрядную шину, передавать данные от источника к приемнику. Источник и приемник являются ведомыми шины АНВ.

17.3.4 Управляющий блок DMA

Этот блок содержит схему управления, позволяющую реализовать следующие функции:

- осуществление арбитража поступающих запросов;
- индикацию активного канала;
- индикацию завершения обмена по каналу;
- индикацию состояния ошибки обмена по шине DMA Bus;
- разрешение медленным устройствам приостанавливать исполнение цикла DMA;
- ожидание запроса на очистку до завершения цикла DMA;
- осуществление одиночных или множественных передач DMA для каждого запроса;
- осуществление следующих типов DMA передач:
 - память – память;
 - память – периферия;
 - периферия – память.

17.3.5 Типы передач

Контроллер интерфейса не поддерживает пакетные передачи. Контроллер выполняет одиночные передачи. Отсутствие возможности осуществлять пакетные передачи оказывает минимальное влияние на производительность системы, так как пакетные передачи более эффективны в одноуровневых системах с шиной АНВ, где

блоки должны «захватывать» шину или обращаться к внешней памяти. В тоже время контроллер DMA предназначен для использования в многоуровневых системах с шиной АНВ, включающих встроенную память.

17.3.6 Разрядность передач данных

Контроллер интерфейса предоставляет возможность осуществлять передачу 8-, 16- и 32-разрядных данных. Таблица перечисляет значения комбинаций шины HSIZE.

Таблица 17-4 – Комбинации шины HSIZE

HSIZE[2]*	HSIZE[1]	HSIZE[0]	Разрядность данных (бит)
0	0	0	8
0	0	1	16
	1	0	32
	1	1	**
* Сигнал постоянно удерживается в состоянии логического «0».			
** Запрещенная комбинация			

Контроллер всегда использует передачи 32-разрядными данными при обращении к управляющим данным канала. Необходимо устанавливать разрядность данных источника, соответствующую разрядности данных приемника.

17.3.7 Управление защитой данных

Контроллер позволяет устанавливать режимы защиты данных протокола АНВ-Lite, определяемые шиной HPROT[3:1]. Возможен выбор следующих режимов защиты:

- кэширование;
- буферизация;
- привилегированный.

Таблица 17-5 перечисляет значения комбинаций шины HPROT.

Таблица 17-5 – Режимы защиты данных

HPROT[3] Кэширование	HPROT[2] буферизация	HPROT[1] Привилегиро- ванный	HPROT[0] Данные/ команда	Описание
-	-	-	1*	Доступ к данным
-	-	0	-	Пользовательский доступ
-	-	1	-	Привилегированный доступ
-	0	-	-	Без буферизации
-	1	-	-	Буферизированный
0	-	-	-	Без кэширования
1	-	-	-	Кэшированный
* Контроллер удерживает HPROT[0] в состоянии логической единицы, чтобы обозначить доступ к данным				

Для каждого цикла DMA возможен выбор режимов защиты данных передач источника и приемника. Более подробно это описано в разделе «Настройка управляющих данных».

Для каждого канала DMA также возможен выбор режима защиты данных. Более подробно это описано в разделе Управление DMA.

17.3.8 Инкремент адреса

Контроллер позволяет управлять инкрементом адреса при чтении данных из источника и при записи данных в приемник. Инкремент адреса зависит от разрядности передаваемых данных. В следующей таблице перечислены возможные комбинации.

Таблица 17-6 – Инкремент адреса

Разрядность данных	Величина инкремента
8	Байт, полуслово, слово
16	Полуслово, слово
32	слово

Минимальная величина инкремента адреса всегда соответствует разрядности передаваемых данных. Максимальная величина инкремента адреса, осуществляемая контроллером, одно слово. Более подробно о настройке инкремента адреса написано в разделе Настройка управляющих данных. Этот раздел описывает разряды управления величиной инкремента адреса в управляющих данных канала.

Примечание – Если необходимо оставлять адрес неизменным при чтении или записи данных, для примера, при работе с FIFO, можно соответствующим образом настроить контроллер на работу с фиксированным адресом (см. раздел 17.5 «Структура управляющих данных канала»).

17.4 Управление DMA

17.4.1 Правила обмена данными

Контроллер использует правила обмена данными, перечисленные далее в Таблица 17-7, при соблюдении следующих условий:

- канал DMA включен, что выполняется установкой в состояние логической единицы разрядов управления `chnl_enable_set[C]` и `master_enable`;
- флаги запроса `dma_req[C]` и `dma_sreq[C]` не замаскированы, что выполняется установкой в состояние логического нуля разряда управления `chnl_req_mask_set [C]`;

Таблица 17-7 – Правила, при которых передача данных по каналам разрешена, и запросы не маскируются

Правило	Описание
1	Если <code>dma_active[C]</code> установлен в 0, то установка в 1 <code>dma_req[C]</code> или <code>dma_sreq[C]</code> на один или более тактов сигнала <code>hclk</code> , следующих или не следующих друг за другом, инициирует передачу по каналу номер C
2	Контроллер осуществляет установку в 1 только одного разряда <code>dma_active[C]</code>
3	Контроллер устанавливает в 1 <code>dma_active[C]</code> в момент начала передачи по каналу C

Правило	Описание
4	Для типов циклов DMA, отличных от периферийного «Исполнение с изменением конфигурации», dma_active[C] остается в состоянии 1 до тех пор, пока контроллер не окончит передачи с номерами меньше, чем значение 2^R или чем число передач, указанное в регистре n_minus_1. В периферийном режиме «Исполнение с изменением конфигурации», dma_active[C] остается в состоянии 1 в течение каждой пары DMA передач, с использованием первичной и альтернативной структур управляющих данных. Таким образом, контроллер выполняет 2^R передач, используя первичную структуру управляющих данных, затем без осуществления арбитража выполняет передачи с номерами меньше, чем значение 2^R (или чем число передач, указанное в регистре n_minus_1), используя альтернативную структуру управляющих данных. По окончании последней передачи dma_active[C] сбрасывается в 0
5	Контроллер устанавливает dma_active[C] в 0 на как минимум один такт сигнала hclk перед тем, как снова установит dma_active[C] или dma_active[] в 1
6	Для каналов, по которым разрешена передача, контроллер осуществляет установку в 1 только одного dma_done[]
7	Если dma_req[C] устанавливается в состояние 1 в момент, когда dma_active[C] или dma_stall также в состоянии 1, то это означает, что контроллер обнаружил запрос
8	Если разряды cycle_ctrl для канала установлены в состояние 3'b100, 3'b101, 3'b110, 3'b111, то dma_done[C] никогда не будет установлен в 1
9	Если все передачи по каналу завершены, и разряды cycle_ctrl позволяют удержание dma_done[C], то по срезу сигнала dma_active[] произойдут события: – если dma_stall в состоянии 0, контроллер устанавливает dma_done[] в состояние 1 продолжительностью один такт hclk – если dma_stall в состоянии 1, работа контроллера приостановлена. После того, как dma_stall будет установлен в 0, контроллер устанавливает dma_done[] в состояние 1 продолжительностью один такт hclk
10	Состояние dma_waitonreq[C] можно изменять только при выключенном канале
11	Если dma_waitonreq[C] в состоянии 1, то сигнал dma_active[C] не перейдет в состояние 0 до тех пор, пока: контроллер завершит 2^R передач (или число передач, указанное в регистре n_minus_1); dma_req[C] будет установлен в 0; dma_sreq[C] будет установлен в 0
12	Если за один такт сигнала hclk перед установкой dma_active[C] в 0 dma_stall устанавливается в 1, то контроллер установит dma_active[C] в 0 на следующем такте сигнала hclk; передача по каналу C не завершится, пока не будет сброшен в 0 dma_stall
13	Контроллер игнорирует dma_sreq[C], если dma_waitonreq[C] в состоянии 0
14	Контроллер игнорирует dma_sreq[C], если chnl_useburst_set[C] в состоянии 1*
15	Для циклов DMA, отличных по типу от периферийного режима «Исполнение с изменением конфигурации», по окончании 2^R передач контроллер устанавливает значение chnl_useburst_set[C] в состояние 0, если количество оставшихся передач меньше, чем 2^R. В периферийном режиме «Исполнение с изменением конфигурации» контроллер устанавливает значение chnl_useburst_set[C] в состояние 0 только, если количество

Правило	Описание
	оставшихся передач с использованием альтернативной структуры управляющих данных меньше, чем 2^R .
16	Для типов циклов DMA, отличных от периферийного режима «Исполнение с изменением конфигурации», если за один такт hclk до установки dma_active[C] в 1 dma_sreq[C] и dma_waitonreq[C] установлены в 1 и dma_req[C] установлен в 0, то контроллер выполняет одну DMA передачу. В периферийном режиме «Исполнение с изменением конфигурации», если за один такт hclk до установки dma_active[C] в 1 dma_sreq[C] и dma_waitonreq[C] установлены в 1 и dma_req[C] установлен в 0, контроллер выполняет 2^R передач с использованием первичной структуры управляющих данных. Затем без осуществления арбитража выполняет одну передачу, используя альтернативную структуру управляющих данных
17	Для типов циклов DMA, отличных от периферийного режима «Исполнение с изменением конфигурации», если за один такт hclk до установки dma_active[C] в 1, а dma_sreq[C] и dma_req[C] установлены в 1, то приоритет предоставляется dma_req[c], и контроллер выполняет 2^R (или число передач, указанное в регистре n_minus_1) DMA передач. В периферийном режиме «Исполнение с изменением конфигурации», если за один такт hclk до установки dma_active[C] в 1 dma_sreq[C] и dma_req[C] установлены в 1, то приоритет предоставляется dma_req[c], и контроллер выполняет 2^R передач с использованием первичной структуры управляющих данных, затем без осуществления арбитража выполняет передачи с номерами меньше, чем значение 2^R (или чем число передач, указанное в регистре n_minus_1), используя альтернативную структуру управляющих данных
18	Когда chnl_req_mask_set[C] установлен в 1, контроллер игнорирует запросы по dma_sreq[C] и dma_req[C]
* Необходимо с осторожностью устанавливать эти разряды. Если значение, указанное в регистре n_minus_1 меньше, чем значение 2^R, то контроллер не очистит разряды chnl_useburst_set и поэтому запросы по dma_sreq[C] будут маскированы. Если периферия не устанавливает dma_req[C] в состояние 1, то контроллер никогда не выполнит необходимых передач	

При отключении канала контроллер осуществляет DMA передачи согласно правилам, представленным в Таблица 17-8.

Таблица 17-8 – Правила осуществления DMA передач при «запрещенных» каналах

Правило	Описание
19	Если dma_req[C] установлен в 1, то контроллер устанавливает dma_done[C] в 1. Это позволяет контроллеру показать центральному процессору запрос готовности, даже если канал выключен (запрещен)
20	Если dma_sreq[C] установлен в 1, то контроллер устанавливает dma_done[C] в 1 при условии dma_waitonreq[C] в 1 и chnl_useburst_set[C] в состоянии 0. Это позволяет контроллеру показать центральному процессору запрос готовности, даже если канал выключен (запрещен)
21	dma_active[C] всегда удерживается в состоянии 0

17.4.2 Диаграммы работы контроллера DMA

Данный раздел описывает примеры функционирования контроллера с использованием правил обмена данными, представленных ранее (Таблица 17-7):

- импульсный запрос на обработку;
- запрос по уровню на обработку;
- флаги завершения;
- флаги ожидания запроса на обработку.

Примечание – Все диаграммы, показанные далее в этом подразделе на рисунках ниже (Рисунок 17-2 - Рисунок 17-6), подразумевают следующее:

- hready находится в состоянии 1;
- АНВ «ведомый» всегда дает ответ «ОКАУ».

17.4.2.1 Импульсный запрос на обработку

Рисунок 17-2 показывает временную диаграмму работы контроллера DMA при получении импульсного запроса от периферии.

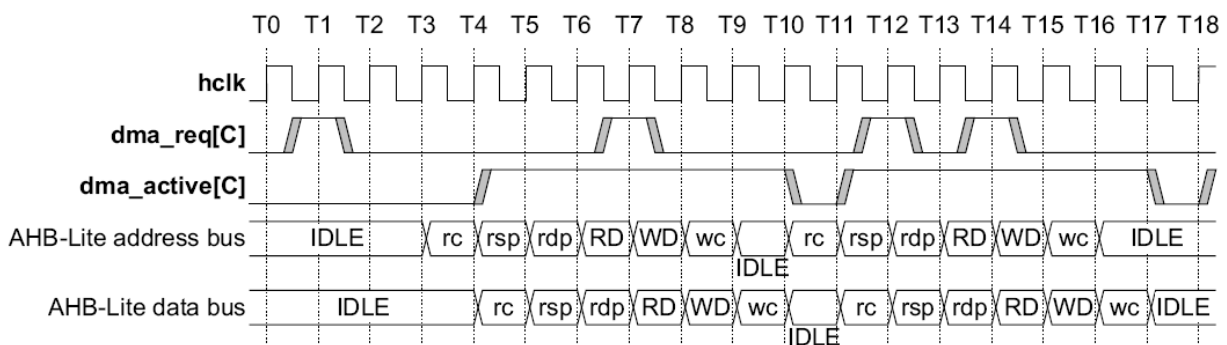


Рисунок 17-2 – Диаграмма работы при получении импульсного запроса

Пояснения к диаграмме (Рисунок 17-2) приведены ниже.

Таблица 17-9 – Пояснения к диаграмме работы при получении импульсного запроса

T1	Контроллер обнаружил запрос на обработку по каналу C (см. правило 1) при условии, что <code>chnl_req_mask_set[C]</code> находится в состоянии 0 (см. правило 18)
T4	Контроллер устанавливает <code>dma_active[C]</code> (см. правила 2 и 3) и начинает DMA передачи по каналу C
T4-T7	Контроллер считывает управляющую данные канала, где: rc – чтение настроек канала, <code>channel_cfg</code> ; rsp – чтение указателя адреса окончания данных источника, <code>src_data_end_ptr</code> ; rdp – чтение указателя адреса окончания данных приемника, <code>dst_data_end_ptr</code>
T7	При установленном <code>dma_active[C]</code> в 1 и при условии, что <code>chnl_req_mask_set[C]</code> находится в состоянии 0, контроллер обнаруживает импульс запроса на обработки по каналу C (см. правило 7). Контроллер обработает этот запрос в течение следующего арбитража
T7-T9	Контроллер выполняет передачу DMA по каналу C, где: RD – чтение данных; WD – запись данных
T9-T10	Контроллер осуществляет запись настроек канала, <code>channel_cfg</code> , где wc – запись настроек канала, <code>channel_cfg</code>

T10	Контроллер сбрасывает сигнал dma_active[C], что указывает на окончание передачи DMA (см. правило 4)
T10-T11	Контроллер удерживает dma_active[C] на,как минимум, один такт hclk (см. правило 5)
T11	Если канал C имеет более высокий приоритет, то контроллер устанавливает dma_active[C], так как ранее на такте T7 был получен запрос на обработку (см. правила 2 и 3)
T12	При установленном dma_active[C] в 1 и при условии, что chnl_req_mask_set[C] находится в состоянии 0, контроллер обнаруживает импульс запроса на обработки по каналу C (см. правило 7). Контроллер обработает этот запрос в течение следующего арбитража
T14	Контроллер игнорирует запрос по каналу C из-за отложенного запроса полученного на такте T12
T17	Контроллер сбрасывает сигнал dma_active[C], что указывает на окончание передачи DMA (см. правило 4)
T17-T18	Контроллер удерживает dma_active[C],как минимум, на один такт hclk (см. правило 5)
T18	Если канал C имеет более высокий приоритет, то контроллер устанавливает dma_active[C], так как ранее на такте T12 был получен запрос на обработку (см. правила 2 и 3)

17.4.2.2 Запрос на обработку по уровню

Рисунок 17-3 показывает временную диаграмму работы контроллера DMA при получении от периферии запроса на обработку по уровню.

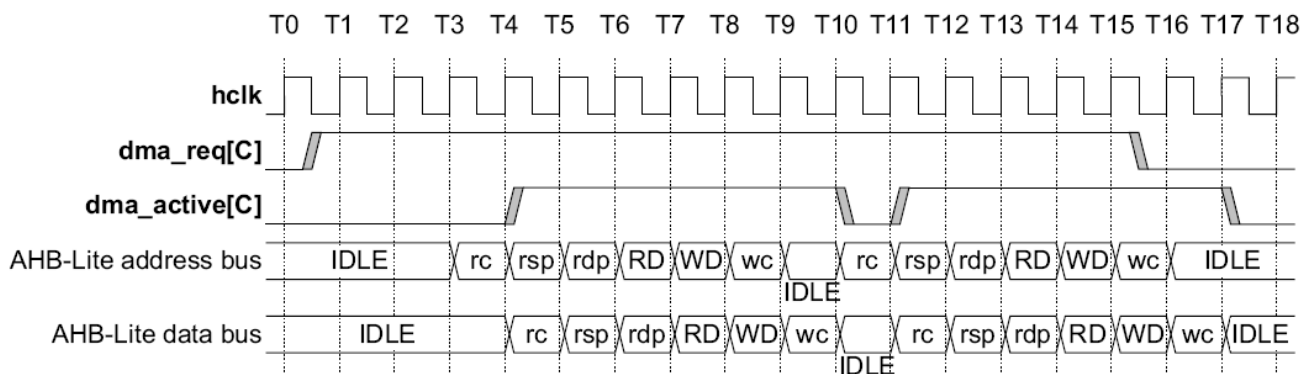


Рисунок 17-3 – Диаграмма работы при получении запроса на обработку по уровню

Пояснения к диаграмме (Рисунок 17-3) представлены ниже (Таблица 17-10).

Таблица 17-10 – Пояснения к диаграмме работы при получении запроса на обработку по уровню

T1	Контроллер обнаружил запрос на обработку по каналу C (Таблица 17-7, правило 1) при условии, что chnl_req_mask_set[C] находится в состоянии 0 (см. правило 18)
T4	Контроллер устанавливает dma_active[C] (см. правила 2 и 3) и начинает DMA передачи по каналу C

T4-T7	Контроллер считывает управляющие данные канала, где: rc – чтение настроек канала, channel_cfg; rsp – чтение указателя адреса окончания данных источника, src_data_end_ptr; rdp – чтение указателя адреса окончания данных приемника, dst_data_end_ptr
T7-T9	Контроллер выполняет передачу DMA по каналу C, где: RD – чтение данных WD – запись данных
T9-T10	Контроллер осуществляет запись настроек канала, channel_cfg, где wc – запись настроек канала, channel_cfg
T10	Контроллер сбрасывает сигнал dma_active[C], что указывает на окончание передачи DMA (см. правило 4). Контроллер обнаружил запрос на обработку по каналу C (см. правило 1) при условии, что chnl_req_mask_set[C] находится в состоянии 0 (см. правило 18).
T10-T11	Контроллер удерживает dma_active[C] на как минимум один такт hclk (см. правило 5)
T11	Если канал C имеет более высокий приоритет, то контроллер устанавливает dma_active[C] и начинает вторую DMA передачу по каналу C
T11-T14	Контроллер считывает управляющие данные канала
T14-T16	Контроллер выполняет передачу DMA по каналу C
T15-T16	Периферийный блок обнаруживает, что передача DMA началась и сбрасывает dma_req[C]
T16-T17	Контроллер осуществляет запись настроек канала channel_cfg
T17	Контроллер сбрасывает сигнал dma_active[C], что указывает на окончание передачи DMA (см. правило 4)

При использовании запроса на обработку по уровню периферийный блок может не обладать достаточным быстродействием, чтобы вовремя снять сигнал запроса, в этом случае он должен установить сигнал dma_stall. Установка сигнала dma_stall предотвращает повторение выполненной передачи.

17.4.2.3 Флаги завершения

Рисунок 17-4 демонстрирует функционирование сигнала (флага) dma_done[] при следующих условиях:

- dma_stall и dma_waitonreq[] находятся в состоянии 0;
- dma_stall установлен в 1;
- dma_waitonreq[] установлен в 1.

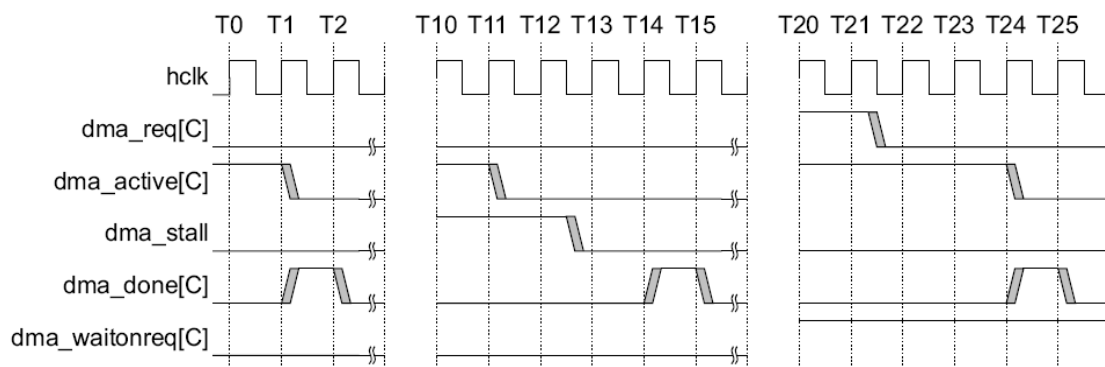


Рисунок 17-4 – Диаграммы функционирования dma_done

Пояснения к диаграмме (Рисунок 17-4), такты от T0 до T2, приведены в таблице ниже (Таблица 17-11).

Таблица 17-11 – Пояснения функционирования dma_done, такты от T0 до T2

T1	Контроллер сбрасывает сигнал dma_active[C], что указывает на окончание передачи DMA (см. Таблица 17-7, правило 4)
T1-T2	Контроллер завершает цикл DMA и если cycle_ctrl[2] установлен в 0, то устанавливает в 1 dma_done[C] на один такт hclk (см. правила 8 и 9). Для других разрешенных каналов сигнал dma_done[C] останется в состоянии 0 (см. правило 6)

Пояснения к диаграмме (Рисунок 17-4), такты от T10 до T15, приведены в таблице ниже (Таблица 17-12).

Таблица 17-12 – Пояснения функционирования dma_done, такты от T10 до T15

T11	Контроллер сбрасывает сигнал dma_active[C], что указывает на окончание передачи DMA (см. правило 4)
T12-T13	Периферийный блок сбрасывает сигнал dma_stall
T14-T15	Контроллер завершает цикл DMA и если cycle_ctrl[2] установлен в 0, то устанавливает в 1 dma_done[C] на один такт hclk (см. правила 8 и 9). Для других разрешенных каналов сигнал dma_done[C] останется в состоянии 0 (см. правило 6)

Примечание к T11 – Контроллер не устанавливает сигнал dma_done[C], так как сигнал dma_stall установлен в 1 в предшествующем такте hclk (см. правила 9, 12).

Пояснения к диаграмме (Рисунок 17-4), такты от T20 до T25, приведены в таблице ниже (Таблица 17-13).

Таблица 17-13 – Пояснения функционирования dma_done, такты от T20 до T25

T20	Контроллер выполнил передачу DMA, но из-за установленного в 1 dma_waitonreq[C] он должен ожидать сброса в 0 сигнала dma_req[C], перед тем как сбросить dma_active[C] (см. правило 11) и установить dma_done[C] (см. правило 9)
T21-T25	Периферийный блок сбрасывает dma_req[C]
T24	Контроллер сбрасывает сигнал dma_active[C], что указывает на окончание передачи DMA (см. правило 4)
T24-T25	Контроллер завершает цикл DMA и, если cycle_ctrl[2] установлен в 0, то устанавливает в 1 dma_done[C] на один такт hclk (см. правила 8 и 9). Для других разрешенных каналов сигнал dma_done[C] останется в состоянии 0 (см. правило 6)

17.4.2.4 Флаги ожидания запроса на обработку

Ниже приведены рисунки, которые демонстрируют примеры использования флагов ожидания запроса на обработку при выполнении 2^R передач и одиночных передач:

– диаграмма работы контроллера DMA при использовании периферией dma_waitonreq;

– диаграмма работы контроллера DMA при использовании периферией dma_waitonreq совместно с dma_sreq.

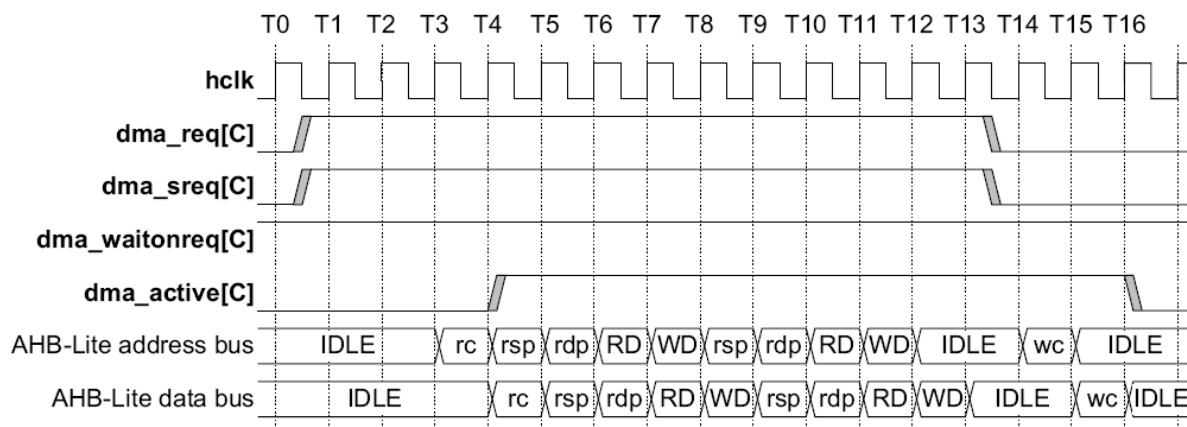


Рисунок 17-5 – Диаграмма работы контроллера DMA при использовании dma_waitonreq

Пояснения к диаграмме (Рисунок 17-5) приведены в таблице ниже (Таблица 17-14).

Таблица 17-14 – Пояснения работы контроллера DMA при использовании dma_waitonreq

T0-T16	Периферийный блок должен оставлять состояние dma_waitonreq[C] постоянно (см. правило 10)
T0-T1	Контроллер обнаружил запрос на обработку по каналу C (см. правило 1) при условии, что chnl_req_mask_set[C] находится в состоянии 0 (см. правило 18)
T3-T4	Периферийный блок удерживает dma_req[C] и dma_sreq[C] в 1. Контроллер игнорирует dma_sreq[C] запрос и отвечает на dma_req[C] запрос (см. правила 16 и 17)
T4	Контроллер устанавливает dma_active[C] (см. правила 2 и 3) и начинает DMA передачи по каналу C
T4-T7	Контроллер считывает управляющие данные канала, где: rc – чтение настроек канала, channel_cfg; rsp – чтение указателя адреса окончания данных источника, src_data_end_ptr; rdp – чтение указателя адреса окончания данных приемника, dst_data_end_ptr
T7-T9	Контроллер выполняет передачу DMA по каналу C, где: RD – чтение данных; WD – запись данных
T9-T11	Контроллер считывает 2 указателя адреса окончания данных rsp и rdp
T11-T13	Периферийный блок сбрасывает сигналы dma_req[C] и dma_sreq[C]
T15-T16	Контроллер осуществляет запись настроек канала, channel_cfg, где wc – запись настроек канала, channel_cfg
T16	Контроллер сбрасывает сигнал dma_active[C], что указывает на окончание передачи DMA (см. правило 11). Контроллер устанавливает значение по чтению регистра chnl_useburst_set[C] в 0, если количество оставшихся передач менее 2^R (см. правило 15)

Рисунок 17-6 показывает работу контроллера DMA при установке dma_waitonreq в «1» и выполнении одиночной DMA передачи.

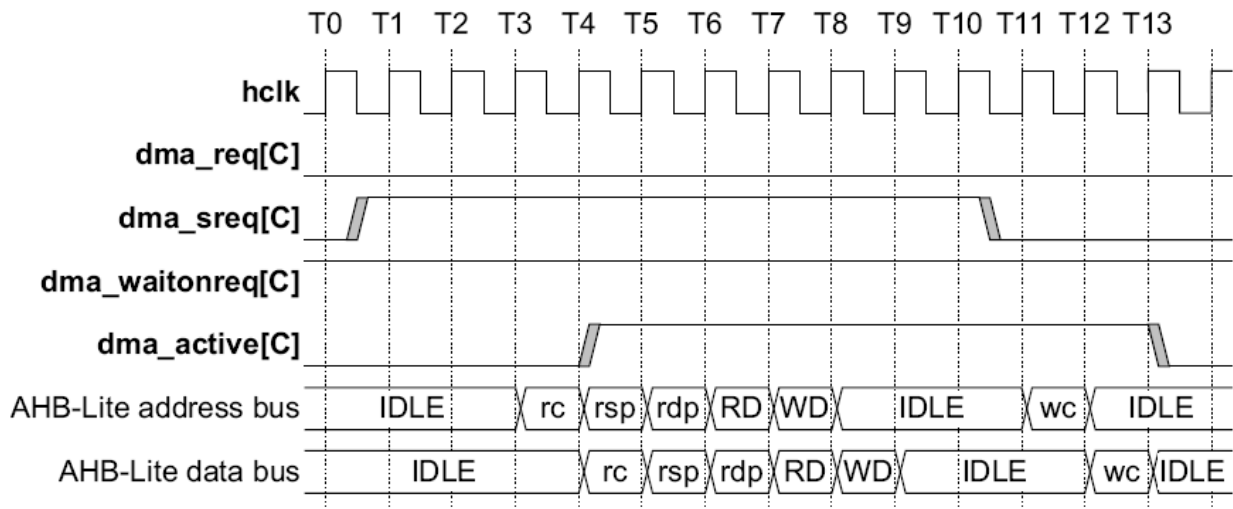


Рисунок 17-6 – Работа DMA при использовании dma_waitonreq совместно с dma_sreq

Пояснения к диаграмме (Рисунок 17-6) приведены в таблице ниже (Таблица 17-15).

Таблица 17-15 – Пояснения работы DMA при использовании dma_waitonreq совместно с dma_sreq

T0-T13	Периферийный блок должен оставлять состояние dma_waitonreq[C] постоянно (см. правило 10)
T0-T1	Контроллер обнаружил запрос на обработку по каналу C (см. правило 1) при условии, что chnl_useburst_set[C] находится в состоянии 0 (см. правила 13 и 14)
T3-T4	Контроллер отвечает на dma_sreq[C] запрос (см. правила 16)
T4	Контроллер устанавливает dma_active[C] (см. правила 2 и 3) и начинает DMA передачи по каналу C
T4-T7	Контроллер считывает управляющие данные канала, где: rc – чтение настроек канала, channel_cfg; rsp – чтение указателя адреса окончания данных источника, src_data_end_ptr; rdp - чтение указателя адреса окончания данных приемника, dst_data_end_ptr
T7-T9	Контроллер выполняет передачу DMA по каналу C, где: RD – чтение данных; WD – запись данных. Это запрос в ответ на dma_sreq[], таким образом, R=0 и, следовательно, контроллер исполнит 1 DMA передачу
T10-T11	Периферийный блок сбрасывает сигнал dma_sreq[C]
T12-T13	Контроллер осуществляет запись настроек канала, channel_cfg, где wc – запись настроек канала, channel_cfg
T13	Контроллер сбрасывает сигнал dma_active[C], что указывает на окончание передачи DMA (см. правило 11)

17.4.3 Правила арбитража DMA

Контроллер имеет возможность настройки момента арбитража при передачах DMA. Эта возможность позволяет уменьшить время отклика при обслуживании каналов с высоким приоритетом.

Контроллер имеет четыре разряда, которые определяют количество транзакций по шине АНВ до повторения арбитража. Эти разряды задают степень R числа 2; изменение R напрямую устанавливает периодичность арбитража как 2^R в степени R . Для примера, если R равно 4, то арбитраж будет проводиться через каждые 16 передач DMA.

Таблица 17-16 показывает возможную периодичность арбитража.

Таблица 17-16 – Периодичность арбитража в единицах передач по шине АНВ

Значение R	Периодичность арбитража каждые x передач DMA
b0000	1
b0001	2
b0010	4
b0011	8
b0100	16
b0101	32
b0110	64
b0111	128
b1000	256
b1001	512
b1010-b1111	1024

Примечание – Необходимо с осторожностью устанавливать большие значения R для низкоприоритетных каналов, так как это может привести к невозможности обслуживать запросы по высокоприоритетным каналам.

При $N > 2^R$ (N – номер передачи) и если результат деления 2^R на N не целое число, контроллер всегда выполняет последовательность из 2^R передач до тех пор, пока не станет верным $N < 2^R$. Контроллер выполняет оставшиеся N передач в конце цикла DMA.

Разряды степени R числа 2 находятся в структуре управляющих данных канала. Местонахождение этих разрядов описано в подразделе 17.5 «Структура управляющих данных канала».

17.4.4 Приоритет

При проведении арбитража определяется канал для обслуживания в следующем цикле DMA. На выбор следующего канала влияют:

- номер канала;
- уровень приоритета, присвоенного каналу.

Каждому каналу может быть присвоен уровень приоритета по умолчанию (низкий) или высокий уровень приоритета. Присвоение уровня приоритета осуществляется установкой или сбросом разряда `chnl_priority_set`.

Канал номер 0 имеет высший уровень приоритета и уровень приоритета снижается с увеличением номера канала. Таблица 17-17 показывает уровень приоритета каналов DMA в порядке его уменьшения.

Таблица 17-17 – Уровень приоритета каналов DMA

Уровень приоритета в порядке его уменьшения	Номер канала	Уровень приоритета, установленный битом chnl_priority_set
Наивысший уровень приоритета	0	Высокий
-	1	Высокий
-	2	Высокий
.....		
-	30	Высокий
-	31	Высокий
-	0	По умолчанию (низкий)
-	1	По умолчанию (низкий)
-	2	По умолчанию (низкий)
.....		
-	30	По умолчанию (низкий)
Низший уровень приоритета	31	По умолчанию (низкий)

После окончания цикла DMA контроллер выбирает следующий для обслуживания канал из всех включенных каналов DMA. Рисунок 17-7 иллюстрирует процесс выбора следующего канала для обслуживания.

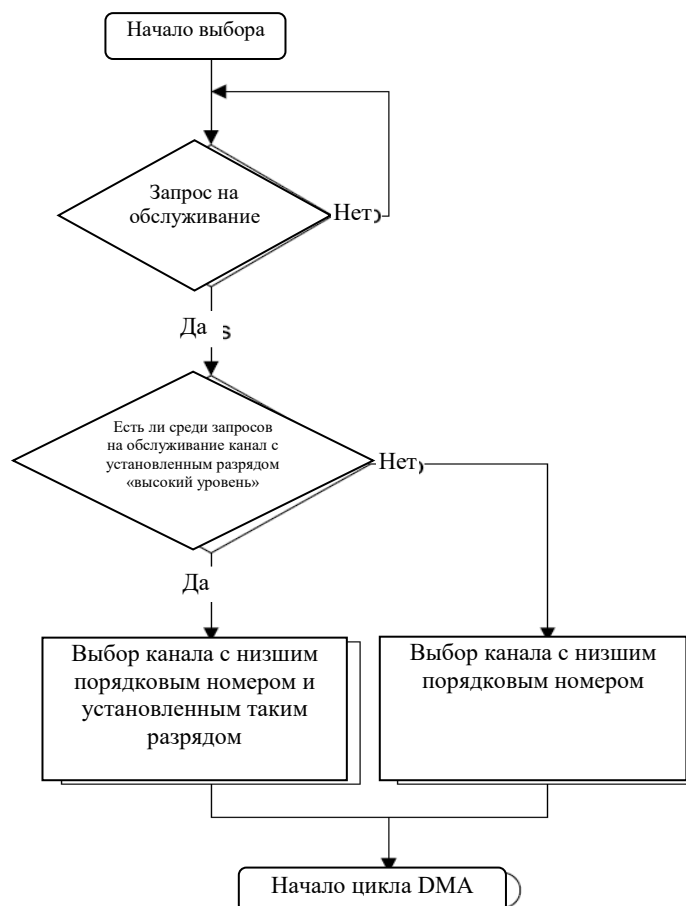


Рисунок 17-7 – Алгоритм выбора следующего канала для обслуживания

17.4.5 Типы циклов DMA

Разряды `cycle_ctrl` определяют, как контроллер будет выполнять циклы DMA. Описание значений этих разрядов приведено ниже.

Таблица 17-18 – Типы циклов DMA

cycle_ctrl	Описание
b000	Структура управляющих данных канала в запрещенном состоянии
b001	Обычный цикл DMA
b010	Авто-запрос
b011	Режим «пинг-понг»
b100	Работа с памятью в режиме «Исполнение с изменением конфигурации» с использованием первичных управляющих данных канала
b101	Работа с памятью в режиме «Исполнение с изменением конфигурации» с использованием альтернативных управляющих данных канала
b110	Работа с периферией в режиме «Исполнение с изменением конфигурации» с использованием первичных управляющих данных канала
b111	Работа с периферией в режиме «Исполнение с изменением конфигурации» с использованием альтернативных управляющих данных канала
Примечание – Разряды <code>cycle_ctrl</code> находятся в области памяти, отведенной под <code>channel_cfg</code> – см. подраздел 17.5 «Структура управляющих данных канала»	

Для всех типов циклов DMA повторный арбитраж происходит после 2^R передач DMA. Если установить длинный период арбитража на низкоприоритетном канале, то это заблокирует все запросы на обработку от других каналов до тех пор, пока не будут выполнены 2^R передач DMA по данному каналу. Поэтому, устанавливая значение R, необходимо учитывать, что это может привести к повышенному времени отклика на запрос на обработку от высокоприоритетных каналов.

Данный раздел описывает следующие типы циклов DMA:

- недействительный;
- основной;
- авто-запрос;
- «пинг-понг»;
- работа с памятью в режиме «исполнение с изменением конфигурации»;
- работа с периферией в режиме «исполнение с изменением конфигурации».

17.4.5.1 Недействительный

После окончания цикла DMA контроллер устанавливает тип цикла в значение «недействительный» для предотвращения повтора выполненного цикла DMA.

17.4.5.2 Основной

В этом режиме контроллер работает только с основными или альтернативными управляющими данными канала. После того, как разрешена работа канала, и контроллер получил запрос на обработку, цикл DMA выглядит следующим образом:

1 Контроллер выполняет 2^R передач. Если число оставшихся передач 0, контроллер переходит к шагу 3.

2 Осуществление арбитража:

– если высокоприоритетный канал выдает запрос на обработку, то контроллер начинает обслуживание этого канала;

– если периферийный блок или программное обеспечение выдает запрос на обработку (повторный запрос на обработку по каналу), то контроллер переходит к шагу 1.

3 Контроллер устанавливает `dma_done[C]` в состояние 1 на один такт сигнала `hclk`. Это указывает центральному процессору на завершение цикла DMA.

17.4.5.3 Авто-запрос

Функционируя в данном режиме, контроллер ожидает получения одиночного запроса на обработку для разрешения работы и выполнения цикла DMA. Такая работа позволяет выполнять передачу больших пакетов данных без существенного увеличения времени отклика на обслуживание высокоприоритетных запросов и не требует множественных запросов на обработку от процессора или периферийных блоков.

Контроллер позволяет выбрать для использования первичную или альтернативную структуру управляющих данных канала. После того, как разрешена работа канала, и контроллер получил запрос на обработку, цикл DMA выглядит следующим образом:

1 Контроллер выполняет 2^R передач для канала C. Если число оставшихся передач 0, контроллер переходит к шагу 3.

2 Контроллер осуществляет арбитраж. Когда канал C становится каналом с самым высоким приоритетом, контроллер переходит к шагу 1.

3 Контроллер устанавливает `dma_done[C]` в состояние 1 на один такт сигнала `hclk`. Это указывает центральному процессору на завершение цикла DMA.

17.4.5.4 Пинг-понг

В данном режиме контроллер выполняет цикл DMA, используя одну из структур управляющих данных, а затем выполняет еще один цикл DMA, используя другую структуру управляющих данных. Контроллер выполняет циклы DMA с переключением структур до тех пор, пока не считает «неправильную» структуру данных или пока процессор не запретит работу канала.

Рисунок 17-8 демонстрирует пример функционирования контроллера в режиме «пинг-понг».

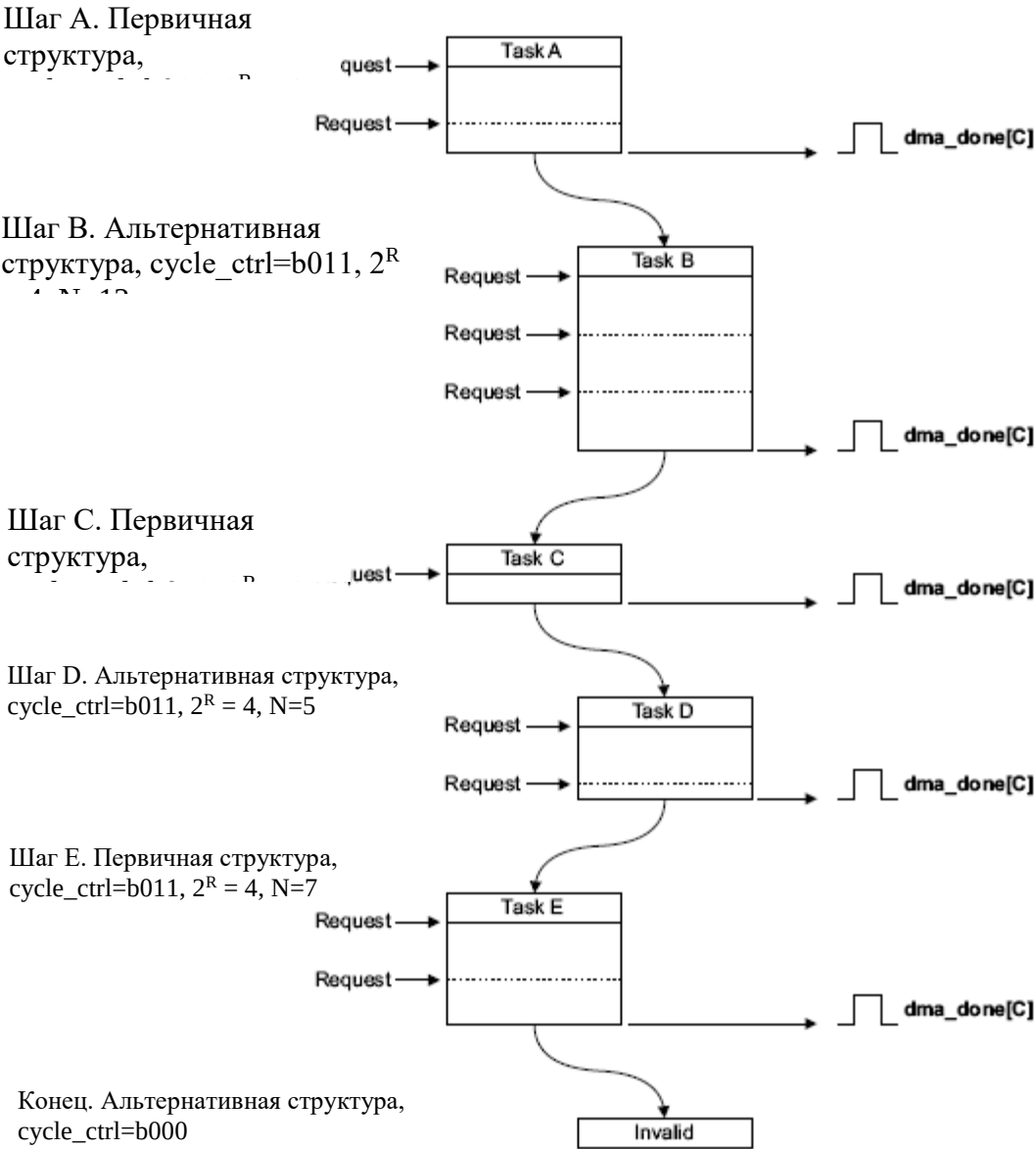


Рисунок 17-8 – Пример функционирования контроллера в режиме «пинг-понг»
Пояснения к схеме (Рисунок 17-8):

Шаг А	Процессор устанавливает первичную структуру управляющих данных для шага А. Процессор устанавливает альтернативную структуру управляющих данных для шага В. Это позволит контроллеру переключиться к шагу В незамедлительно после выполнения шага А, при условии, что контроллер не получит запрос на обработку от высокоприоритетного канала. Контроллер получает запрос и выполняет четыре передачи DMA. Контроллер выполняет арбитраж. После получения запроса на обработку от этого же канала, контроллер продолжает цикл в ситуации отсутствия высокоприоритетных запросов. Контроллер выполняет оставшиеся две передачи DMA. Контроллер устанавливает $\text{dma\_done}[C]$ в состояние 1 на один такт сигнала синхронизации hclk и входит в процедуру арбитража

После выполнения шага А процессор может установить первичные управляющие данные канала для шага С. Это позволит контроллеру переключиться к шагу С

незамедлительно после выполнения шага В, при условии, что контроллер не получит запрос на обработку от высокоприоритетного канала.

После получения нового запроса на обработку от канала при условии его наивысшего приоритета выполняется шаг В:

Шаг В	Контроллер выполняет четыре передачи DMA. Контроллер выполняет арбитраж. После получения запроса на обработку от этого же канала контроллер продолжает цикл в ситуации отсутствия высокоприоритетных запросов. Контроллер выполняет четыре передачи DMA. Контроллер выполняет арбитраж. После получения запроса на обработку от этого же канала контроллер продолжает цикл в ситуации отсутствия высокоприоритетных запросов. Контроллер выполняет оставшиеся четыре передачи DMA. Контроллер устанавливает dma_done[C] в состояние 1 на один такт сигнала синхронизации hclk и входит в процедуру арбитража

После выполнения шага В процессор может установить альтернативные управляющие данные канала для шага D.

После получения нового запроса на обработку от канала при условии его наивысшего приоритета выполняется шаг С:

Шаг С	Контроллер выполняет две передачи DMA. Контроллер устанавливает dma_done[C] в состояние 1 на один такт сигнала синхронизации hclk и входит в процедуру арбитража

После выполнения шага С процессор может установить первичные управляющие данные канала для шага E.

После получения нового запроса на обработку от канала при условии его наивысшего приоритета выполняется шаг D:

Шаг D	Контроллер выполняет четыре передачи DMA. Контроллер выполняет арбитраж. После получения запроса на обработку от этого же канала контроллер продолжает цикл в ситуации отсутствия высокоприоритетных запросов Контроллер выполняет оставшуюся передачу DMA. Контроллер устанавливает dma_done[C] в состояние 1 на один такт сигнала синхронизации hclk и входит в процедуру арбитража

После получения нового запроса на обработку от канала при условии его наивысшего приоритета выполняется шаг E:

Шаг Е	Контроллер выполняет четыре передачи DMA. Контроллер выполняет арбитраж. После получения запроса на обработку от этого же канала контроллер продолжает цикл в ситуации отсутствия высокоприоритетных запросов. Контроллер выполняет оставшиеся три передачи DMA. Контроллер устанавливает dma_done[C] в состояние 1 на один такт сигнала синхронизации hclk и входит в процедуру арбитража

Если контроллер получит новый запрос на обработку от данного канала и этот запрос будет самым приоритетным, контроллер предпримет попытку выполнения следующего шага. Однако из-за того, что процессор не установил альтернативные управляющие данные, и по окончании шага D контроллер установил cycle_ctrl в состояние b000, передачи DMA прекращаются.

Примечание – Для прерывания цикла DMA, исполняемого в режиме «пинг-понг», также возможен перевод режима работы контроллера на шаге Е в режим «Основной цикл DM» путем установки cycle_ctrl в 3'b001.

17.4.5.5 *Режим работы с памятью «исполнение с изменением конфигурации»*

В данном режиме контроллер, получая начальный запрос на обработку, выполняет четыре передачи DMA, используя первичные управляющие данные. По окончании этих передач контроллер начинает цикл DMA используя альтернативные управляющие данные. Затем контроллер выполняет еще четыре передачи DMA, используя первичные управляющие данные. Контроллер продолжает выполнять циклы ПДА, меняя структуры управляющих данных, пока не произойдет одно из следующих условий:

- процессор переведет контроллер в режим «Основной» во время цикла с альтернативной структурой
- контроллер считает «неправильную» структуру управляющих данных.

Примечание – После исполнения контроллером N передач с использованием первичных управляющих данных он делает эти управляющие данные «неправильными» путем установки cycle_ctrl в 3'b000.

Контроллер устанавливает флаг dma_done[C] в этом режиме работы только тогда, когда передача DMA заканчивается с использованием основного цикла.

В данном режиме контроллер использует первичные управляющие данные для программирования альтернативных управляющих данных. Таблица 17-19 перечисляет области памяти channel_cfg, как те, которые должны быть определены константами, так и те, значения которых определяются пользователем.

Таблица 17-19 – Channel_cfg для первичной структуры управляющих данных в режиме работы с памятью «исполнение с изменением конфигурации»

Номер бита	Обозначение	Значение	Описание
Области с константными значениями			
31...30	dst_inc	b'10	Контроллер производит инкремент адреса пословно
29...28	dst_size	b'10	Контроллер осуществляет передачу пословно
27...26	src_inc	b'10	Контроллер производит инкремент адреса пословно
25...24	src_size	b'10	Контроллер осуществляет передачу пословно
17...14	R_power	b'0010	Контроллер выполняет четыре передачи DMA
3	next_useburst	b'0	Для данного режима этот разряд должен быть равен 0
2...0	cycle_ctrl	b'100	Контролер работает в режиме работы с периферией «исполнение с изменением конфигурации»
Области со значениями, определяемыми пользователем			
23...21	dst_prot_ctrl	-	Определяет состояние HPROT при записи данных в приемник
20...18	src_prot_ctrl	-	Определяет состояние HPROT при чтении данных из источника
13...4	n_minus_1	N*	Настраивает контроллер на выполнение N передач DMA, где N кратно 4
* Так как R_power задает значение 4, то необходимо задавать значение N, кратное 4. Число, равное N/4, это количество раз, которое нужно настраивать альтернативные управляющие данные			

Рисунок 17-9 демонстрирует пример функционирования в режиме работы с памятью «Исполнение с изменением конфигурации».

Инициализация:

1 Настройка первичных управляющих данных для разрешения копирования A, B, C и D: cycle_ctrl=b'100, $2^R=4$, N=16.

2 Запись первичных данных в память с использованием структуры, показанной в таблице ниже.

	src_data_end_ptr	dst_data_end_ptr	channel_cfg	Unused
Data for Task A	0x0A000000	0x0AE00000	cycle_ctrl = b101, 2 ^R = 4, N = 3	0xFFFFFFFF
Data for Task B	0x0B000000	0x0BE00000	cycle_ctrl = b101, 2 ^R = 2, N = 8	0xFFFFFFFF
Data for Task C	0x0C000000	0x0CE00000	cycle_ctrl = b101, 2 ^R = 8, N = 5	0xFFFFFFFF
Data for Task D	0x0D000000	0x0DE00000	cycle_ctrl = b001, 2 ^R = 4, N = 4	0xFFFFFFFF

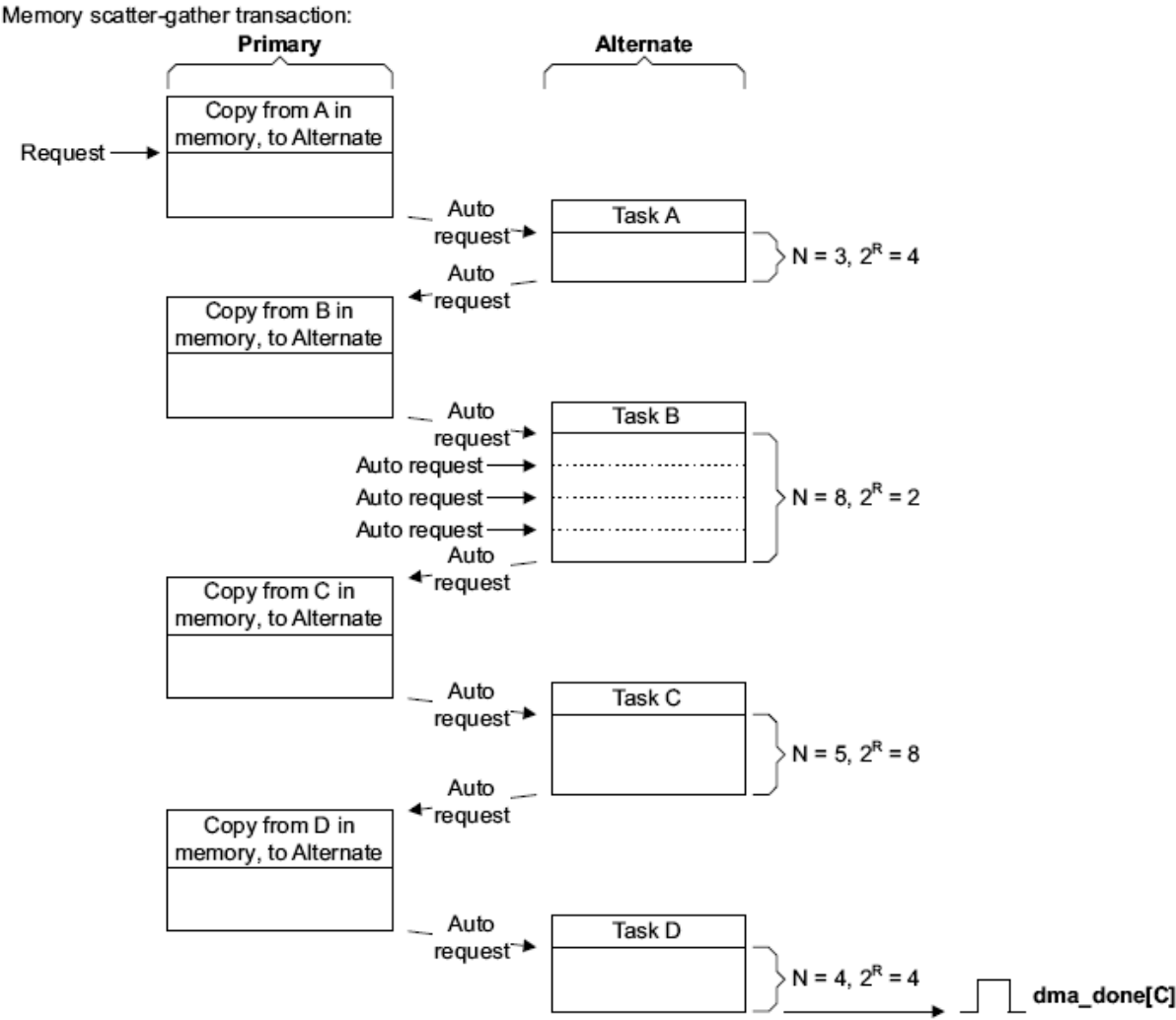


Рисунок 17-9 – Пример работы DMA в режиме «Исполнение с изменением конфигурации»

Пояснения к схеме (Рисунок 17-9):

Инициализация:

- 1 Процессор настраивает первичную структуру управляющих данных для работы в режиме работы с памятью «исполнение с изменением конфигурации» путем установки cycle_ctrl в b100. Так как управляющие данные канала состоят из четырех слов, необходимо установить 2^R в 4. В этом примере количество задач равно 4, и поэтому N установлен в 16.
- 2 Процессор записывает управляющие данные для шагов A, B, C, D в область памяти с адресом, указанным в src_data_end_ptr.
- 3 Процессор разрешает работу канала DMA.

Передачи в данном режиме начинают исполняться при получении контроллером запроса на обслуживание по `dma_req[]` или запроса от процессора. Порядок выполнения следующий:

Первичная, копирование А

По получению запроса на обслуживание контроллер выполняет четыре передачи DMA. Эти передачи записывают альтернативную структуру управляющих данных для шага А.

Контроллер генерирует автозапрос для канала, после чего проводит процедуру арбитража.

Шаг А

Контроллер выполняет шаг А. По окончании контроллер генерирует автозапрос для канала и проводит процедуру арбитража.

Первичная, копирование В

Контроллер выполняет четыре передачи DMA. Эти передачи записывают альтернативную структуру управляющих данных для шага В.

Контроллер генерирует автозапрос для канала, после чего проводит процедуру арбитража.

Шаг В

Контроллер выполняет шаг В. По окончании контроллер генерирует автозапрос для канала и проводит процедуру арбитража.

Первичная, копирование С

Контроллер выполняет четыре передачи DMA. Эти передачи записывают альтернативную структуру управляющих данных для шага С.

Контроллер генерирует автозапрос для канала, после чего проводит процедуру арбитража.

Шаг С

Контроллер выполняет шаг С. По окончании контроллер генерирует автозапрос для канала и проводит процедуру арбитража.

Первичная, копирование D

Контроллер выполняет четыре передачи DMA. Эти передачи записывают альтернативную структуру управляющих данных для шага D.

Контроллер устанавливает `cycle_ctrl` первичных управляющих данных в `b000` для индикации о том, что эта структура управляющих данных является «неправильной».

Контроллер генерирует автозапрос для канала, после чего проводит процедуру арбитража.

Шаг D

Контроллер выполняет шаг D, используя основной цикл DMA.

Контроллер устанавливает флаг `dma_done[C]` в состояние 1 на один такт сигнала `hclk` и входит в процедуру арбитража.

17.4.5.6 Режим работы с периферией «исполнение с изменением конфигурации»

В данном режиме контроллер, получая начальный запрос на обработку, выполняет четыре передачи DMA, используя первичные управляющие данные. По окончании этих передач контроллер начинает цикл DMA, используя альтернативные управляющие данные без осуществления арбитража и не устанавливая сигнал dma_active[C] в 0.

Примечание – Это единственный случай, при котором контроллер не осуществляет процедуру арбитража после выполнения передачи DMA, используя первичные управляющие данные.

После того, как этот цикл завершился, контроллер выполняет арбитраж и по получении запроса на обслуживание от периферии, имеющего наивысший приоритет, он выполняет еще четыре передачи DMA, используя первичные управляющие данные. По окончании этих передач контроллер начинает цикл DMA, используя альтернативные управляющие данные без осуществления арбитража и не устанавливая сигнал dma_active[C] в 0.

Контроллер продолжает выполнять циклы ПДА, меняя структуры управляющих данных, пока не произойдет одно из следующих условий:

процессор переведет контроллер в режим «Основной» во время цикла с альтернативной структурой;

контроллер считает «неправильную» структуру управляющих данных.

Примечание – После исполнения контроллером N передач с использованием первичных управляющих данных, он делает эти управляющие данные «неправильными» путем установки cycle_ctrl в 3'b000.

Контроллер устанавливает флаг dma_done[C] в этом режиме работы только тогда, когда передача DMA заканчивается с использованием основного цикла.

В данном режиме контроллер использует первичные управляющие данные для программирования альтернативных управляющих данных. Таблица 17-20 перечисляет области памяти channel_cfg, как те, которые должны быть определены константами, так и те, значения которых определяются пользователем.

Таблица 17-20 – Channel_cfg для первичной структуры управляющих данных в режиме работы с периферией «Исполнение с изменением конфигурации»

Номер бита	Обозначение	Значение	Описание
Области с константными значениями			
31...30	dst_inc	b'10	Контроллер производит инкремент адреса пословно
29...28	dst_size	b'10	Контроллер осуществляет передачу пословно
27...26	src_inc	b'10	Контроллер производит инкремент адреса пословно
25...24	src_size	b'10	Контроллер осуществляет передачу пословно
17...14	R_power	b'0010	Контроллер выполняет четыре передачи DMA
2...0	cycle_ctrl	b'110	Контролер работает в режиме работы с периферией «исполнение с изменением конфигурации»

Номер бита	Обозначение	Значение	Описание
Области со значениями, определяемыми пользователем			
23...21	dst_prot_ctrl	-	Определяет состояние HPROT при записи данных в приемник
20...18	src_prot_ctrl	-	Определяет состояние HPROT при чтении данных из источника
13...4	n_minus_1	N*)	Настраивает контроллер на выполнение N передач DMA, где N кратно 4
3	next_useburst	-	При установке в 1 контроллер установит chnl_useburst_set[C] в 1 после выполнения передачи с альтернативной структурой
* Так как R_power задает значение 4, то необходимо задавать значение N, кратное 4. Число, равное N/4, это количество раз, которое нужно настраивать альтернативные управляющие данные			

Следующий рисунок демонстрирует пример функционирования в режиме работы с периферией «исполнение с изменением конфигурации».

Инициализация:

1. Настройка первичных управляющих данных для разрешения копирования A, B, C и D: cycle_ctrl=b110, $2^R=4$, N=16.
2. Запись первичных данных в память с использованием структуры, показанной в таблице ниже.

	src_data_end_ptr	dst_data_end_ptr	channel_cfg	Unused
Data for Task A	0x0A000000	0x0AE00000	cycle_ctrl = b111, 2 ^R = 4, N = 3	0xFFFFFFFF
Data for Task B	0x0B000000	0x0BE00000	cycle_ctrl = b111, 2 ^R = 2, N = 8	0xFFFFFFFF
Data for Task C	0x0C000000	0x0CE00000	cycle_ctrl = b111, 2 ^R = 8, N = 5	0xFFFFFFFF
Data for Task D	0x0D000000	0x0DE00000	cycle_ctrl = b001, 2 ^R = 4, N = 4	0xFFFFFFFF

Peripheral scatter-gather transaction:

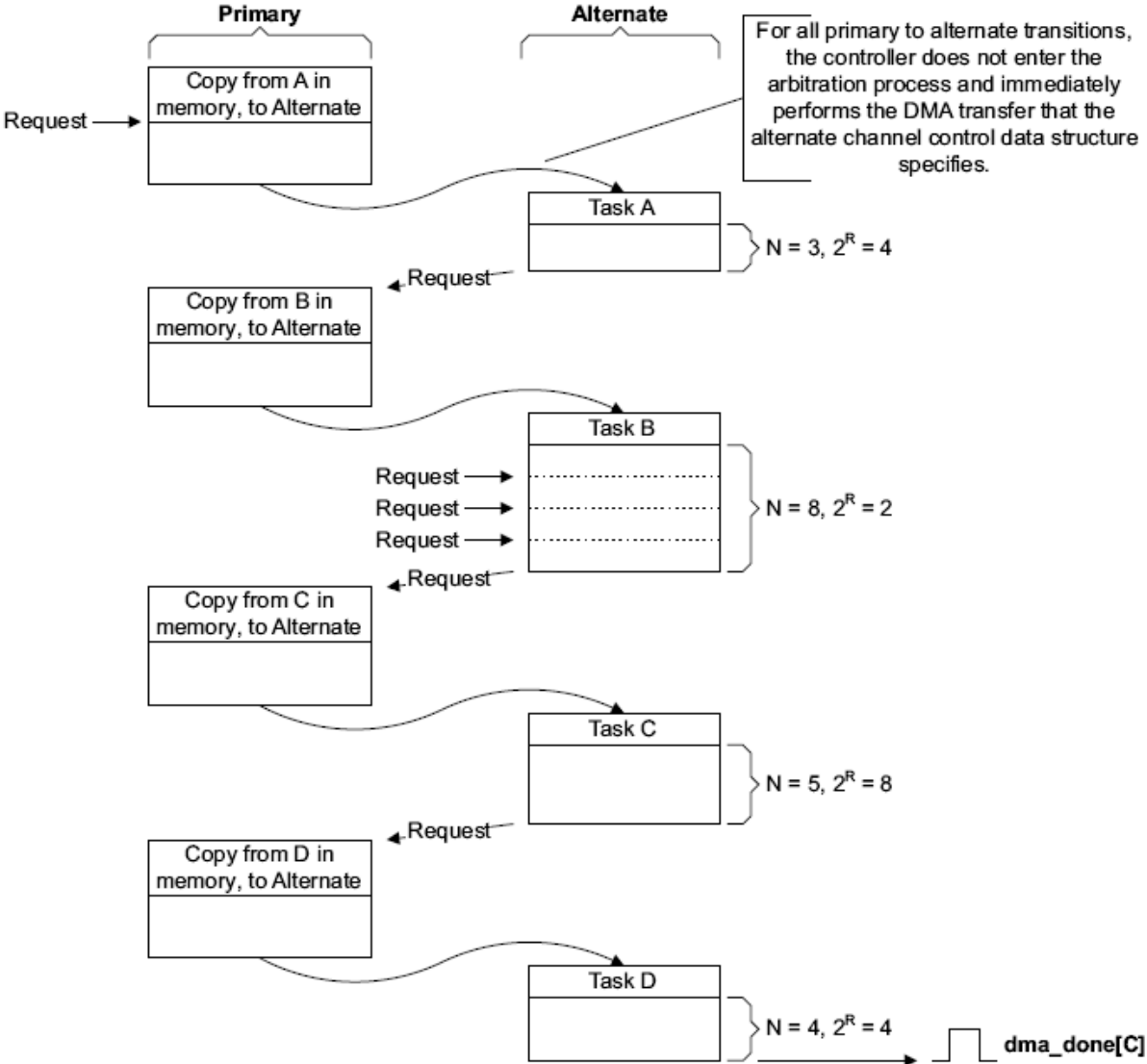


Рисунок 17-10 – Пример работы DMA в режиме с «Исполнением с изменением конфигурации»

Пояснения к схеме (Рисунок 17-10):

Инициализация:

1 Процессор настраивает первичную структуру управляющих данных для работы в режиме работы с периферией «исполнение с изменением конфигурации» путем установки cycle_ctrl в b110. Так как управляющие данные канала состоят из четырех слов, необходимо установить 2^R в 4. В этом примере количество задач равно 4, и поэтому N установлен в 16.

2 Процессор записывает управляющие данные для шагов A, B, C, D в область памяти с адресом, указанным в src_data_end_ptr.

3 Процессор разрешает работу канала DMA.

Передачи в данном режиме начинают исполняться при получении контроллером запроса на обслуживание по dma_req[]. Передачи выполняются следующим образом:

Первичная, копирование из области А памяти

По получению запроса на обслуживание, контроллер выполняет четыре передачи DMA. Эти передачи записывают альтернативную структуру управляющих данных для шага А.

Шаг А

Контроллер выполняет шаг А.

По окончании контроллер проводит процедуру арбитража.

Первичная, копирование из области В памяти

Контроллер выполняет четыре передачи DMA. Эти передачи записывают альтернативную структуру управляющих данных для шага В.

Шаг В

Контроллер выполняет шаг В. Для завершения задачи периферия должна установить последовательно три запроса.

По окончании контроллер проводит процедуру арбитража.

Первичная, копирование из области С памяти

Контроллер выполняет четыре передачи DMA. Эти передачи записывают альтернативную структуру управляющих данных для шага С.

Шаг С

Контроллер выполняет шаг С.

По окончании контроллер проводит процедуру арбитража.

После выставления периферией нового запроса на обслуживание, при условии, что этот запрос является наиболее приоритетным, процесс продолжается следующим образом:

Первичная, копирование из области D памяти

Контроллер выполняет четыре передачи DMA. Эти передачи записывают альтернативную структуру управляющих данных для шага D.

Контроллер устанавливает `cycle_ctrl` первичных управляющих данных в `b000` для индикации о том, что эта структура управляющих данных является «неправильной».

Шаг D

Контроллер выполняет шаг D, используя основной цикл DMA.

Контроллер устанавливает флаг `dma_done[C]` в состояние 1 на один такт сигнала `hclk` и входит в процедуру арбитража.

17.4.6 Индикация ошибок

При получении контроллером по шине АНВ ответа об ошибке, он выполняет следующие действия:

- 1 отключает канал, связанный с ошибкой;
- 2 устанавливает флаг dma_err в состояние 1.

После обнаружения процессором флага dma_err процессор определяет номер канала, который был активен в момент появления ошибки. Для этого он осуществляет следующее:

- чтение регистра chnl_enable_set с целью создания списка отключенных каналов;
- если канал установил флаг dma_done[], то контроллер отключает канал. Программа, выполняемая процессором, должна всегда хранить данные о каналах, которые недавно установили флаги dma_done[];
- процессор должен сравнить список выключенных каналов, полученный в шаге 1, с данными о каналах, которые недавно устанавливали флаги dma_done[]. Канал, по которому отсутствуют данные об установке флага dma_done[], это и есть канал, с которым связана ошибка.

17.5 Структура управляющих данных канала

В системной памяти должна быть отведена область для хранения управляющих данных каналов. Системная память должна:

- предоставлять смежную область системной памяти, к которой контроллер и процессор имеют доступ;
- иметь базовый адрес, который целочисленно кратен общему размеру структуры управляющих данных канала.

Рисунок 17-11 показывает область памяти, необходимую контроллеру для структур управляющих данных канала, при использовании всех 32 каналов и опциональной альтернативной структуры управляющих данных.

Alternate data structure		Primary data structure	
Alternate_Ch_31	0x3F0	Primary_Ch_31	0x1F0
Alternate_Ch_30	0x3E0	Primary_Ch_30	0x1E0
Alternate_Ch_29	0x3D0	Primary_Ch_29	0x1D0
Alternate_Ch_28	0x3C0	Primary_Ch_28	0x1C0
Alternate_Ch_27	0x3B0	Primary_Ch_27	0x1B0
Alternate_Ch_26	0x3A0	Primary_Ch_26	0x1A0
Alternate_Ch_25	0x390	Primary_Ch_25	0x190
Alternate_Ch_24	0x380	Primary_Ch_24	0x180
Alternate_Ch_23	0x370	Primary_Ch_23	0x170
Alternate_Ch_22	0x360	Primary_Ch_22	0x160
Alternate_Ch_21	0x350	Primary_Ch_21	0x150
Alternate_Ch_20	0x340	Primary_Ch_20	0x140
Alternate_Ch_19	0x330	Primary_Ch_19	0x130
Alternate_Ch_18	0x320	Primary_Ch_18	0x120
Alternate_Ch_17	0x310	Primary_Ch_17	0x110
Alternate_Ch_16	0x300	Primary_Ch_16	0x100
Alternate_Ch_15	0x2F0	Primary_Ch_15	0x0F0
Alternate_Ch_14	0x2E0	Primary_Ch_14	0x0E0
Alternate_Ch_13	0x2D0	Primary_Ch_13	0x0D0
Alternate_Ch_12	0x2C0	Primary_Ch_12	0x0C0
Alternate_Ch_11	0x2B0	Primary_Ch_11	0x0B0
Alternate_Ch_10	0x2A0	Primary_Ch_10	0x0A0
Alternate_Ch_9	0x290	Primary_Ch_9	0x090
Alternate_Ch_8	0x280	Primary_Ch_8	0x080
Alternate_Ch_7	0x270	Primary_Ch_7	0x070
Alternate_Ch_6	0x260	Primary_Ch_6	0x060
Alternate_Ch_5	0x250	Primary_Ch_5	0x050
Alternate_Ch_4	0x240	Primary_Ch_4	0x040
Alternate_Ch_3	0x230	Primary_Ch_3	0x030
Alternate_Ch_2	0x220	Primary_Ch_2	0x020
Alternate_Ch_1	0x210	Primary_Ch_1	0x010
Alternate_Ch_0	0x200	Primary_Ch_0	0x000

Unused	0x00C
Control	0x008
Destination End Pointer	0x004
Source End Pointer	0x000

Рисунок 17-11 – Карта памяти для 32-х каналов, включая альтернативную структуру

Пример, приведенный выше (Рисунок 17-11), использует 1 Кбайт системной памяти. В этом примере контроллер использует младшие 0x10 разрядов адреса для доступа ко всем элементам структуры управляющих данных, и поэтому базовый адрес структуры должен быть 0xXXXXX000, далее 0xXXXXX400, далее 0xXXXXX800, далее 0xXXXXXC00.

Возможно, установить базовый адрес для первичной структуры управляющих данных путем записи соответствующего значения в регистр ctrl_base_ptr.

Необходимый размер области системной памяти зависит:

- от количества каналов, используемых в контроллере;
- от того, используется или нет альтернативная структура управляющих данных.

Таблица 17-21 перечисляет разряды адреса, обеспечивающие контроллеру доступ к различным элементам структуры управляющих данных, в зависимости от количества каналов, используемых в контроллере.

Таблица 17-21 – Разряды адреса, соответствующие элементам структуры управляющих данных

	Разряды адреса						
Количество каналов, используемых в контроллере	[9]	[8]	[7]	[6]	[5]	[4]	[3:0]
1						A	0x0 0x4 0x8
2					A	C[0]	
3-4				A	C[1]	C[0]	
5-8			A	C[2]	C[1]	C[0]	
9-16		A	C[3]	C[2]	C[1]	C[0]	
17-32	A	C[4]	C[3]	C[2]	C[1]	C[0]	

Где A выбирает одну из структур управляющих данных канала:
A = 0 выбирает первичную структуру управляющих данных;
A = 1 выбирает альтернативную структуру управляющих данных.

C[x:0] Выбирает канал DMA.

Address[3:0] Выбирает один из управляющих элементов:
0x0 выбирает указатель конца данных источника;
0x4 выбирает указатель конца данных приемника;
0x8 выбирает конфигурацию управляющих данных;
0xC контроллер не имеет доступа к этому адресу. Если это необходимо, то возможно разрешить процессору использовать эти адреса в качестве системной памяти.

Примечание – Совсем не обязательно вычислять базовый адрес альтернативной структуры управляющих данных, так как регистр alt_ctrl_base_ptr содержит эту информацию

Рисунок 17-12 демонстрирует пример реализации контроллера с использованием трех каналов DMA и с альтернативной структурой управляющих данных

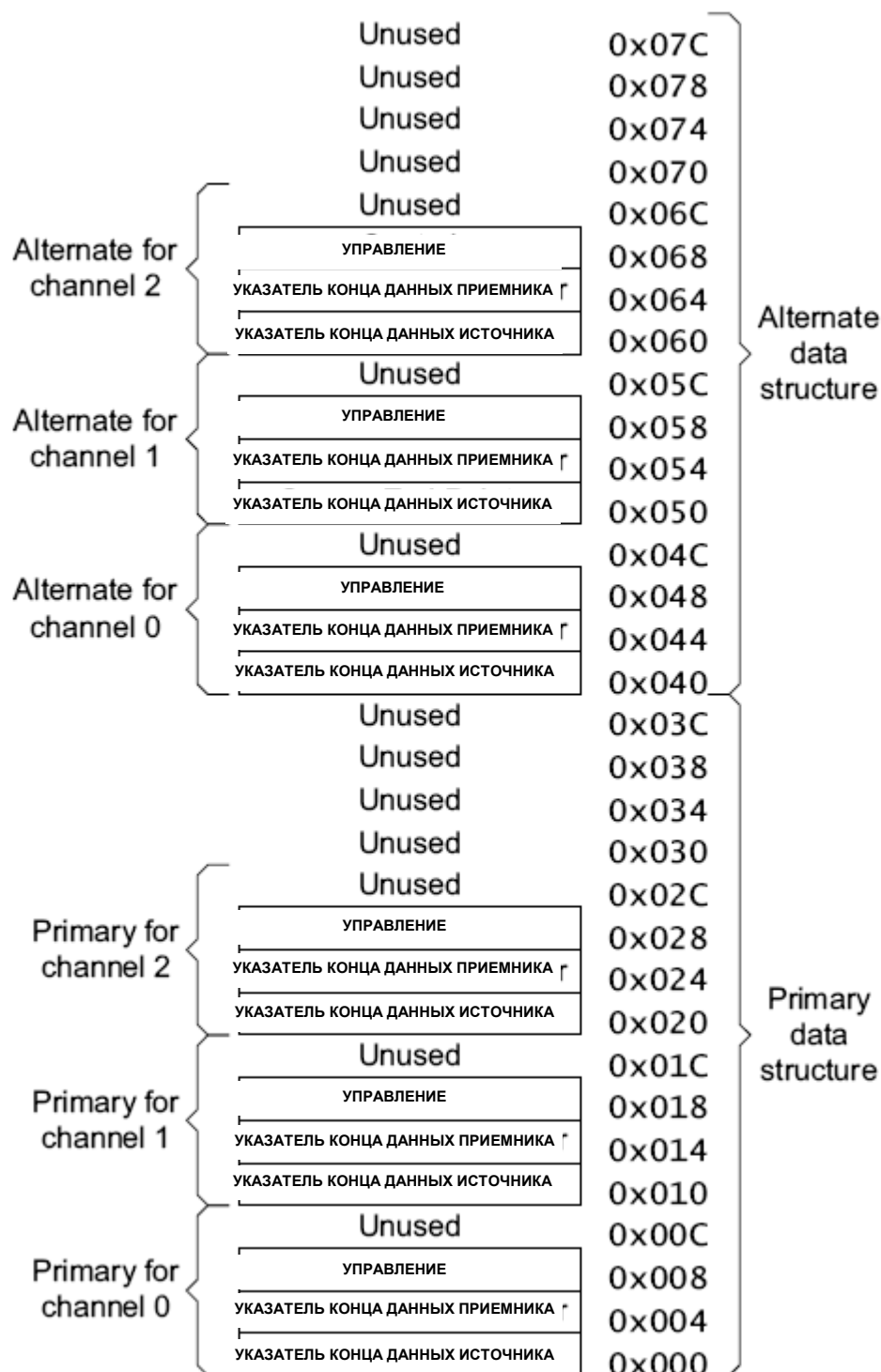


Рисунок 17-12 – Карта памяти для трех каналов DMA, включая альтернативную структуру

Этот пример структуры управляющих данных использует 128 байт системной памяти. В нем контроллер использует младшие 0x06 разрядов адреса для доступа ко всем элементам структуры управляющих данных. Поэтому базовый адрес структуры должен быть 0xXXXXXX00, далее 0xXXXXXX80.

Таблица 17-22 перечисляет все разрешенные значения базового адреса для первичной структуры управляющих данных в зависимости от количества каналов DMA, использованных в контроллере.

Таблица 17-22 – Разрешенные базовые адреса

Количество каналов DMA	Разрешенные значения базового адреса для первичной структуры управляющих данных
17-32	0хXXXXXX000, 0хXXXXXX400, 0хXXXXXX800, 0хXXXXXXC00

Контроллер использует системную память для доступа к двум указателям адреса конца данных и разрядам управления каждого канала. Эти 32-разрядные области памяти и процедуру вычисления контроллером адреса передачи DMA описывают следующие подразделы:

- указатель конца данных источника;
- указатель конца данных приемника;
- разряды управления;
- вычисление адреса.

Указатель конца данных источника

Область памяти под названием `src_data_end_ptr` содержит указатель на последний адрес месторасположения данных источника.

Таблица 17-23 – Значения разрядов `src_data_end_ptr`

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...0	<code>src_data_end_ptr</code>	Указатель последнего адреса данных источника

Перед тем, как контроллер выполнит передачу DMA, необходимо определить эту область памяти. Контроллер считывает значение этой области перед началом 2^R передачи DMA.

Примечания:

- 1 Контроллер не имеет доступа по записи в эту область памяти.
 - 2 Значение записываемое в этот регистр должно быть кратно разрядности транзакций.
- (Для 32-разрядных передач младшие два байта – нули, для 16-разрядных передач младший байт – ноль, в случае байтовых передач возможен любой адрес).

Указатель конца данных приемника

Область памяти под названием `dst_data_end_ptr` содержит указатель на последний адрес месторасположения данных приемника.

Таблица 17-24 – Значения разрядов `dst_data_end_ptr`

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...0	<code>dst_data_end_ptr</code>	Указатель на последний адрес данных приемника

Перед тем, как контроллер выполнит передачу DMA, необходимо определить эту область памяти. Контроллер считывает значение этой области перед началом 2^R передачи DMA.

Примечания:

1 Контроллер не имеет доступа по записи в эту область памяти.

2 Значение записываемое в этот регистр должно быть кратно разрядности транзакций.
(Для 32-разрядных передач младшие два байта – нули, для 16-разрядных передач младший байт – ноль, в случае байтовых передач возможен любой адрес).

Разряды управления

Область памяти под названием channel_cfg обеспечивает управление каждой передачей DMA.

Таблица 17-25 – Название разрядов области памяти channel_cfg

Номер	31	30	29	28	27	26	25	24	23...21	20...18	17...14	13...4	3	2...0
Доступ														
Сброс														
	dst_inc		dst_size		src_inc		src_size		dst_prot_ctrl	Src_prot_ctrl	R_power	n_minus_1	next_useburst	cycle_ctrl

Таблица 17-26 объясняет назначение разрядов этой области памяти.

Таблица 17-26 – Назначение разрядов channel_cfg

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31, 30	dst_src	Шаг инкремента адреса приемника. Шаг инкремента адреса зависит от разрядности данных источника. Разрядность данных источника = байт: b00 = байт; b01 = полуслово (16 разрядов); b10 = слово (32 разряда); b11 = нет инкремента. Адрес остается равным значению области памяти dst_data_end_ptr. Разрядность данных источника = полуслово: b00 = зарезервировано; b01 = полуслово; b10 = слово; b11 = нет инкремента. Адрес остается равным значению области памяти dst_data_end_ptr. Разрядность данных источника = слово: b00 = зарезервировано; b01 = зарезервировано; b10 = слово (32 разряда); b11 = нет инкремента. Адрес остается равным значению области памяти dst_data_end_ptr

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
29, 28	dst_size	Размерность данных приемника Примечание – Значение этого поля должно быть равно значению поля src_size
27, 26	src_inc	Шаг инкремента адреса источника. Шаг инкремента адреса зависит от разрядности данных источника. Разрядность данных источника = байт: b00 = байт; b01 = полуслово; b10 = слово (32 разряда); b11 = нет инкремента. Адрес остается равным значению области памяти src_data_end_ptr. Разрядность данных источника = полуслово: b00 = зарезервировано b01 = полуслово b10 = слово b11 = нет инкремента. Адрес остается равным значению области памяти src_data_end_ptr. Разрядность данных источника = слово: b00 = зарезервировано; b01 = зарезервировано; b10 = слово; b11 = нет инкремента. Адрес остается равным значению области памяти src_data_end_ptr
25, 24	src_size	Задаёт размерность данных источника: b00 = байт; b01 = полуслово (в русском обычно слово); b10 = слово (в русском обычно двойное слово); b11 = зарезервировано
23, 21	dst_prot_ctrl	Задаёт состояние HPROT[3:1], когда контроллер записывает данные в приемник. Разряд 23 управляет разрядом HPROT[3]: 0 = HPROT[3] в состоянии 0 и доступ не кэшируется; 1 = HPROT[3] в состоянии 1 и доступ кэшируется. Разряд 22 управляет разрядом HPROT[2]: 0 = HPROT[2] в состоянии 0 и доступ не буферизуется; 1 = HPROT[2] в состоянии 1 и доступ буферизуется. Разряд 21 управляет разрядом HPROT[1]: 0 = HPROT[1] в состоянии 0 и доступ непривилегированный; 1 = HPROT[1] в состоянии 1 и доступ привилегированный
20...18	src_prot_ctrl	Задаёт состояние HPROT[3:1], когда контроллер считывает данные из источника. Разряд 20 управляет разрядом HPROT[3]: 0 = HPROT[3] в состоянии 0 и доступ не кэшируется; 1 = HPROT[3] в состоянии 1 и доступ кэшируется.

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
		Разряд 19 управляет разрядом HPROT[2]: 0 = HPROT[2] в состоянии 0 и доступ не буферизуется; 1 = HPROT[2] в состоянии 1 и доступ буферизуется. Разряд 18 управляет разрядом HPROT[1]: 0 = HPROT[1] в состоянии 0 и доступ непривилегированный; 1 = HPROT[1] в состоянии 1 и доступ привилегированный
17...14	R_power	Задаёт количество передач DMA до выполнения контроллером процедуры арбитража. Возможные значения: b0000 – арбитраж производится после каждой передачи DMA; b0001 – арбитраж производится после 2 передач DMA; b0010 – арбитраж производится после 4 передач DMA; b0011 – арбитраж производится после 8 передач DMA; b0100 – арбитраж производится после 16 передач DMA; b0101 – арбитраж производится после 32 передач DMA; b0110 – арбитраж производится после 64 передач DMA; b0111 – арбитраж производится после 128 передач DMA; b1000 – арбитраж производится после 256 передач DMA; b1001 – арбитраж производится после 512 передач DMA; b1010 – b1111 – арбитраж производится после 1024 передач DMA. Это означает, что арбитраж не производится, так как максимальное количество передач DMA равно 1024
13...4	n_minus_1	Перед выполнением цикла DMA эти разряды указывают общее количество передач DMA, из которых состоит цикл DMA. Необходимо установить эти разряды в значение, соответствующее размеру желаемого цикла DMA. 10-разрядное число плюс 1 задаёт количество передач DMA. Возможные значения: b0000000000 = 1 передача DMA; b0000000001 = 2 передачи DMA; b0000000010 = 3 передачи DMA; b0000000011 = 4 передачи DMA; b0000000100 = 5 передач DMA; b0000000101 = 6 передач DMA; b1111111111 = 1024 передачи DMA. Контроллер обновит это поле перед тем, как произвести процесс арбитража. Это позволяет контроллеру хранить количество оставшихся передач DMA до завершения цикла DMA
3	next_useburst	Контролирует, не установлен ли chnl_useburst_set[C] в состояние 1, если контроллер работает в режиме работы с периферией «Исполнение с изменением конфигурации» и если контроллер завершает цикл DMA, используя альтернативные управляющие данные.

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
		Примечание – Перед завершением цикла DMA, использующего альтернативные управляющие данные, контроллер устанавливает <code>chnl_useburst_set[C]</code> в значение 0, если количество оставшихся передач DMA меньше, чем 2^R. Установка <code>next_useburst</code> разряда определяет, будет ли контроллер дополнительно переопределять разряд <code>chnl_useburst_set[C]</code>. Если контроллер выполняет цикл DMA в режиме работы с периферией «Исполнение с изменением конфигурации», то после окончания цикла, использующего альтернативные управляющие данные, происходит следующее в зависимости от состояния <code>next_useburst</code>: 0 – контроллер не изменяет значение <code>chnl_useburst_set[C]</code>. Если <code>chnl_useburst_set[C]</code> установлен в 0, то для всех оставшихся циклов DMA в режиме работы с периферией «Исполнение с изменением конфигурации», контроллер отвечает на запросы по <code>dma_req[]</code> и <code>dma_sreq[]</code>, при выполнении циклов DMA он использует альтернативные управляющие данные. 1 – контроллер изменяет значение <code>chnl_useburst_set[C]</code> в состояние 1. Поэтому для оставшихся циклов DMA в режиме работы с периферией «Исполнение с изменением конфигурации», контроллер реагирует только на запросы по <code>dma_req[]</code>, при выполнении циклов DMA он использует альтернативные управляющие данные
2...0	<code>cycle_ctrl</code>	Режим работы при выполнении цикла DMA: b000 – Стоп. Означает, что структура управляющих данных является «неправильной»; b001 – Основной. Контроллер должен получить новый запрос для окончания цикла DMA, перед этим он должен выполнить процедуру арбитража; b010 – Авто-запрос. Контроллер автоматически осуществляет запрос на обработку по соответствующему каналу в течение процедуры арбитража. Это означает, что начального запроса на обработку достаточно для выполнения цикла DMA; b011 – Пинг-понг. Контроллер выполняет цикл DMA используя одну из структур управляющих данных. По окончании выполнения цикла DMA, контроллер выполняет следующий цикл DMA, используя другую структуру. Контроллер сигнализирует об окончании каждого цикла DMA, позволяя процессору перенастраивать неактивную структуру данных. Контроллер продолжает выполнять циклы DMA, до тех пор, пока он не прочитает «неправильную» структуру данных или пока процессор не изменит <code>cycle_ctrl</code> поле в состояние b001 или b010;

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
		b100 – Режим работы с памятью «Исполнение с изменением конфигурации». Смотрите соответствующий раздел. При работе контроллера в данном режиме значение этого поля в первичной структуре управляющих данных должно быть b100; b101 – Режим работы с памятью «Исполнение с изменением конфигурации». Смотрите соответствующий раздел. При работе контроллера в данном режиме значение этого поля в альтернативной структуре управляющих данных должно быть b101; b110 – Режим работы с периферией «исполнение с изменением конфигурации». Смотрите соответствующий раздел. При работе контроллера в данном режиме значение этого поля в первичной структуре управляющих данных должно быть b110; b111 – Режим работы с периферией «исполнение с изменением конфигурации». Смотрите соответствующий раздел. При работе контроллера в данном режиме значение этого поля в альтернативной структуре управляющих данных должно быть b111

В начале цикла DMA или 2^R передачи DMA контроллер считывает значение `channel_cfg` из системной памяти. После выполнения 2^R или N передач он сохраняет обновленное значение `channel_cfg` в системную память.

Контроллер не поддерживает значений `dst_size`, отличных от значений `src_size`. Если контроллер обнаруживает неравные значения этих полей, он использует значение `src_size` в качестве размера данных и приемника, и источника и при ближайшем обновлении поля `n_minus_1`, он также устанавливает значение поля `dst_size`, равное `src_size`.

После выполнения контроллером N передач, контроллер устанавливает значение поля `cycle_ctrl` в b000, делая тем самым `channel_cfg` данные «неправильными». Это позволяет избежать повторения выполненной передачи DMA.

Вычисление адреса

Для вычисления адреса источника передачи DMA, контроллер выполняет сдвиг влево значения `n_minus_1` на количество разрядов, соответствующее полю `src_inc`, и затем вычитает получившееся значение от значения указателя адреса конца данных источника. Подобным образом вычисляется адрес передатчика передачи DMA, контроллер выполняет сдвиг влево значения `n_minus_1` на количество разрядов, соответствующее полю `dst_inc`, и затем вычитает получившееся значение от значения указателя адреса конца данных приемника.

В зависимости от значения полей `src_inc` и `dst_inc` вычисления адресов приемника и источника выполняются по следующим уравнениям:

src_inc=b00 and dst_inc=b00

адрес источника = src_data_end_ptr - n_minus_1

адрес приемника = dst_data_end_ptr - n_minus_1.

src_inc=b01 and dst_inc=b01

адрес источника = src_data_end_ptr - (n_minus_1<<1)

адрес приемника = dst_data_end_ptr - (n_minus_1<<1).

src_inc=b01 and dst_inc=b10

адрес источника = src_data_end_ptr - (n_minus_1<<2)

адрес приемника = dst_data_end_ptr - (n_minus_1<<2).

src_inc=b11 and dst_inc=b11

- адрес источника = src_data_end_ptr

- адрес приемника = dst_data_end_ptr.

Таблица 17-27 перечисляет адреса приемника цикла DMA для шести слов.

Таблица 17-27 – Цикла DMA для шести слов с пословным инкрементом

Начальные значения channel_cfg перед циклом DMA				
src_size=b10, dst_inc=b10, n_minus_1=b101, cycle_ctrl=1				
Передачи DMA	Указатель конца данных	Счетчик	Отличие*	Адрес
	0x2AC	5	0x14	0x298
	0x2AC	4	0x10	0x29C
	0x2AC	3	0xC	0x2A0
	0x2AC	2	0x8	0x2A4
	0x2AC	1	0x4	0x2A8
	0x2AC	0	0x0	0x2AC
Конечные значения channel_cfg после цикла DMA				
src_size=b10, dst_inc=b10, n_minus_1=0, cycle_ctrl=0				
* Это значение, полученное после сдвига влево значения счетчика на количество разрядов, соответствующее dst_inc				

Таблица 17-28 перечисляет адреса приемника для передач DMA 12 байт с использованием «полусловного» инкремента.

Таблица 17-28 – Цикла DMA для 12 байт с «полусловным» инкрементом

Начальные значения channel_cfg перед циклом DMA				
src_size=b00, dst_inc=b01, n_minus_1=b1011, cycle_ctrl=1, R_power=b11				
DMA передачи	Указатель конца данных	Счетчик	Отличие*	Адрес
	0x5E7	11	0x16	0x5D1
	0x5E7	10	0x14	0x5D3
	0x5E7	9	0x12	0x5D5
	0x5E7	8	0x10	0x5D7
	0x5E7	7	0xE	0x5D9
	0x5E7	6	0xC	0x5DB
	0x5E7	5	0xA	0x5DD
	0x5E7	4	0x8	0x5DF

Значения channel_cfg после 2 ^R передач DMA				
src_size=b00, dst_inc=b01, n_minus_1=b011, cycle_ctrl=1, R_power=b11				
DMA передачи	0x5E7	3	0x6	0x5E1
	0x5E7	2	0x4	0x5E3
	0x5E7	1	0x2	0x5E5
	0x5E7	0	0x0	0x5E7
Конечные значения channel_cfg после цикла DMA				
src_size=b00, dst_inc=b01, n_minus_1=0, cycle_ctrl=0**, R_power=b11				
* Это значение, полученное после сдвига влево значения счетчика на количество разрядов, соответствующее dst_inc. ** После окончания цикла DMA контроллер делает channel_cfg «неправильным», сбрасывая в 0 поле cycle_ctrl				

17.6 Описание регистров контроллера DMA

Данный раздел описывает регистры контроллера и управление контроллером через них.

Раздел содержит следующие сведения:

- о регистровой модели контроллера;
- описание регистров.

Основные положения регистровой модели контроллера:

- нужно избегать адресации при доступе к зарезервированным или неиспользованным адресам, так как это может привести к непредсказуемым результатам;
- необходимо заполнять неиспользуемые или зарезервированные разряды регистров нулями при записи и игнорировать значения таких разрядов при считывании, кроме случаев, специально описанных в разделе;
- системный сброс или сброс по установке питания сбрасывает все регистры в состояние 0, кроме случаев, специально описанных в разделе;
- все регистры поддерживают доступ по чтению и записи, кроме случаев, специально описанных в разделе. Доступ по записи обновляет содержание регистра, а доступ по чтению возвращает содержимое регистра.

Таблица 17-29 – Перечень регистров контроллера

Смещение отн. базового адреса	Наименование	Тип	Значение по сбросу	Описание
0x40028000	MDR_DMA			Контроллер DMA
0x000	STATUS	RO	0x-0nn0000*	MDR_DMA->STATUS Статусный регистр DMA
0x004	CFG	WO	-	MDR_DMA->CFG Регистр конфигурации DMA
0x008	CTRL_BASE_PTR	R/W	0x00000000	MDR_DMA->CTRL_BASE_PTR Регистр базового адреса управляющих данных каналов

Смещение отн. базового адреса	Наименование	Тип	Значение по сбросу	Описание
0x00C	ALT_CTRL_ BASE_PTR	RO	0x000000nn**	MDR_DMA-> ALT_CTRL_BASE_PTR Регистр базового адреса альтер- нативных управляющих данных каналов
0x010	WAITONREQ_ STATUS	RO	0x00000000	MDR_DMA-> WAITONREQ_STATUS Регистр статуса ожидания запроса на обработку каналов
0x014	CHNL_SW_ REQUEST	WO	-	MDR_DMA->CHNL_SW_REQUEST Регистр программного запроса на обработку каналов
0x018	CHNL_ USEBURST_SET	R/W	0x00000000	MDR_DMA-> CHNL_USEBURST_SET Регистр установки пакетного обмена каналов
0x01C	CHNL_ USEBURST_CLR	WO	-	MDR_DMA-> CHNL_USEBURST_CLR Регистр сброса пакетного обмена каналов
0x020	CHNL_REQ_ MASK_SET	R/W	0x00000000	MDR_DMA-> CHNL_REQ_MASK_SET Регистр маскирования запросов на обслуживание каналов
0x024	CHNL_REQ_ MASK_CLR	WO	-	MDR_DMA-> CHNL_REQ_MASK_CLR Регистр очистки маскирования запросов на обслуживание каналов
0x028	CHNL_ ENABLE_SET	R/W	0x00000000	MDR_DMA->CHNL_ENABLE_SET Регистр установки разрешения каналов
0x02C	CHNL_ ENABLE_CLR	WO	-	MDR_DMA->CHNL_ENABLE_CLR Регистр сброса разрешения каналов
0x030	CHNL_PRI_ ALT_SET	R/W	0x00000000	MDR_DMA->CHNL_PRI_ALT_SET Регистр установки первичной/ альтернативной структуры управляю- щих данных каналов
0x034	CHNL_PRI_ ALT_CLR	WO	-	MDR_DMA->CHNL_PRI_ALT_CLR Регистр сброса первичной/ альтернативной структуры управляю- щих данных каналов

Смещение отн. базового адреса	Наименование	Тип	Значение по сбросу	Описание
0x038	CHNL_ PRIORITY_SET	R/W	0x00000000	DFMDR_DMA-> CHNL_PRIORITY_SET Регистр установки приоритета каналов
0x03C	CHNL_ PRIORITY_CLR	WO	-	MDR_DMA->CHNL_PRIORITY_CLR Регистр сброса приоритета каналов
0x040-0x048	-		-	Зарезервировано
0x04C	ERR_CLR	R/W	0x00000000	MDR_DMA->ERR_CLR Регистр сброса флага ошибки
0x050-0xDFC	-	-		Зарезервировано
* Значение по сбросу зависит от количества каналов DMA, использованных в контроллере, а также от того, интегрирована ли схема тестирования. ** Значение по сбросу зависит от количества каналов DMA, использованных в контроллере				

17.6.1 MDR_DMA->STATUS

Статусный регистр DMA

Данный регистр имеет доступ только на чтение. При чтении регистр возвращает состояние контроллера. Если контроллер находится в состоянии сброса, то чтение регистра запрещено. Таблица 17-31 перечисляет назначение разрядов регистра.

Таблица 17-30 – Статусный регистр DMA

Номер	31...28	27...21	20...16	15...8	7...4	3...1	0
Доступ	RO	U	RO	U	RO	U	RO
Сброс	0	0	0	0	0	0	0
	test_status	-	chnls_minus1	-	state	-	master_enable

Таблица 17-31 – Назначение разрядов регистра dma_status

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...28	test_status	Значение при чтении: 0x0 = контроллер не имеет интегрированной схемы тестирования; 0x1 = контроллер имеет интегрированную схему тестирования; 0x2 = 0xF = не определено
27...21	-	Не определено
20...16	chnls_minus1	Количество доступных каналов DMA минус 1. Например: b00000 = контроллер имеет 1 канал DMA; b00001 = контроллер имеет 2 канала DMA; b00010 = контроллер имеет 3 канала DMA; ... b11111 = контроллер имеет 32 канала DMA

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15...8	-	Не определено
7...4	state	Текущее состояние автомата управления контроллера. Состояние может быть одним из следующих: b0000 = в покое; b0001 = чтение управляющих данных канала; b0010 = чтение указателя конца данных источника; b0011 = чтение указателя конца данных приемника; b0100 = чтение данных источника; b0101 = запись данных в приемник; b0110 = ожидание запроса на выполнение DMA; b0111 = запись управляющих данных канала; b1000 = приостановлен; b1001 = выполнен; b1010 = режим работы с периферией «Исполнение с изменением конфигурации»; b1011-b1111 = не определено
3...1	-	Не определено
0	master_enable	Состояние контроллера: 0 = работа контроллера запрещена; 1 = работа контроллера разрешена

17.6.2 MDR_DMA->CFG

Регистр конфигурации DMA

Данный регистр имеет доступ только на запись. Регистр определяет состояние контроллера.

Таблица 17-33 перечисляет назначение разрядов регистра.

Таблица 17-32 – Регистр конфигурации DMA

Номер	31...8	7...5	4...1	0
Доступ	U	WO	U	WO
Сброс	0	0	0	0
	-	chnl_prot_ctrl	-	master_enable

Таблица 17-33 – Назначение разрядов регистра dma_cfg

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...8	-	Не определено, следует записывать 0.
7...5	chnl_prot_ctrl	Определяет уровни индикации сигналов HPROT[3:1] защиты шины АHB-Lite: Разряд 7 управляет сигналом HPROT[3], с целью индикации о появлении доступа с кэшированием; Разряд 6 управляет сигналом HPROT[2], с целью индикации о появлении доступа с буферизацией;

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
		Разряд 5 управляет сигналом HPROT[1], с целью индикации о появлении привилегированного доступа. Примечания: Если разряд[n] = 1, то соответствующий сигнал HPROT в состоянии 1; Если разряд[n] = 0, то соответствующий сигнал HPROT в состоянии 0
4...1	-	Не определено. Следует записывать 0.
0	master_enable	Определяет состояние контроллера: 0 – запрещает работу контроллера; 1 – разрешает работу контроллера

17.6.3 MDR_DMA->CTRL_BASE_PTR

Регистр базового адреса управляющих данных каналов

Данный регистр имеет доступ на запись и чтение. Регистр определяет базовый адрес системной памяти размещения управляющих данных каналов.

Примечание – Контроллер не содержит внутреннюю память для хранения управляющих данных каналов.

Размер системной памяти, предназначенной контроллеру, зависит от количества каналов DMA, использующихся контроллером, а также от возможности использования альтернативных управляющих данных каналов. Поэтому количество разрядов регистра, необходимых для задания базового адреса, варьируется и зависит от варианта построения системы.

Если контроллер находится в состоянии сброса, то чтение регистра запрещено. Таблица 17-35 перечисляет назначение разрядов регистра ctrl_base_ptr.

Таблица 17-34 – Регистр базового адреса управляющих данных каналов

Номер	31...10	9...0
Доступ	R/W	U
Сброс	0	0
	ctrl_base_ptr	-

Таблица 17-35 – Назначение разрядов регистра ctrl_base_ptr

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...10	ctrl_base_ptr	Указатель на базовый адрес первичной структуры управляющих данных. См. соответствующий раздел
9...0	-	Не определено. Следует записывать 0

17.6.4 MDR_DMA->ALT_CTRL_BASE_PTR

Регистр базового адреса альтернативных управляющих данных каналов

Данный регистр имеет доступ только на чтение. Регистр возвращает при чтении указатель базового адреса альтернативных управляющих данных каналов. Если контроллер находится в состоянии сброса, то чтение регистра запрещено. Этот регистр позволяет не производить вычисления базового адреса альтернативных управляющих данных каналов.

Таблица 17-37 перечисляет назначение разрядов регистра.

Таблица 17-36 – Регистр базового адреса альтернативных управляющих данных каналов

Номер	31... 0
Доступ	RO
Сброс	0
	Alt_ctrl_base_ptr

Таблица 17-37 – Назначение разрядов регистра alt_ctrl_base_ptr

Номер Бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...0	alt_ctrl_base_ptr	Указатель базового адреса альтернативной структуры управляющих данных

17.6.5 MDR_DMA->WAITONREQ_STATUS

Регистр статуса ожидания запроса на обработку каналов

Данный регистр имеет доступ только на чтение. Регистр возвращает при чтении состояние сигналов dma_waitonreq[]. Если контроллер находится в состоянии сброса, то чтение регистра запрещено.

Таблица 17-39 перечисляет назначение разрядов регистра.

Таблица 17-38 – Регистр статуса ожидания запроса на обработку каналов

Номер	31		2	1	0
Номер	RO		RO	RO	RO
Доступ	0		0	0	0
	dma_waitonreq_status for dma_waitnreq[31]		dma_waitonreq_status for dma_waitnreq[2]	dma_waitonreq_status for dma_waitnreq[1]	dma_waitonreq_status for dma_waitnreq[0]

Таблица 17-39 – Назначение разрядов регистра dma_waitonreq_status

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...0	dma_waitonreq_status	Состояние сигналов ожидания запроса на обработку каналов DMA. При чтении: Разряд [C] = 0 означает, что dma_waitonreq[C] в состоянии 0 Разряд [C] = 1 означает, что dma_waitonreq[C] в состоянии 1

17.6.6 MDR_DMA->CHNL_SW_REQUEST

Регистр программного запроса на обработку каналов

Данный регистр имеет доступ только на запись. Регистр позволяет устанавливать программно запрос на выполнение цикла DMA.

Таблица 17-41 перечисляет назначение разрядов регистра.

Таблица 17-40 – Регистр программного запроса на обработку каналов

Номер	31		2	1	0
Доступ	WO		WO	WO	WO
Сброс	0		0	0	0
	chnl_sw_request for channel[31]		chnl_sw_request for channel[2]	chnl_sw_request for channel[1]	chnl_sw_request for channel[0]

Таблица 17-41 – Назначение разрядов регистра chnl_sw_request

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...0	chnl_sw_request	Устанавливает соответствующий разряд для генерации программного запроса на выполнение цикла DMA по соответствующему каналу DMA. При записи: Разряд [C] = 0 означает, что запрос на выполнение цикла DMA по каналу C не будет установлен; Разряд [C] = 1 означает, что запрос на выполнение цикла DMA по каналу C будет установлен. Запись разряда соответствующего нереализованному каналу, означает, что запрос на выполнение цикла DMA не будет установлен

17.6.7 MDR_DMA->CHNL_USEBURST_SET

Регистр установки пакетного обмена каналов

Данный регистр имеет доступ на чтение и запись. Регистр отключает выполнение одиночных запросов по установке dma_sreq[] и поэтому будут обрабатываться и исполняться только запросы по dma_req[]. Регистр возвращает при чтении состояние установок пакетного обмена.

Таблица 17-43 перечисляет назначение разрядов регистра.

Таблица 17-42 – Регистр установки пакетного обмена каналов

Номер	31		2	1	0
Доступ	R/W		R/W	R/W	R/W
Сброс	0		0	0	0
	chnl_useburst_set for channel [31]		chnl_useburst_set for channel [2]	chnl_useburst_set for channel [1]	chnl_useburst_set for channel [0]

Таблица 17-43 – Назначение разрядов регистра chnl_useburst_set

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...0	chnl_useburst_set	Отключает обработку запросов на выполнение циклов DMA от dma_sreq[] и возвращает при чтении состояние этих настроек. При чтении: Разряд [C] = 0 означает, что канал DMA C выполняет циклы DMA в ответ на запросы, полученные от dma_sreq[] и dma_req[]. Контроллер выполняет одиночные передачи или 2^R передач. Разряд [C] = 1 означает, что канал DMA C выполняет циклы DMA в ответ на запросы, полученные только от dma_req[]. Контроллер выполняет 2^R передач. При записи: Разряд [C] = 0 не дает эффекта. Необходимо использовать chnl_useburst_clr регистр и установить соответствующий разряд C в 0; Разряд [C] = 1 отключает возможность обрабатывать запросы на выполнение циклов DMA, полученные от dma_sreq[]. Контроллер выполняет 2^R передач. Запись разряда соответствующего нереализованному каналу не дает никакого эффекта

После выполнения предпоследней передачи из 2^R передач, в том случае, если число оставшихся передач (N) меньше чем 2^R , контроллер сбрасывает разряд chnl_useburst_set в 0. Это позволяет выполнять оставшиеся передачи, используя dma_sreq[] и dma_req[].

Примечание – При программировании channel_cfg значением N меньшим, чем 2^R , запрещена установка соответствующего разряда chnl_useburst_set в случае, если периферийный блок не поддерживает сигнал dma_req[].

В режиме работы с периферией «исполнение с изменением конфигурации», если разряд next_useburst установлен в channel_cfg, то контроллер устанавливает chnl_useburst_set [C] в 1 после окончания цикла DMA, использующего альтернативные управляющие данные.

17.6.8 MDR_DMA->CHNL_USEBURST_CLR

Регистр сброса пакетного обмена каналов

Данный регистр имеет доступ только на запись. Регистр разрешает выполнение одиночных запросов по установке dma_sreq[]. Таблица 17-45 перечисляет назначение разрядов регистра chnl_useburst_clr.

Таблица 17-44 – Регистр сброса пакетного обмена каналов

Номер	31		2	1	0
Доступ	WO		WO	WO	WO
Сброс	0		0	0	0
	chnl_useburst_clr for channel [31]		chnl_useburst_clr for channel [2]	chnl_useburst_clr for channel [1]	chnl_useburst_clr for channel [0]

Таблица 17-45 – Назначение разрядов регистра chnl_useburst_clr

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...0	chnl_useburst_clr	Установка соответствующего разряда разрешает обработку запросов на выполнение циклов DMA от dma_sreq[]. При записи: Разряд [C] = 0 не дает эффекта. Необходимо использовать chnl_useburst_set регистр для отключения обработки запросов от dma_sreq[]; Разряд [C] = 1 разрешает обрабатывать запросы на выполнение циклов DMA, полученные от dma_sreq[]. Запись разряда соответствующего нереализованному каналу не дает никакого эффекта

17.6.9 MDR_DMA->CHNL_REQ_MASK_SET

Регистр маскирования запросов на обслуживание каналов

Данный регистр имеет доступ на чтение и запись. Регистр отключает установку запросов на выполнение циклов DMA на dma_sreq[] и dma_req[]. Регистр возвращает при чтении состояние установок маскирования запросов от dma_sreq[] и dma_req[] на обслуживание каналов. Таблица 17-47 перечисляет назначение разрядов регистра chnl_req_mask_set.

Таблица 17-46 – Регистр маскирования запросов на обслуживание каналов

Номер	31		2	1	0
Доступ	R/W		R/W	R/W	R/W
Сброс	0		0	0	0
	chnl_req_mask_set for dma_req[31] and dma_sreq[31]		chnl_req_mask_set for dma_req[2] and dma_sreq[2]	chnl_req_mask_set for dma_req[1] and dma_sreq[1]	chnl_req_mask_set for dma_req[0] and dma_sreq[0]

Таблица 17-47 – Назначение разрядов регистра chnl_req_mask_set

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...0	chnl_req_mask_set	Отключает обработку запросов по dma_sreq[] и dma_req[] на выполнение циклов DMA от каналов и возвращает при чтении состоянии этих настроек. При чтении: Разряд [C] = 0 означает, что канал DMA C выполняет циклы DMA в ответ на поступающие запросы; Разряд [C] = 1 означает, что канал DMA C не выполняет циклы DMA в ответ на поступающие запросы. При записи: Разряд [C] = 0 не дает эффекта. Необходимо использовать chnl_req_mask_clr регистр для разрешения установки запросов; Разряд [C] = 1 отключает установку запросов на выполнение циклов DMA, по dma_sreq[] и dma_req[]. Запись разряда соответствующего нереализованному каналу не дает никакого эффекта

17.6.10 MDR_DMA->CHNL_REQ_MASK_CLR

Регистр очистки маскирования запросов на обслуживание каналов.

Данный регистр имеет доступ только на запись. Регистр разрешает установку запросов на выполнение циклов DMA на dma_sreq[] и dma_req[].

Таблица 17-49 перечисляет назначение разрядов регистра chnl_req_mask_clr.

Таблица 17-48 – Регистр очистки маскирования запросов на обслуживание каналов

Номер	31		2	1	0
Доступ	WO		WO	WO	WO
Сброс	0		0	0	0
	chnl_req_mask_clr for dma_req[31] and dma_sreq[31]		chnl_req_mask_clr for dma_req[2] and dma_sreq[2]	chnl_req_mask_clr for dma_req[1] and dma_sreq[1]	chnl_req_mask_clr for dma_req[0] and dma_sreq[0]

Таблица 17-49 – Назначение разрядов регистра chnl_req_mask_clr

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...0	chnl_req_mask_clr	Установка соответствующего разряда разрешает установку запросов по dma_sreq[] и dma_req[] на выполнение циклов DMA от каналов. При записи: Разряд [C] =0 не дает эффекта. Необходимо использовать chnl_req_mask_set регистр для отключения установки запросов; Разряд [C] =1 разрешает установку запросов на выполнение циклов DMA, по dma_sreq[] и dma_req[]. Запись разряда соответствующего нереализованному каналу не дает никакого эффекта

17.6.11 MDR_DMA->CHNL_ENABLE_SET

Регистр установки разрешения каналов

Данный регистр имеет доступ на чтение и запись. Регистр разрешает работу каналов DMA. Регистр возвращает при чтении состояние разрешений работы каналов DMA.

Таблица 17-51 перечисляет назначение разрядов регистра chnl_enable_set.

Таблица 17-50 – Регистр установки разрешения каналов

Номер	31		2	1	0
Доступ	R/W		R/W	R/W	R/W
Сброс	0		0	0	0
	chnl_enable_set for channel [31]		chnl_enable_set for channel [2]	chnl_enable_set for channel [1]	chnl_enable_set for channel [0]

Таблица 17-51 – Назначение разрядов регистра chnl_enable_set

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...0	chnl_enable_set	Разрешает работу каналов DMA и возвращает при чтении состоянии этих настроек. При чтении: Разряд [C] = 0 означает, что канал DMA C отключен; Разряд [C] = 1 означает, что работа канала DMA C разрешена. При записи: Разряд [C] = 0 не дает эффекта. Необходимо использовать chnl_enable_clr регистр для отключения канала; Разряд [C] = 1 разрешает работу канала DMA C. Запись разряда соответствующего нереализованному каналу не дает никакого эффекта

17.6.12 MDR_DMA->CHNL_ENABLE_CLR

Регистр сброса разрешения каналов

Данный регистр имеет доступ только на запись. Регистр запрещает работу каналов DMA.

Таблица 17-53 перечисляет назначение разрядов регистра chnl_enable_clr.

Таблица 17-52 – Регистр сброса разрешения каналов

Номер	31		2	1	0
Доступ	WO		WO	WO	WO
Сброс	0		0	0	0
	chnl_enable_clr for channel 31		chnl_enable_clr for channel 2	chnl_enable_clr for channel 1	chnl_enable_clr for channel 0

Таблица 17-53 – Назначение разрядов регистра chnl_enable_clr

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...0	chnl_enable_clr	Установка соответствующего разряда запрещает работу соответствующего канала DMA. При записи: Разряд [C] = 0 не дает эффекта. Необходимо использовать chnl_enable_set регистр для разрешения работы канала; Разряд [C] = 1 запрещает работу канала DMA C. Запись разряда соответствующего нереализованному каналу не дает никакого эффекта. Примечание – Контроллер может отключить канал DMA, установив соответствующий разряд в следующих случаях: – при завершении цикла DMA; – при чтении из channel_cfg с полем cycle_ctrl установленным в b000; – при появлении ошибки на шине AHB-Lite

17.6.13 MDR_DMA->CHNL_PRI_ALT_SET

Регистр установки первичной/альтернативной структуры управляющих данных каналов

Данный регистр имеет доступ на запись и чтение. Регистр разрешает работу канала DMA с использованием альтернативной структуры управляющих данных. Чтение регистра возвращает состояние каналов DMA (какую структуру управляющих данных использует каждый канал DMA).

Таблица 17-55 перечисляет назначение разрядов регистра chnl_pri_alt_set.

Таблица 17-54 – Регистр установки первичной/альтернативной структуры управляющих данных каналов

Номер	31		2	1	0
Доступ	R/W		R/W	R/W	R/W
Сброс	0		0	0	0
	chnl_pri_alt_set for channel [31]		chnl_pri_alt_set for channel [2]	chnl_pri_alt_set for channel [1]	chnl_pri_alt_set for channel [0]

Таблица 17-55 – Назначение разрядов регистра chnl_pri_alt_set

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...0	chnl_pri_alt_set	Установка соответствующего разряда подключает использование альтернативных управляющих данных для соответствующего канала DMA, чтение возвращает состояние этих настроек. При чтении: Разряд [C] = 0 означает, что канал DMA C использует первичную структуру управляющих данных; Разряд [C] = 1 означает, что канал DMA C использует альтернативную структуру управляющих данных. При записи: Разряд [C] = 0 не дает эффекта. Необходимо использовать chnl_pri_alt_clr регистр для сброса разряда [C] в 0; Разряд [C] = 1 подключает использование альтернативной структуры управляющих данных каналом DMA C. Запись разряда соответствующего нереализованному каналу не дает никакого эффекта. Примечание – Контроллер может переключить значение разряда chnl_pri_alt_set[C] в следующих случаях: – при завершении четырех передач DMA указанных в первичной структуре управляющих данных при выполнении цикла DMA в режимах работы с памятью или периферией «исполнение с изменением конфигурации»; – при завершении всех передач DMA указанных в первичной структуре управляющих данных при выполнении цикла DMA в режиме «Пинг-понг»;

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
		– при завершении всех передач DMA указанных в альтернативной структуре управляющих данных при выполнении цикла DMA в режимах: – «пинг-понг»; – работа с памятью «Исполнение с изменением конфигурации»; – работа с периферией «Исполнение с изменением конфигурации»;

17.6.14 MDR_DMA->CHNL_PRI_ALT_CLR

Регистр сброса первичной/альтернативной структуры управляющих данных каналов

Данный регистр имеет доступ только на запись. Регистр разрешает работу канала DMA с использованием первичной структуры управляющих данных.

Таблица 17-57 перечисляет назначение разрядов регистра chnl_pri_alt_clr.

Таблица 17-56 – Регистр сброса первичной/альтернативной структуры управляющих данных каналов

Номер	31		2	1	0
Доступ	WO		WO	WO	WO
Сброс	0		0	0	0
	chnl_pri_alt_clr for channel[31]		chnl_pri_alt_clr for channel[2]	chnl_pri_alt_clr for channel[1]	chnl_pri_alt_clr for channel[0]

Таблица 17-57 – Назначение разрядов регистра chnl_pri_alt_clr

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...0	chnl_pri_alt_clr	Установка соответствующего разряда подключает использование первичных управляющих данных для соответствующего канала DMA. При записи: Разряд [C] = 0 не дает эффекта. Необходимо использовать chnl_pri_alt_set регистр для выбора альтернативных управляющих данных; Разряд [C] = 1 подключает использование первичной структуры управляющих данных каналом DMA C. Запись разряда соответствующего нереализованному каналу не дает никакого эффекта. Примечание – Контроллер может переключить значение разряда chnl_pri_alt_clr[C] в следующих случаях: – при завершении четырех передач DMA указанных в первичной структуре управляющих данных при выполнении цикла DMA в режимах работы с памятью или периферией «исполнение с изменением конфигурации»;

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
		– при завершении всех передач DMA указанных в первичной структуре управляющих данных при выполнении цикла DMA в режиме «пинг-понг»; – при завершении всех передач DMA указанных в альтернативной структуре управляющих данных при выполнении цикла DMA в режимах: – «пинг-понг» – работа с памятью «Исполнение с изменением конфигурации» – работа с периферией «Исполнение с изменением конфигурации»

17.6.15 DFMDR_DMA->CHNL_PRIORITY_SET

Регистр установки приоритета каналов

Данный регистр имеет доступ на запись и чтение. Регистр позволяет присвоить высокий приоритет каналу DMA. Чтение регистра возвращает состояние приоритета каналов DMA.

Таблица 17-59 перечисляет назначение разрядов регистра chnl_priority_set.

Таблица 17-58 – Регистр установки приоритета каналов

Номер	31		2	1	0
Доступ	R/W		R/W	R/W	R/W
Сброс	0		0	0	0
	chnl_priorit_set for channel[31]		chnl_priority_set for channel[2]	chnl_priority_set for channel[1]	chnl_priority_set for channel[0]

Таблица 17-59 – Назначение разрядов регистра chnl_priority_set

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...0	chnl_priority_set	Установка высокого приоритета каналу DMA, чтение возвращает состояние приоритета каналов DMA. При чтении: Разряд [C] = 0 означает, что каналу DMA C присвоен уровень приоритета по умолчанию; Разряд [C] = 1 означает, что каналу DMA C присвоен высокий уровень приоритета. При записи: Разряд [C] = 0 не дает эффекта. Необходимо использовать chnl_priority_clr регистр для установки каналу C уровня приоритета по умолчанию; Разряд [C] = 1 устанавливает каналу DMA C высокий уровень приоритета. Запись разряда соответствующего нереализованному каналу не дает никакого эффекта

17.6.16 MDR_DMA->CHNL_PRIORITY_CLR

Регистр сброса приоритета каналов

Данный регистр имеет доступ только на запись. Регистр позволяет присвоить каналу DMA уровень приоритета по умолчанию.

Таблица 17-61 перечисляет назначение разрядов регистра chnl_priority_clr.

Таблица 17-60 – Регистр сброса приоритета каналов

Номер	31		2	1	0
Доступ	WO		WO	WO	WO
Сброс	0		0	0	0
	chnl_priorit_clr for channel[31]		chnl_priority_clr for channel[2]	chnl_priority_clr for channel[1]	chnl_priority_clr for channel[0]

Таблица 17-61 – Назначение разрядов регистра chnl_priority_clr

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
[31:0]	chnl_priority_clr	Установка разряда присваивает соответствующему каналу DMA уровень приоритета по умолчанию. При записи: Разряд [C] = 0 не дает эффекта. Необходимо использовать chnl_priority_set регистр для установки каналу C высокого уровня приоритета. Разряд [C] = 1 устанавливает каналу DMA C уровень приоритета по умолчанию. Запись разряда соответствующего нереализованному каналу не дает никакого эффекта

17.6.17 MDR_DMA->ERR_CLR

Регистр сброса флага ошибки

Данный регистр имеет доступ на запись и чтение. Регистр позволяет сбрасывать сигнал dma_err в 0. Чтение регистра возвращает состояние сигнала dma_err.

Таблица 17-63 перечисляет назначение разрядов регистра err_clr.

Таблица 17-62 – Регистр сброса флага ошибки

Номер	31...1	0
Доступ	U	R/W
Сброс	0	0
	-	err_clr

Таблица 17-63 – Назначение разрядов регистра err_clr

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...1	-	Не определено. Следует записывать 0
0	err_clr	Установка сигнала в состояние 0, чтение возвращает состояние сигнала (флага) dma_err. При чтении: Разряд [C] = 0 означает, что dma_err находится в состоянии 0; Разряд [C] = 1 означает, что dma_err находится в состоянии 1. При записи: Разряд [C] =0 не дает эффекта. Состояние dma_err останется неизменным; Разряд [C] =1 сбрасывает сигнал (флаг) dma_err в состояние 0. Примечание – При сбросе сигнала dma_err одновременно с появлением ошибки на шине АНВ-Lite, то приоритет отдается ошибке и следовательно, значение регистра (и dma_err) останется неизменным (несброшенным)

18 Организация управления подсистемой DSP

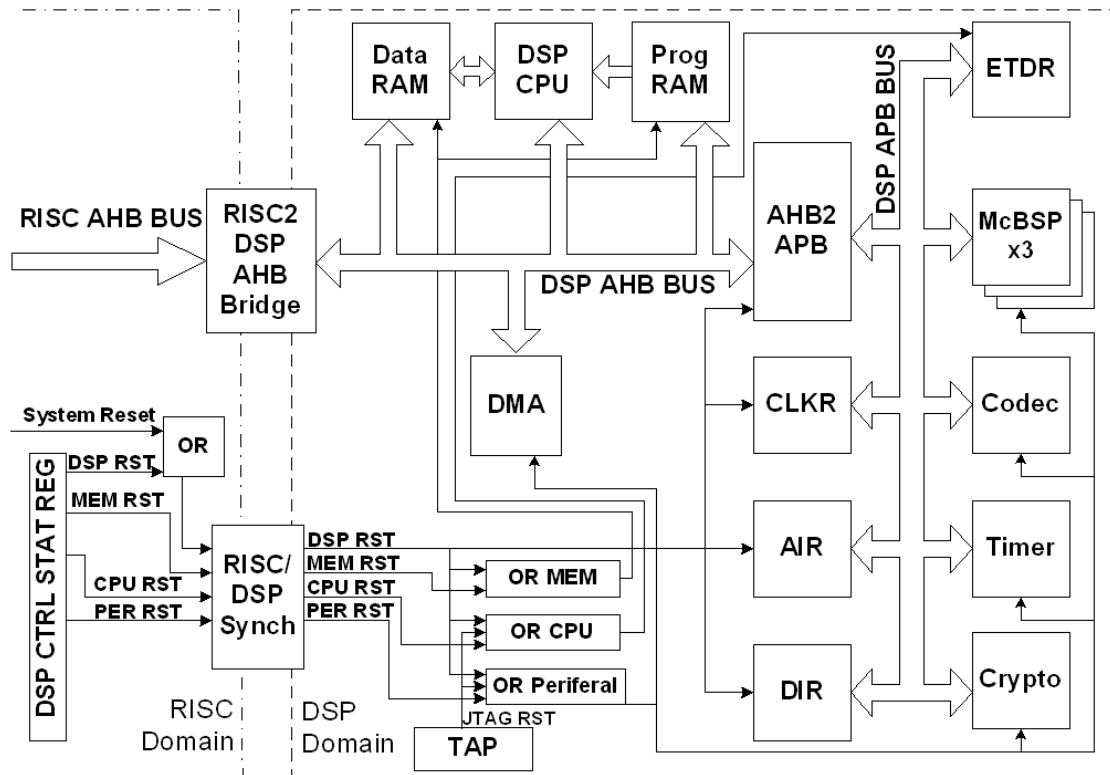


Рисунок 18-1 – Структурная схема подсистемы DSP

18.1 Средства управления подсистемой DSP

При включении питания подсистема DSP не активна (находится в сбросе, тактовая частота на подсистему не подается). Активировать DSP подсистему можно только при помощи RISC. Для этого ядро RISC обладает следующими средствами:

- Управление тактовой частотой DSP. Для управления тактовой частотой DSP в системе реализован регистр управления тактовой частотой DSP **DSP_CLOCK** в подсистеме RISC и **CLKMD** в подсистеме DSP. Возможны различные варианты источника синхросигнала DSP (в т. ч. в системе реализована отдельная PLL для DSP). Возможно отключение тактовых сигналов от отдельных модулей подсистемы DSP.
- Управление сбросами подсистемы DSP. Возможен общий сброс системы, сброс отдельно ядра DSP, всей памяти подсистемы, всех периферийных модулей.
- Ядро RISC и DMA RISC имеет доступ ко всему адресному пространству памяти программ и памяти данных DSP, кроме регистров ядра. Таким образом RISC имеет возможность задавать и изменять программы для ядра DSP, управлять всеми периферийными устройствами DSP, управлять DMA DSP.

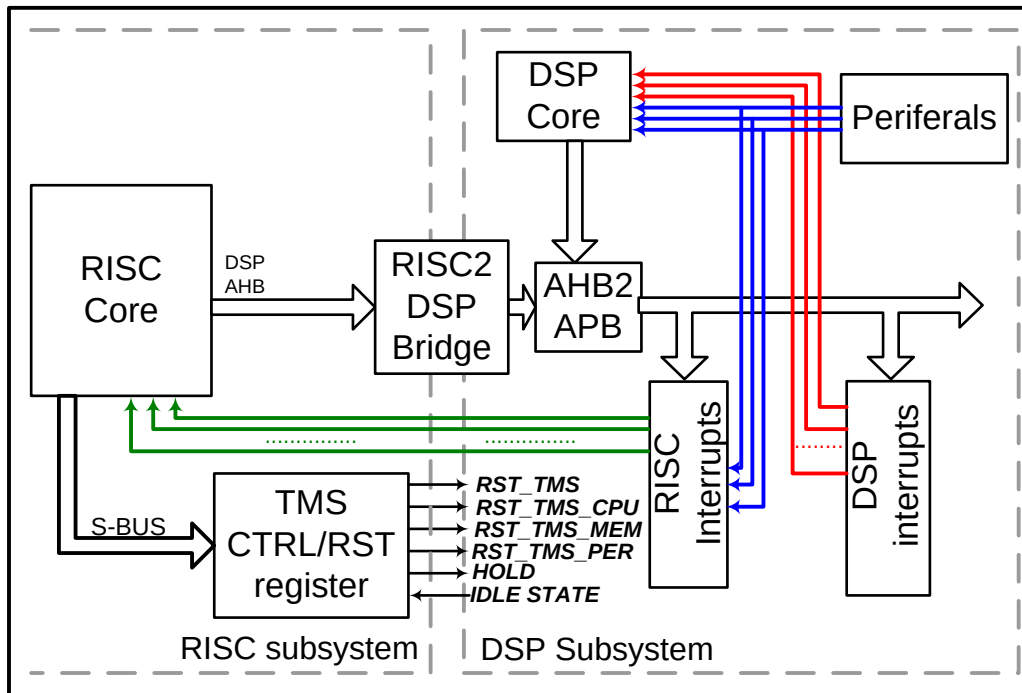


Рисунок 18-2 – Организация управления подсистемой DSP

18.2 Синхросигналы подсистемы DSP

Первичными синхросигналами для подсистемы DSP являются:

- синхросигнал CLK_DSP, который формируется в блоке контроллера тактовой частоты RISC, регистр DSP_CLOCK;
- синхросигнал TCK, который формируется аппаратным эмулятором при отладке и подается на выводы TCK.

Вторичным синхросигналом подсистемы DSP является сигнал CLK_MUX, который формируется путем мультиплексирования исходных синхросигналов CLK_DSP в рабочем режиме и сигнала TCK интерфейса JTAG в режиме отладки. То есть в случае начала работы с контроллером TAP средствами DSP отладочного интерфейса JTAG через среду Code Composer вся система переходит на синхросигнал TCK. Если отладка ядра DSP не производится (в том числе в случае отладки только ядра RISC), подсистема DSP работает на основной частоте CLK_DSP. Сигнал CLK_MUX подается на все шины подсистемы, регистры прерываний AIRQ и DIRQ, модуль управления синхросигналами и регистр CLKMD. В случае отсутствия сигнала CLK_MUX ни одна часть подсистемы не активна.

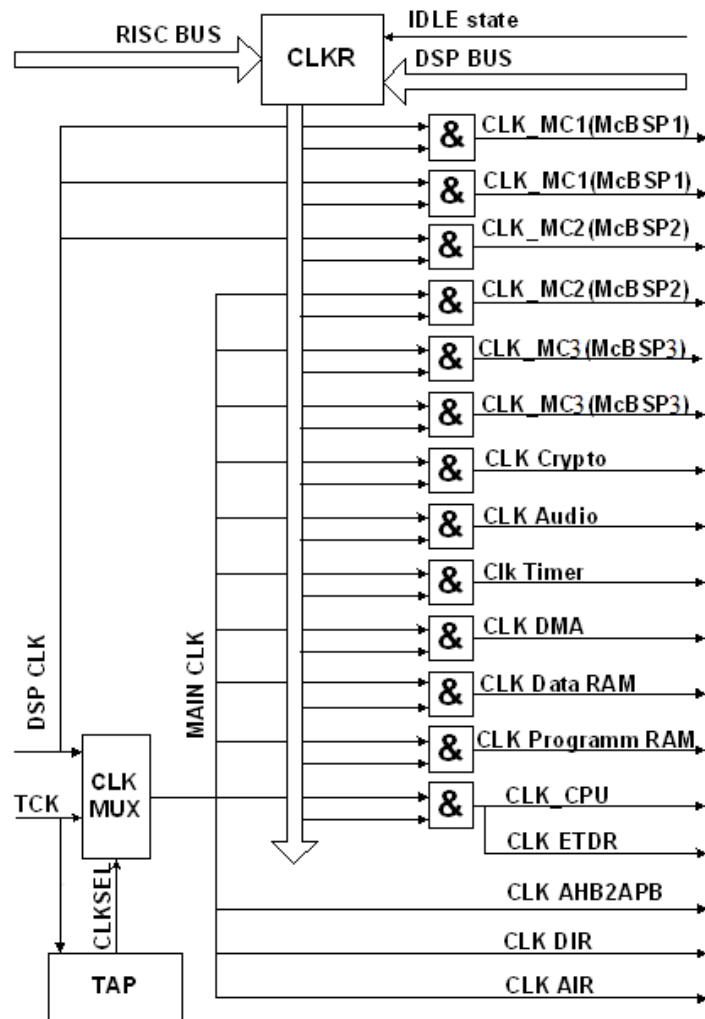


Рисунок 18-3 – Структурная схема дерева синхросигналов подсистемы DSP

Все остальные синхросигналы сопроцессора формируются путем логического умножения CLK_MUX и соответствующего сигнала разрешения синхронизации заданного устройства регистра CLKMD. Регистр CLKMD отображен в адресное пространство RISC и DSP.

18.2.1 Режимы отключения синхросигналов подсистемы DSP (IDLE1, IDLE2, IDLE3)

Регистр CLKMD программно доступен как по записи, так и по чтению со стороны обоих ядер и DMA контроллеров системы. Однако при записи данных в регистр CLKMD со стороны ядра DSP, биты 15 – 13 не могут быть обнулены (т.е. DSP не может программно отключить синхросигнал для самого себя и своей памяти данных или программ). Для выполнения этих операций в подсистеме реализованы команды пониженного потребления IDLE1, IDLE2, IDLE3. При выполнении инструкции IDLE 1 отключается только синхросигнал ядра сопроцессора (обнуляется бит 15). При выполнении инструкции IDLE 2 отключаются синхросигнал ядра сопроцессора и синхросигналы периферийных устройств (обнуляются биты 15, 12 – 3). При выполнении инструкции IDLE 3 отключаются синхросигнал ядра DSP, синхросигналы периферийных устройств, а также синхросигналы памяти программ и памяти данных DSP (обнуляются биты 15 – 3). Переход процессора в состояние IDLE генерирует соответствующее прерывание RISC процессору. Код состояния DSP может быть считан в

регистре DSP_CONTROL_STATUS. В случае выполнения любой из команд, ядро DSP заканчивает выполнение выбранных команд и очищает конвейер. Вывод процессора DSP из режимов отключения синхросигналов может быть осуществлен при помощи прерывания от RISC или от периферийных модулей DSP. В этом случае биты регистра CLKMD автоматически установятся, подача синхросигналов будет возобновлена, ядро DSP перейдет по адресу вектора активного прерывания. Не рекомендуется отключать синхросигналы ядра и памяти DSP путем записи в регистр, в тот момент, когда ядро DSP находится в активном состоянии.

18.3 Сбросы подсистемы DSP

Управление сбросами устройств подсистемы DSP может быть осуществлено только со стороны RISC. Для этого в регистр DSP_CONTROL_STATUS включены биты управления сбросами: RST_DSP, RST_DSP_CPU, RST_DSP_MEM, RST_DSP_PER.

Общий сброс подсистемы DSP

Бит RST_DSP в регистре DSP_CONTROL_STATUS служит для общего сброса подсистемы DSP. В начальный момент времени он равен нулю (неактивен), но, т.к. он пересинхронизируется на частоту DSP, вывести подсистему из состояния сброса он может только после подачи синхросигнала CLK_DSP. Данный бит автоматически выводит из сброса все шины подсистемы, регистры прерываний AIRQ и DIRQ, модуль управления синхросигналами и регистр CLKMD. Остальные устройства подсистемы DSP будут выведены из сброса при условии неактивных частных битов управления сбросами. В том случае, если бит равен единице (активен) сброшены все устройства подсистемы DSP, запрещен любой обмен данными, все адреса подсистемы читаются как 0.

Сброс ядра DSP

Бит RST_DSP_CPU управляет сбросом ядра DSP. После подачи питания находится в активном состоянии (ядро сброшено).

Сброс памяти DSP

Бит RST_DSP_MEM управляет сбросом памяти программ и данных DSP. После подачи питания находится в активном состоянии (память сброшена). Влияет только на возможность считывать и записывать данные в память. Не влияет на содержимое ячеек памяти.

Сброс периферийных модулей DSP

Бит RST_DSP_PER управляет сбросом всех периферийных модулей подсистемы DSP. После подачи питания находится в активном состоянии (периферия сброшена).

18.4 Прерывания и запросы DMA между подсистемами

Для утановки запросов прерываний и DMA запросов в системе реализованы регистры AIRQ и DIRQ. Оба регистра доступны RISC и DSP. Регистр AIRQ используется для установки прерываний от RISC к DSP и для организации запросов к DMA RISC

со стороны ядра DSP. Регистр DIRQ используется для передачи запросов прерываний от ядра и периферии DSP к RISC.

Для предотвращения потери прерываний при перезаписи регистров в каждом регистре имеется бит SNR.

Для перезаписи битов прерываний (регистры AIRQ, DIRQ) необходимо сформировать следующее слово (Таблица 18-1 и Таблица 18-2):

1 В бит SNR вносится значение, которое требуется записать в один или несколько бит регистра.

2 Если записываемый регистр AIRQ, устанавливаем SID в 1.

3 Один или несколько бит соответствующих прерываниям, которые требуется перезаписать, устанавливается в «1».

То есть в случае записи прерываний значащим битом является SNR, биты соответствующие прерываниям являются выбором записываемых бит. Бит SID регистра AIRQ является выбором перезаписи прерываний или запросов DMA.

Для установки запросов RISC DMA (регистр AIRQ) необходимо сформировать следующее слово:

1 Бит SID = 0. В этом случае запись в регистр не оказывает влияния на биты запросов прерываний.

2 Запись бит соответствующие запросам DMA RISC производится обычным образом (значение регистра SNR не влияет).

Таблица 18-1 – Биты регистра запроса прерываний от RISC к DSP AIRQ

Записываемое слово				Изменение регистра	
SID	SNR	ANMI(i), AIRQ(i)	DMARQ(i) DMADONE(i)	ANMI, AIRQ(i)	DMARQ(i) DMADONE(i)
1	0	0	X	Не изменяется	Не изменяется
1	1	0	X	Не изменяется	Не изменяется
1	0	1	X	0	Не изменяется
1	1	1	X	1	Не изменяется
0	0	0	NEW DATA	Не изменяется	NEW DATA
0	1	0	NEW DATA	Не изменяется	NEW DATA
0	0	1	NEW DATA	Не изменяется	NEW DATA
0	1	1	NEW DATA	Не изменяется	NEW DATA

Таблица 18-2 – Биты регистра запросов прерываний от DSP к RISC DIRQ.

Data_in(i)	Data_in(15)	DIRQ(i)	Соответствующее прерывание
0	0	Не изменяется	Не изменяется
0	1	Не изменяется	Не изменяется
1	0	0	Сбрасывается (выход равен 1)
1	1	1	Устанавливается (выход равен 0)

18.5 Специальные возможности управления

Сигналы HOLD, HOLDA

В микросхеме реализована возможность остановки конвейера DSP со стороны RISC. Для этого в регистр DSP_CONTROL_STATUS включены биты HOLD и HOLDA. Активный уровень обоих сигналов – единица. Для остановки конвейера необходимо установить бит HOLD. При этом конвейер процессора остановится и бит HOLDA установится в 1.

Флаги BIO, XF

Также в системе реализованы флаги BIO и XF. Состояние данных флагов может быть обработано специальными командами процессора DSP.

В микросхемах ревизии 3 добавлена возможность вывода сигнала XF на порт PA[15] в режиме работы «Переопределенной и DSP функции». При этом для обеспечения совместимости микросхемы с предыдущими ревизиями данная функция совмещена с ранее реализованной на данном порту функцией внешнего прерывания. В регистр «Управления и статуса DSP» включен ранее зарезервированный бит XF_EN, переводящий вывод PA[15] в режим вывода флага XF.

Таким образом для прямого вывода флага XF необходимо:

- Перевести порт PA[15] микросхемы в режим «Переопределенной и DSP функции». Порт, при этом, будет работать в режиме внешнего прерывания.
- Установить бит XF_EN регистра «Управления и статуса DSP» (Таблица 18-4). Порт переключится на выход и будет транслировать состояние флага XF.

Данная функция реализована для упрощения отладки программ ядра DSP.

18.6 Работа моста пересинхронизации RISC – DSP

Так как подсистемы RISC и DSP работают на разных синхросигналах, для корректной передачи данных между ними организован модуль пересинхронизации. Для осуществления операций записи по шине RISC-DSP организовано асинхронное FIFO глубиной четыре запроса, позволяющее передавать данные на частоте наименьшей из частот двух подсистем, без потерь на пересинхронизацию. Операции чтения организованы по принципу двухфазного запроса-ответа. Минимальная задержка вносимая пересинхронизацией составляет три такта частоты RISC. В случае, если частота DSP меньше частоты RISC, данная задержка может быть существенно увеличена. При этом, чтение всегда имеет приоритет над записью.

Особенностью передачи данных с использованием асинхронного FIFO является некоторая задержка между получением сигнала READY по запросу и реальной записью данных. Таким образом, если последовательно записать несколько адресов DSP, и сразу после этого осуществить чтение данных по только что записанным адресам, существует опасность считать прежнее значение. Особенно опасна данная ситуация может быть при работе с периферийными модулями (например при записи регистра SPSA модуля McBSP, и следом регистра данных см. Контроллер McBSP(DSP)). Для того, чтобы убедиться что

все транзакции закончены и ячейки памяти содержат корректные данные, в регистре DSP_CONTROL_STATUS реализован бит BRTRD. В том случае, если он равен единице, все операции моста завершены.

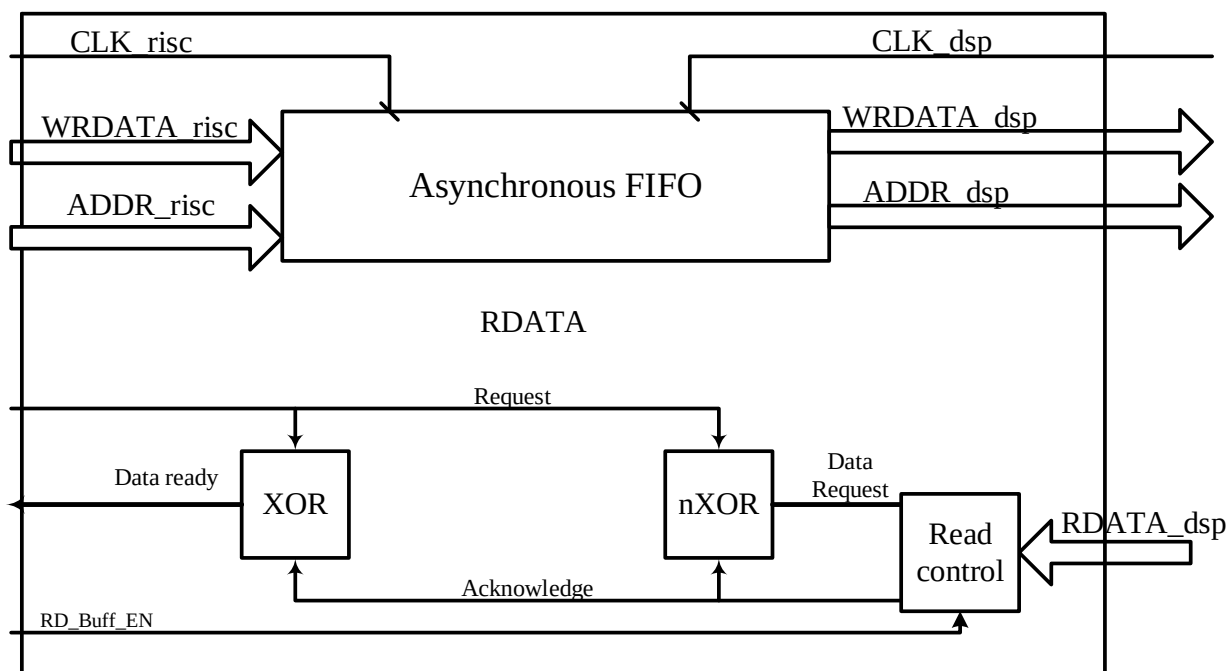


Рисунок 18-4— Структура буфера пересинхронизации RISC– DSP

Для ускорения обработки запросов чтения в модуле пересинхронизации реализован механизм предчтения. Данный механизм активизируется сигналом RD_Buff_EN регистра DSP_CONTROL_STATUS и заключается в том, что после чтения по адресу N, в случае если нет других активных задач доступа к данным, контроллер чтения автоматически вычитывает адрес N+m. Значение m зависит от размера запрошенных данных:

- m=1, если был запрошен байт;
- m=2, если было запрошено полуслово (16 бит);
- m=4, если было запрошено слово (32 бит).

Полученные данные сохраняются в специальном буфере. Если следующее чтение осуществляется по адресу N+m, данные выдаются из буфера без задержек. Данный режим рекомендуется использовать в случае чтения больших массивов последовательных данных. Не рекомендуется данный режим при обращении к регистрам (так как автоматическое чтение некоторых регистров может изменить их значение). После подачи питания механизм предчтения данных неактивен.

18.7 Способы управления подсистемой DSP

18.7.1 Нормальная работа DSP.

Для начала работы с ядром DSP необходимо:

- 1 Подать синхросигнал DSP_CLK (регистр DSP_CLOCK модуля управления синхросигналами RISC).

2 Включить память DSP (регистр DSP_CONTROL_STATUS). При этом удостоверится что общий сброс подсистемы DSP неактивен.

3 Записать в память программы данных DSP образ программного обеспечения (программу, необходимые константы, таблицу векторов прерываний).

4 Снять сброс с ядра DSP и, при необходимости, с периферии (регистр DSP_CONTROL_STATUS).

После выполнения всех операций ядро DSP начнет свою работу в штатном режиме, как автономный процессор DSP. Для окончания работы ядром DSP необходимо только установить сброс ядра DSP. После чего может быть изменен весь образ памяти DSP, либо отдельные функции. Также может быть прекращена подача синхросигнала DSP.

18.7.2 Динамическое изменение программного обеспечения ядра DSP

Изменения программы DSP без остановки ядра может быть осуществлено как при помощи RISC так и самим DSP.

Для динамического изменения программы при помощи RISC необходимо:

1 При создании образа памяти DSP, выделить участок памяти, куда гарантированно не будет обращений со стороны DSP в ходе выполнения программы. В подсистеме DSP использована расслоенная память, позволяющая одновременное безконфликтное обращение двух устройств.

2 Занести в свободный участок памяти новую функцию, требуемую к исполнению.

3 Уведомить ядро DSP о готовности функции. Для этого можно использовать прерывания регистра DIRQ, аппаратные флаги XF и BIO, программные флаги.

Для динамического изменения программы при помощи DSP можно использовать DMA контроллеры DMA DSP, если программа находится внутри подсистемы DSP, либо DMA-RISC, если программа находится вне подсистемы DSP. Ядро DSP не имеет доступа к памяти программ на запись

Для изменения программ с использованием DMA RISC со стороны DSP необходимо:

1 Со стороны RISC настроить контроллер DMA RISC таким образом, чтобы его управляющие структуры находились в подсистеме DSP. Включить каналы соответствующие запросам DSP.

2 Со стороны DSP указать в управляющей структуре адрес источника и назначения для данных.

3 Подать DMA запрос на контроллер DMA RISC (регистр AIRQ).

4 Периодически проверять готовность данных (регистр AIRQ).

18.7.3 Работа с периферийными модулями и памятью DSP без участия ядра DSP

В случае необходимости, RISC может использоваться периферийные модули и память без участия ядра DSP. Для этого необходимо:

1 Подать синхросигнал DSP_CLK (регистр DSP_CLOCK модуля управления синхросигналами RISC).

2 Включить память и(или) периферию DSP (регистр DSP_CONTROL_STATUS). При этом удостоверится что общий сброс подсистемы DSP неактивен.

3 После чего ядро RISC может без участия ядра DSP может работать с памятью и периферией подсистемы DSP.

18.8 Регистры управления подсистемой DSP

Для управления DSP частью в системе реализован набор регистров (Таблица 18-3).

Таблица 18-3 – Регистры управления подсистемой DSP

Адрес RISC	Адрес DSP	Регистр	Описание
0x4002_0038	-	DSP_CTRL_STAT	Регистр управления и статуса DSP
0x3000_00BC	005Eh	CLKMD	Управление синхросигналами DSP
0x3000_0078	003Ch	DIR	Регистры запросов прерывания от DSP к RISC
0x3000_007A	003Dh	AIR	Регистры запросов прерывания от RISC к DSP

Таблица 18-4 – Регистр управления и статуса DSP

Номер	15	14	13, 12	11	10	9	7	6	5	4	3	2	1	0
Доступ														
Сброс														
	RD_Buff_EN	BRTRD	-	IDLE num1	HOLDA	-	XF_EN	BIO	HOLD	RST_DSP_PER	RST_DSP_MEM	RST_DSP_CPU	RST_DSP	

Таблица 18-5 – Биты регистра управления подсистемой DSP

Биты	Наименование	Назначение
15	RD_Buff_EN	Разрешение предвыборки чтения адресного пространства DSP: 0 – запрещено. 1 – разрешено
14	BRTRD	Статус моста DSP: 0 – идет передача данных. 1 – все операции завершены

Биты	Наименование	Назначение
13, 12	-	Зарезервировано
11-10	IDLE num1	Код состояния DSP: 00 – подсистема DSP в нормальном режиме работы; 01 – отключен синхросигнал только от ядра DSP; 10 – отключен синхросигнал от ядра и памяти DSP; 11 – отключен синхросигнал от ядра, памяти и периферии DSP. Активен только глобальный синхросигнал DSP
9	HOLDA	Уведомление об остановки конвейера DSP: 0 – конвейер DSP остановлен; 1 – конвейер DSP работает (Не было запроса, либо конвейер еще не остановлен)
8	XF	Флаг общего назначения DSP к RISC. Обработывается специальными командами DSP.
7	-	Зарезервировано
6	XF_EN	Разрешение вывода сигнала XF напрямую на порт PA [15] микросхемы
5	BIO	Флаг общего назначения от RISC к DSP. Обработывается специальными командами DSP
4	HOLD	Запрос остановки конвейера DSP: 0 – запрос на остановку конвейера DSP; 1 – DSP работает в штатном режиме
3	RST_DSP_PER	Выключение периферийных блоков DSP: 0 – периферийные блоки работают; 1 – периферийные находятся в сбросе
2	RST_DSP_MEM	Выключение блоков памяти DSP: 0 – блоки памяти работают; 1 – блоки памяти находятся в сбросе
1	RST_DSP_CPU	Выключение ядра DSP: 0 – ядро выполняет программу; 1 – ядро находится в сбросе
0	RST_DSP	Общий сброс DSP части устройства: 0 – подсистема DSP работает; 1 – подсистема DSP находится в сбросе

18.8.1 Регистр управления синхронизацией CLKMD

Таблица 18-6 – Регистр управления синхронизацией CLKMD

Номер	15	14	13	12	11	10	9	8	7	6	5	4	3	2...0
Доступ														
Сброс														
	CPU	CPM	CDM	PC1	MC1	PC2	MC2	PC3	MC3	DMA	TMR	CDC	CRP	-

Таблица 18-7 – Биты регистра управления синхронизацией CLKMD

Бит	Название	Назначение бита регистра
15	CPU	0 – синхросигнал ядра сопроцессора запрещен; 1 – синхросигнал ядра сопроцессора разрешен
14	CPM	0 – синхросигнал памяти программ сопроцессора запрещен; 1 – синхросигнал памяти программ сопроцессора разрешен
13	CDM	0 – синхросигнал памяти данных сопроцессора запрещен; 1 – синхросигнал памяти данных сопроцессора разрешен
12	PC1	0 – синхросигнал PCLK устройства MCBSP1 запрещен; 1 – синхросигнал PCLK устройства MCBSP1 разрешен
11	MC1	0 – синхросигнал MCLK устройства MCBSP1 запрещен; 1 – синхросигнал MCLK устройства MCBSP1 разрешен
10	PC2	0 – синхросигнал PCLK устройства MCBSP2 запрещен; 1 – синхросигнал PCLK устройства MCBSP2 разрешен
9	MC2	0 – синхросигнал MCLK устройства MCBSP2 запрещен; 1 – синхросигнал MCLK устройства MCBSP2 разрешен
8	PC3	0 – синхросигнал PCLK устройства MCBSP3 запрещен; 1 – синхросигнал PCLK устройства MCBSP3 разрешен
7	MC3	0 – синхросигнал MCLK устройства MCBSP3 запрещен; 1 – синхросигнал MCLK устройства MCBSP3 разрешен
6	DMA	0 – синхросигнал контроллера DMA запрещен; 1 – синхросигнал контроллера DMA разрешен
5	TMR	0 – синхросигнал таймера запрещен; 1 – синхросигнал таймера разрешен
4	CDC	0 – синхросигнал кодека запрещен; 1 – синхросигнал кодека разрешен
3	CRP	0 – синхросигнал устройства шифрации запрещен; 1 – синхросигнал устройства шифрации разрешен
2...0	-	Резервные биты (при записи данных в регистр должны быть установлены в 0, при чтении регистра равны 0)

Таблица 18-8 – Регистр прерываний от RISC к DSP и запросов прерываний для RISC AIRQ

Номер	15	14	13	12	11	10	9	8
Доступ RISC	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Доступ DSP	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0	0	0
	SNR	SID	ADMA DONE3	ADMA DONE2	ADMA DONE1	ADMA DONE0	ADMA RQ3	ADMA RQ2

Номер	7	6	5	4	3	2	1	0
Доступ RISC	R/W	R/W	U	R/W	R/W	R/W	R/W	R/W
Доступ DSP	R/W	R/W	U	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0	0	0
	ADMA RQ1	ADMA RQ0	-	ANMI	AIRQ3	AIRQ2	AIRQ1	AIRQ0

Таблица 18-9 – Описание бит регистра прерываний от RISC к DSP и запросов прерываний для RISC AIRQ

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15	SNR	Управление установкой или сбросом битов управления прерываниями: 1 – установка; 0 – сброс
14	SID	Управление мультиплексором записи: 0 – запись бит управления DMA; 1 – запись бит прерываний (активизируется вместе с выбором требуемых бит и значением бита SNR)
13	ADMADONE3	Бит состояния готовности данных DMA RISC
12	ADMADONE2	Бит состояния готовности данных DMA RISC
11	ADMADONE1	Бит состояния готовности данных DMA RISC
10	ADMADONE0	Бит состояния готовности данных DMA RISC
9	ADMARQ3	Бит запроса передачи данных DMA RISC
8	ADMARQ2	Бит запроса передачи данных DMA RISC
7	ADMARQ1	Бит запроса передачи данных DMA RISC
6	ADMARQ0	Бит запроса передачи данных DMA RISC
5	-	Зарезервировано
4	ANMI	Немаскируемое прерывание от RISC к DSP
3	AIRQ3	Прерывание 3 от RISC к DSP
2	AIRQ2	Прерывание 2 от RISC к DSP
1	AIRQ1	Прерывание 1 от RISC к DSP
0	AIRQ0	Прерывание 0 от RISC к DSP

18.8.2 Регистр прерываний от DSP к RISC

Таблица 18-10 – Регистр запросов прерываний от DSP к RISC

Номер	15	14	13	12		11	10	9	8
Доступ RISC	R/W	U	U	R/W		R/W	R/W	R/W	R/W
Доступ DSP	R/W	U	U	R/W		R/W	R/W	R/W	R/W
Сброс	0	0	0	0		0	0	0	0
	SNR	-	-	CRIRQ		CDIRQ	XIRQ3	RIRQ3	XIRQ2

Номер	7	6	5	4	3	2	1	0
Доступ RISC	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Доступ DSP	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0	0	0
	RIRQ2	XIRQ1	RIRQ1	TIRQ	DIRQ3	DIRQ2	DIRQ1	DIRQ0

Таблица 18-11 – Описание бит регистра запросов прерываний от DSP к RISC

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15	SNR	Управление установкой или сбросом битов прерываний: 1 – установка; 0 – сброс
14	-	Зарезервировано
13	DMAIRQ	Бит состояния готовности данных DMA RISC
12	CRIRQ	прерывание от криптографического модуля
11	CDIRQ	прерывание от кодека
10	XIRQ3	прерывание от передатчика третьего устройства MCBSP
9	RIRQ3	прерывание от приемника третьего устройства MCBSP
8	XIRQ2	прерывание от передатчика второго устройства MCBSP
7	RIRQ2	прерывание от приемника второго устройства MCBSP
6	XIRQ1	прерывание от передатчика первого устройства MCBSP
5	RIRQ1	прерывание от приемника первого устройства MCBSP
4	TIRQ	прерывание от таймера DSP
3	DIRQ3	Прерывание 3 от DSP к RISC
2	DIRQ2	Прерывание 2 от DSP к RISC
1	DIRQ1	Прерывание 1 от DSP к RISC
0	DIRQ0	Прерывание 0 от DSP RISC

19 Процессорное ядро DSP

19.1 Основные особенности ядра DSP

- 40-битовый арифметико-логический модуль (арифметико-логическое устройство), Включающее 40-битовое циклическое сдвиговое и два независимых 40-битовых аккумулятора.
- 17×17-разрядный параллельный умножитель, совмещенный с 40-битовым специализированным сумматором, выполняющий в одном цикле (без конвейеризации) операцию умножения с накоплением (MAC).
- Модуль сравнения, выбора и хранения (CSSU) для операции сложения/сравнения и выбора в операторе Viterbi.
- Кодер экспоненты, для вычисления значения экспоненты 40-битового аккумулятора в одном такте.
- Два генератора адреса с восьмью вспомогательными регистрами и двумя вспомогательными арифметическими регистрами (ARAU).
- Инструкции повторения одной команды и повторения блока программного кода.
- Команды с 32-разрядными длинными словами.
- Команды с чтением двух и трех операндов (бинарные и тернарные).
- Арифметические команды с параллельным сохранением и параллельной загрузкой.
- Условные команды.
- Быстрый возврат из прерывания.
- 187 инструкций (16-разрядные инструкции, содержащие от одного до трех слов).
- шестистадийный конвейер обработки.
- Аппаратное разрешение конфликтов конвейерной обработки.

19.2 Архитектура ядра DSP

Микропроцессорное ядро ЦОС (DSP) использует расширенную, модифицированную гарвардскую архитектуру, которая увеличивает скорость обработки, поддерживая три отдельных шинных структуры для памяти данных и одну для памяти программы. Раздельные пространства программ и данных позволяют реализовать одновременный доступ к командам и данным, обеспечивая высокую степень параллелизма. Например, два чтения и одна запись могут быть выполнены в одном цикле. Команды с параллельным хранением в специфических приложениях могут полностью использовать эту архитектуру. Такой параллелизм поддерживается мощным набором арифметики, логики и операций побитовой обработки, которые могут быть выполнены в одном машинном цикле. Кроме того, имеются механизмы управления прерываниями, повтором операций, и вызовом функций. Функциональная блок-схема включает основные блоки и шинную структуру в описываемом устройстве.

Ядро DSP включает в себя следующие модули:

- Устройства управления;
- 40-битовое арифметико-логическое устройство (ALU);
- Два 40-битовых регистра аккумулятора;
- Циклический сдвигатель, поддерживающий диапазон сдвигов от -16 до 31;
- Блок умножения с накоплением;
- 16-битовый временный регистр (T);
- 16-битовый регистр перехода (TRN);
- Устройство сравнения, выбора и хранения (CSSU);
- Шифратор показателя.

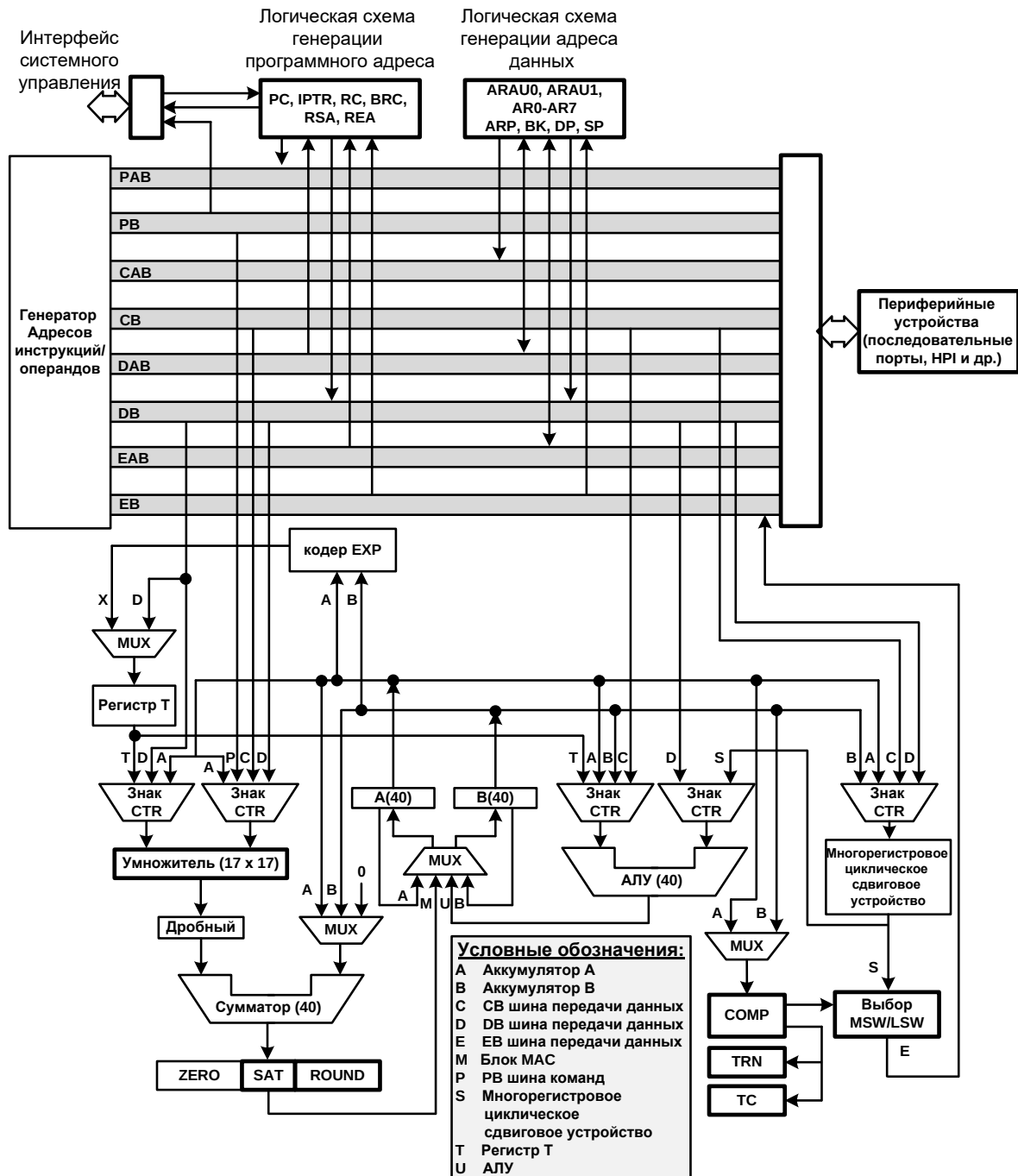


Рисунок 19-1 – Блок-схема микропроцессорного ядра

19.2.1 Шинная структура DSP

Процессорное ядро DSP использует четыре 16-разрядные шины:

- Шина чтения программ ({PAB,PB}), которая доставляет код инструкции и операнды из памяти программы;
- Две шины чтения данных ({CAB,CB}, {DAB,DB}) и одна шина записи данных ({EAB,EB}). В процессе работы может быть осуществлена одновременное чтение двух операндов и запись результата.

Микропроцессорное ядро имеет возможность генерировать до двух адресов памяти данных в цикле, которые сохраняются в двух вспомогательных регистрах модуля арифметики (ARAU0 и ARAU1).

В ЦОС части микросхемы все шины ядра преобразуются в шины стандарта AMBA. Обращение к периферийным модулям ЦОС части устройства происходит по шине APB, к памяти по шине AHB.

19.2.2 Устройство управления выполнением программ

Устройство управления выполняет следующие функции:

- Декодирует команды, управляет конвейером, хранит состояние операций, и декодирует условные операции. Некоторые из элементов аппаратных средств, включенных в контроллер программы – счетчик адреса текущей команды, регистр состояния и управления, стек и логика генерации адресов.
- Некоторые из программных механизмов, используемых для управления программой включают команды переходов, вызовы подпрограмм, условные команды, повторение команд, сброс и прерывания.

19.2.3 Арифметико-логическое устройство

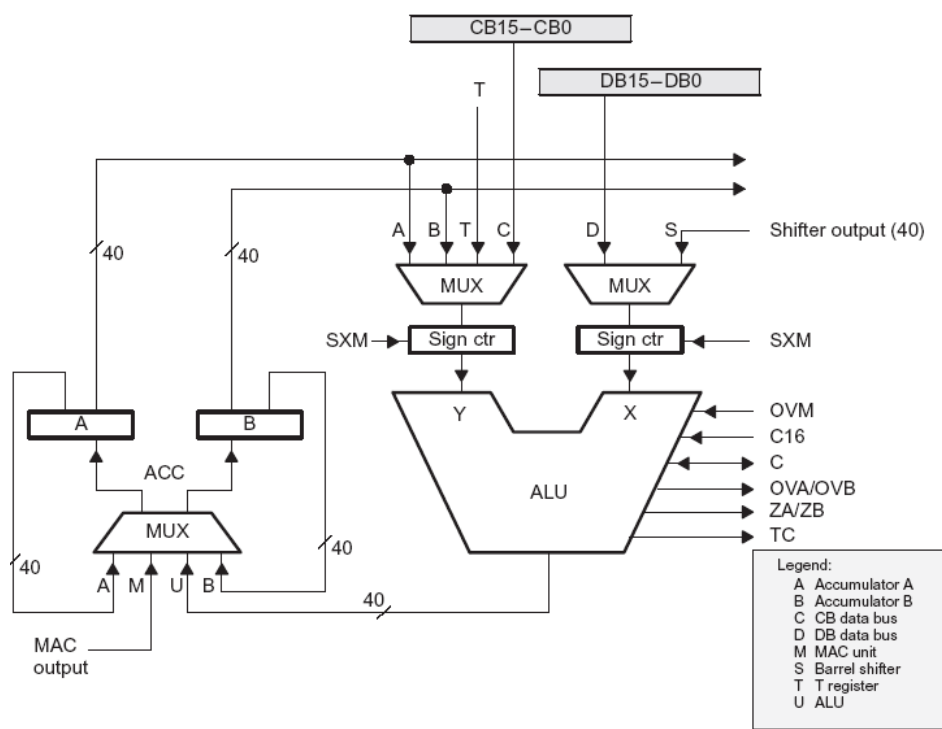


Рисунок 19-2 – Структура арифметико-логического устройства процессора DSP

Арифметико-логическое устройство может функционировать как два 16-разрядных арифметико-логических устройства и выполнять две 16-разрядных операции одновременно, когда бит C16 в регистре состояния 1 (ST1) установлен.

40-разрядное АЛУ, реализует много различных арифметических и логических операций, большинство из которого выполняется за один такт. После того, как действие будет выполнено в АЛУ, результат обычно передается на аккумулятор назначения (аккумулятор А или В). Инструкции, которые выполняют действия типа память-память (ADDM, ANDM, ORM, и XORM) являются исключением.

Источник входа X АЛУ любая из двух величин:

- Выход сдвигового устройства (32- битный или 16-битный операнд памяти данных или сдвинутое значение аккумулятора).
- Операнд памяти данных из шины данных DB.

Источник входа Y на АЛУ - любая из трех величин:

- Значение одного из аккумуляторов (А или В).
- Операнд памяти данных из шины данных CB.
- Значение в регистре Т.

Когда 16-битный операнд памяти данных подан через шину данных CB или DB, 40-разрядное значение образуется одним из двух способов:

- Если биты с 15 по 0 содержат операнд памяти данных, биты с 39 по 16 заполняются нулем ($SXM = 0$) или расширением знака ($SXM = 1$).
- Если биты 31 по 16 содержат операнд памяти данных, биты 15 по 0 заполняются нулем, и биты с 39 по 32 – заполняются или нулем ($SXM = 0$) или расширением знака ($SXM = 1$).

19.2.3.1 Переполнение АЛУ

Механизм насыщения АЛУ предохраняет результат от переполнения. Эта особенность полезна, например, для реализации фильтра.

Механизм переполнения включается, когда бит режима переполнения (OVM) регистра статуса ST1 установлен в единицу.

В том случае, если произошло переполнение, в зависимости от бита OVM выполняются следующие действия:

- Если $OVM = 0$, аккумуляторы загружены результатом АЛУ без модификации.
- Если $OVM = 1$, аккумуляторы загружаются либо максимальной положительной 32-разрядной величиной (00 7FFF FFFFh), либо максимальной отрицательной 32-разрядной величиной (0FF 8000 0000h) (в зависимости от знака переполнения).
- Флаг переполнения (OVA/OVB) в регистре статуса ST0 устанавливается для аккумулятора назначения и остается установленным пока не произойдет одно из условий:

Выполнен сброс (подсистемы DSP либо ядра DSP).

Условная инструкция (например: переход, возврат, вызов или исполнение) выполнена по условию переполнения.

Флаг переполнения (OVA/OVB) сброшен.

Аккумулятор может быть программно насыщен, с использованием инструкции SAT, независимо от значения OVM.

19.2.3.2 Бит переноса

АЛУ имеет связанный бит переноса (C), который формируется большинством инструкций арифметики АЛУ, включая действия сдвига и циклического сдвига. Бит переноса поддерживает эффективное вычисление арифметических действий повышенной точности. Бит переноса не подвержен операции загрузки аккумулятора, выполнению логических действий, или выполнению других неарифметических или управляющих инструкций, так что может быть использован для управления переполнением.

Два операнда условий, C и NC, разрешают переходы, вызовы, возвраты и условное выполнение согласно состоянию (установленное или сброшенное) бита переноса.

Также, могут быть использованы инструкции SSBX и RSBX, чтобы установить или сбросить бит переноса соответственно. Бит переноса автоматически устанавливается при сбросе.

Для арифметических действий, АЛУ может действовать в 16-разрядном режиме, когда выполняются два 16-разрядных действия (например, два сложения или два вычитания) в одном цикле. Вы можете выбрать этот режим, установив бит C16 в регистре ST1.

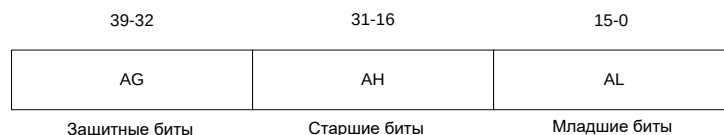
19.2.3.3 Парный 16-разрядный режим

Для арифметических действий, АЛУ может действовать в специальном парном 16-разрядном арифметическом режиме, когда выполняются два 16-разрядных действия (например, два сложения или два вычитания) в одном цикле. Данный режим может быть установлен выбором бита C16 в регистре ST1.

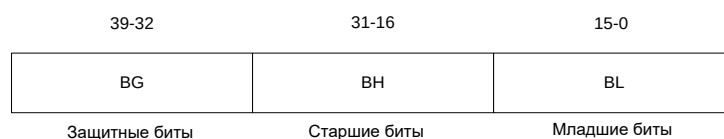
19.2.4 Аккумуляторы А и В

Аккумуляторы АССА и АССВ, хранят результаты вычислений или используются в качестве второго операнда модулей АЛУ и множителя/сумматора. Кроме того, они могут быть использованы для выполнения инструкций MIN и MAX или инструкции LD||MAC.

Каждый аккумулятор разделен на три части:



Аккумулятор А



Аккумулятор В

Рисунок 19-3 – Структура регистров аккумуляторов

Биты охраны (защитные) использованы как резерв для вычислений. Резервные биты позволяют предохраняться от переполнения в итеративных вычислениях (например автокорреляции).

AG, BG, AX, BH, AL, и BL являются регистрами отображенными в память, которые могут быть как записаны так и прочитаны из стека для сохранения контекста и его восстановления, используя инструкции PSHM и POPM. Эти регистры могут быть также использованы другими инструкциями, которые используют отображенные регистры памяти (MMR) для адресации страницы 0. Единственное различие между аккумуляторами A и B то, что биты 32-16 регистра A могут быть использованы как вход умножителя в устройстве умножения/сложения.

19.2.4.1 *Сохранение значения аккумуляторов*

Содержание аккумуляторов может быть загружено в память данных, используя инструкции STH, STL, STLM и SACCD, или используя инструкции параллельного сохранения. Для сохранения 16 младших разрядов аккумулятора в память со сдвигом, используйте инструкции STH, SACCD и параллельного сохранения. Для операций сдвига вправо, биты из AG и BG сдвигаются к AH и BH. Для операций левого сдвига, биты из AL и BL заменяют AH и BH, соответственно.

Для сохранения младших 16 разрядов аккумулятора в памяти со сдвигом, используйте инструкцию STL. Для операций сдвига вправо, биты из AH и BH заменяют AL и BL, соответственно, а младшие теряются. Для действий левого сдвига, биты в AL и BL заполняются нулями. Поскольку действия сдвига выполняются в сдвиговом устройстве, содержание аккумулятора остается неизменным.

Следующие инструкции сдвигают или циклически сдвигают содержание аккумулятора через бит переноса:

- SFTA (сдвиг арифметический).
- SFTL (сдвиг логический).
- SFTC (сдвиг условный).
- ROL (циклический сдвиг аккумулятора влево).
- ROR (циклический сдвиг аккумулятора вправо).
- ROLTC (циклический сдвиг аккумулятора влево с TC).

При выполнении команд SFTA и SFTL, количество сдвигов определено как $16 \leq \text{SHIFT} \leq 15$. Выполнение команды SFTA зависит от бита SXM. Когда $\text{SXM} = 1$ и сдвиг является отрицательной величиной, SFTA выполняет арифметический сдвиг вправо и поддерживает знак аккумулятора. Когда $\text{SXM} = 0$, старший разряд аккумулятора нулевой. Инструкция SFTL не зависит от бита SXM; поэтому выполняется действие сдвига для битов $31 - 0$, сдвигая нулевой разряд в сторону старших или младших разрядов, в зависимости от направления перемещения.

Инструкция SFTC выполняет 1-битовый сдвиг влево, когда биты 31 и 30 или оба в «1» или оба в «0». Это нормализует 32-битовый аккумулятор, устраняя наиболее значимый повторяющийся бит.

Инструкция ROL циклически сдвигает каждый бит аккумулятора влево на один бит, перемещает величину бита переноса в младший бит аккумулятора, перемещает

величину старшего бита аккумулятора в бит переноса и очищает биты охраны аккумулятора.

Инструкция ROR циклически сдвигает каждый бит аккумулятора вправо на один бит, перемещает величину бита переноса в старший бит, перемещает величину младшего разряда аккумулятора в бит переноса и очищает биты охраны аккумулятора.

Инструкция ROLTC (циклически сдвинуть аккумулятор влево с битом TC), вращает аккумулятор налево и сдвигает бит управления тестирования (TC) в младшие разряды аккумулятора.

19.2.4.2 *Насыщение при сохранении значений аккумуляторов*

Бит SST в PMST определяет необходимость насыщения данных в аккумуляторе перед сохранением его значения в памяти. Насыщение выполняется после операции сдвига. Насыщение при сохранении доступно с десятью инструкциями:

- STH
- ST || ADD
- ST || MPY
- STL
- ST || LD
- ST || SUB
- STLM
- ST || MAC[R]
- DST
- ST || MAS[R]

Следующие шаги выполняются при насыщении в случае сохранения аккумулятора:

1 40-разрядное значение данных сдвигается (вправо или влево) в зависимости от инструкции. Сдвиг такой же, как и описанный в инструкции SFTA и зависит от величины бита SXM.

2 40-битовая величина насыщается в 32-битовую величину. Насыщение зависит от значения бита SXM (число всегда предполагается положительным):

- SXM = 0. 7FFF FFFFh генерируется, если 40-битовая величина больше или равна 7FFF FFFFh.
- SXM = 1. 7FFF FFFFh генерируется, если 40-битовая величина больше, чем 7FFF FFFFh.
- 8000 0000h генерируется, если 40-битовая величина меньше чем 8000 0000h.

3 Данные загружаются в память в зависимости от инструкции (16-бит младшая часть, 16-бит старшая часть, или 32-битовые данные). Аккумулятор остается неизменным в течение этого процесса.

19.2.4.3 Специальные инструкции

Каждый аккумулятор предназначен для использования в специализированных инструкциях с параллельными операциями. Это включает операции симметричного фильтра с конечной импульсной характеристикой, использующие инструкцию FIRS, операции адаптивного фильтра, использующие инструкцию LMS, вычисления расстояния по Евклиду, использующие инструкцию SQDST, и другие параллельные действия:

- FIRS выполняет действия для симметричных фильтров с конечной импульсной характеристикой использующей умножение с накоплением (MACs) одновременно со сложением.
- LMS выполняет умножение с накоплением и параллельное сложение с округлением, чтобы эффективно обновлять коэффициенты в фильтре с конечной импульсной характеристикой.
- SQDST выполняет умножение с накоплением и параллельное вычитание.

Инструкция FIRS умножает значение аккумулятора (32-16) со значением программной памяти и добавляет результат к величине аккумулятора В. В то же самое время, она складывает операнды памяти Xmem и Ymem, сдвигает результат влево на 16 бит и загружает эту величину на аккумулятор А.

В инструкции LMS, аккумулятор В хранит временные результаты свертки входной последовательности и коэффициентов фильтра; аккумулятор А обновляет коэффициенты фильтра. Аккумулятор А может быть также использован как вход для MAC, который содействует единственному циклу выполнения инструкций с параллельными действиями.

Инструкция SQDST вычисляет квадрат расстояния между двумя векторами. Значение аккумулятора А (32-16) возводится в квадрат производится сложение с аккумулятором В. Результат сохраняется на аккумуляторе В. В то же самое время, Ymem вычитается из Xmem и разница сохраняется на аккумуляторе А. Значение, которое возводится в квадрат, есть значение аккумулятора перед вычитанием.

19.2.5 Циклическое сдвиговое устройство

Циклическое сдвиговое устройство имеет 40-разрядный ввод от аккумулятора или памяти данных (CB, DB) и 40-битовый вывод результата в АЛУ или память данных (EB). Циклическое сдвиговое устройство производит сдвиг влево от 0 до 31 бит или сдвиг вправо от 0 до 16 бит. Требования сдвига определены в поле количества (ASM) регистра ST1 или определены во временном регистре (TREG), который определяется как регистр количества сдвигов. Это сдвигающее устройство и определитель экспоненты нормализуют значения в аккумуляторе в одном цикле. Наименьшие значащие биты (младшие биты) выхода заполнены нулями, а наиболее значащие биты (MSBs) могут быть заполнены или нулями или расширены знаковым разрядом в зависимости от состояния признака бита режима (SXM) регистра ST1. Дополнительные возможности сдвига дают возможность процессору выполнить числовое масштабирование, извлечение отдельных битов, расширенную арифметику и операции предотвращения переполнения.

Циклическое сдвиговое устройство используется для масштабирования данных:

- Предварительное масштабирование входных операндов из памяти данных или значение аккумулятора перед операцией АЛУ;
- Выполнение логического или арифметического сдвига значения аккумулятора;
- Нормализация аккумулятора;
- Масштабирование аккумулятора перед сохранением значения аккумулятора в памяти данных.

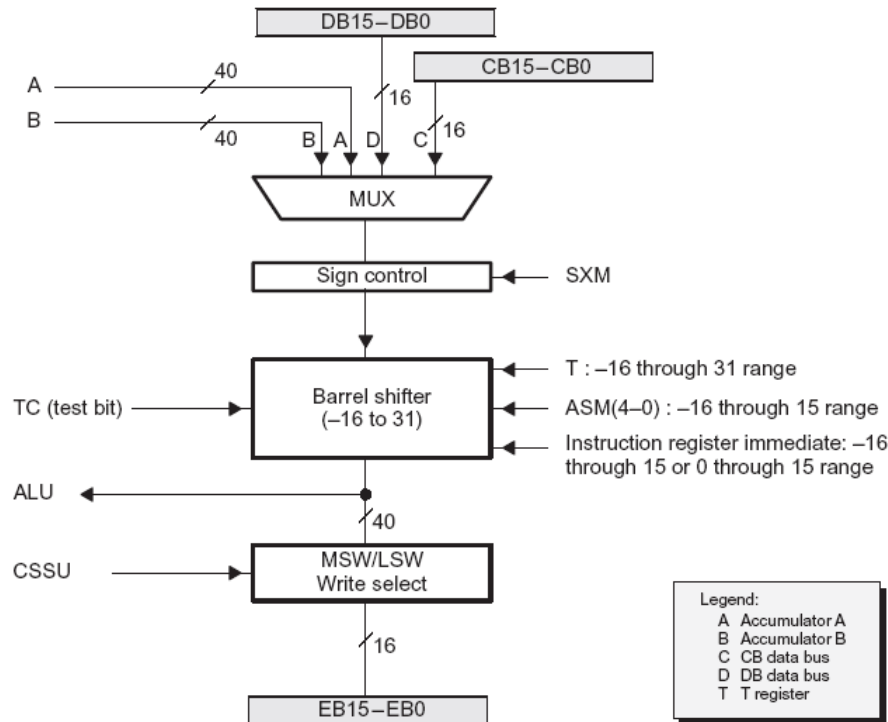


Рисунок 19-4 – Структурная схема сдвигового устройства

40-разрядное сдвиговое устройство соединено следующим образом:

- Вход модуля соединен с
 - шиной DB для входного операнда 16-битовых данных;
 - шинами DB и CB для входного операнда 32-битовых данных.
 - с одним из двух 40-разрядных аккумуляторов.
- Выход модуля соединен с:
 - одним из входов АЛУ;
 - шиной EB через устройство выбора записи MSW/LSW.

Бит SXM управляет знаковым или беззнаковым расширением операндов данных (когда бит установлен, расширение знака выполняется). Некоторые инструкции, как например, LDU, ADDS, и SUBS оперируют с беззнаковыми операндами памяти и не выполняют расширение знака, независимо от величины SXM.

Положительные значения сдвига соответствуют сдвигам влево, тогда как отрицательные значения соответствуют сдвигам вправо. Количество сдвигов определено в дополнительном коде, зависящим от типа инструкции. Непосредственный операнд, поле ASM регистра ST1 в режиме сдвига аккумулятора, или значение регистра T может быть использовано, чтобы определять количество сдвигов:

– 4- или 5-битовое непосредственное значение определенное в операнде инструкции представляет значение сдвига в диапазоне от -16 до 15. Например:

ADD A, -4, B ; сложение аккумулятора (сдвинутого вправо на 4 бита) с аккумулятором B
; (одно слово, один цикл).

SFTL A, +8 ; сдвиг (логический) аккумулятора на восемь бит влево
; (одно слово, один цикл)

– величина ASM представляет величину сдвига в диапазоне от -16 до 15 и может быть загружено инструкцией LD (с непосредственным операндом или с операндом памяти данных). Например:

ADD A, ASM, B ; сложение аккумулятору A с аккумулятором B со сдвигом определенным полем ASM

– Шесть младших бит регистра T представляют величину сдвига в диапазоне от -16 до 31. Например:

NORM A ; нормализация аккумулятора A
(T содержит величину показателя)

19.2.6 Устройство умножения/сложения

Умножитель/сумматор выполняет 17*17-битовое умножение чисел в дополнительном коде с 40-битовым накоплением в одном цикле команды. Умножитель/сумматор состоит из нескольких элементов: умножитель, сумматор, знаковый/беззнаковый входной контроль, контроль на ноль, округление в дополнительном коде, включает логику переполнения и насыщения, и временный регистр TREG. Умножитель имеет два входа: одним входом является либо TREG, операнд памяти данных или аккумулятор; другой выбирается из памяти программы, памяти данных, аккумулятора или является непосредственно заданным операндом. Быстрый аппаратный умножитель позволяет микропроцессору эффективно выполнять операции типа свертки, нахождения корреляционной функции и фильтрации. Кроме того, умножитель и арифметико-логическое устройство вместе выполняют операцию умножения с накоплением (MAC) и арифметико-логическую операцию параллельно в одном конвейерном цикле. Эта функция используется в определении расстояния по Евклиду, и для реализации симметричных фильтров и фильтров по методу наименьшего среднего квадрата (LMS), которые требуются для сложных алгоритмов DSP.

Умножитель может выполнить знаковое, беззнаковое, и знаковое/беззнаковое умножение со следующими ограничениями:

– Для знакового умножения, каждый 16-разрядный операнд памяти переводится в 17-разрядное слово с расширением знака.

– Для беззнакового умножения нуль добавляется к старшему биту (биту 16) в каждом входном операнде.

– Для знакового/беззнакового умножения, один из операндов – с расширением знака, а другой расширен нулем.

Выход умножителя может быть сдвинут влево на один бит, чтобы скомпенсировать дополнительный бит знака сгенерированный умножением двух 16-битовых дробных чисел в дополнительном коде. (Дробный режим выбирается когда бит FRCT = 1 в ST1.)

Сумматор в умножителе/сумматоре содержит детектор нуля, схему округления (в дополнительном коде) и логику переполнения/насыщения. Округление состоит из сложения 215 к результату и затем очищению младших 16 бит аккумулятора назначения.

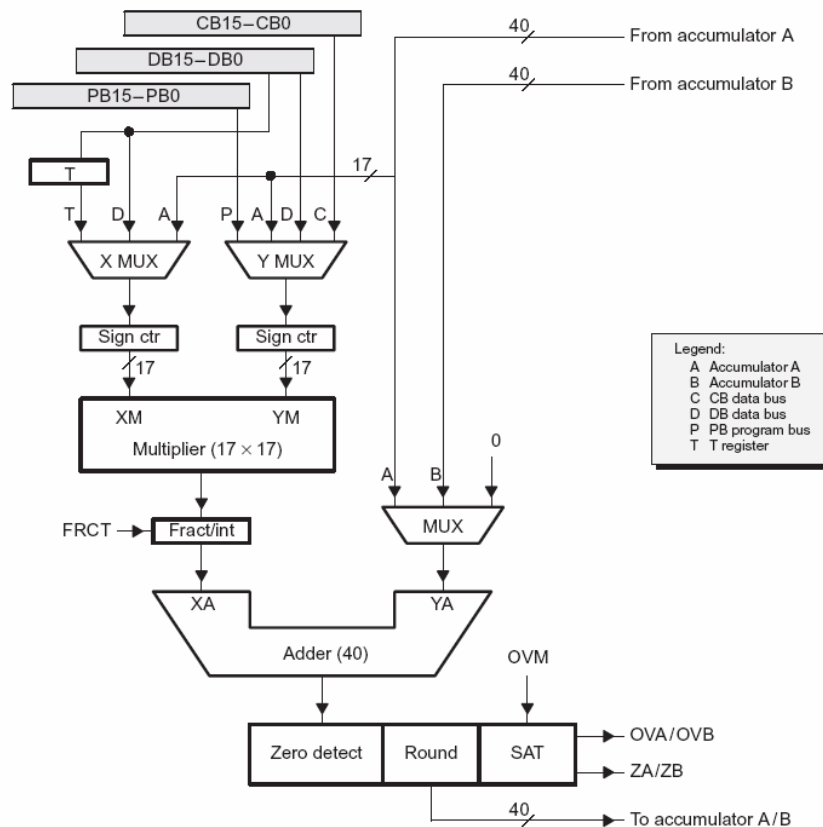


Рисунок 19-5 – 17 × 17-разрядный аппаратный умножитель

Округление выполняется в некоторых операциях умножения, операциях MAC и умножении/вычитании (MAS), когда суффикс R включен инструкцией. Инструкция LMS также округляет, чтобы минимизировать ошибки квантования в обновленных коэффициентах.

19.2.6.1 Источники данных для модуля умножителя

Вход XM в умножителе – любое из следующих значений:

- временный регистр (T);
- операнд памяти данных из шины данных DB;
- биты 32-16 аккумулятора A.

Входной источник YM в умножителе – любое из следующего значений:

- операнда памяти данных из шины данных DB;
- операнда памяти данных из шины данных CB;
- операнд из памяти программ из программной шины PB;
- биты аккумулятора 32-16.

Таблица 19-1 показывает, как ввод данных в умножитель производится для некоторых инструкций. В общей сложности имеется девять используемых комбинаций ввода.

Таблица 19-1 – Примеры ввода данных в умножитель

№ п/п	Тип инструкции	Умножитель X			Умножитель Y			
		T	DB	A	PB	CB	DB	A
1	MPY #1234h, A	√					√	
2	MPY[R] *AR2, A	√					√	
3	MPYA B	√						√
4	MACP *AR2, pmad, A		√		√			
5	MPY *AR2, *AR3, B		√			√		
6	SQUR *AR2		√				√	
7	MPYA		√					√
8	FIRS *AR2, *AR3, pmad			√	√			
9	SQUR A, B			√				√

Для инструкций, использовавших T как один ввод, второй ввод может быть получен или как непосредственная величина или из памяти данных через шину данных (DB), или с аккумулятора A.

Для инструкций, использовавших адресацию единственного операнда памяти данных, один операнд подается в умножитель через шину DB. Второй операнд может загружаться из T, как непосредственная величина или из программной памяти через шину PB, или из аккумулятора A.

Для инструкций, использовавших адресацию двойного операнда памяти данных, данные в умножитель поступают из шин DB и CB.

Последние два случая используются инструкцией FIRS и SQUR и инструкцией SQDST. Инструкция FIRS получает вводные данные с PB и аккумулятора A. SQUR и SQDST получают оба операнда из аккумулятора A.

Регистр T предоставляет один операнд для инструкции умножения и умножения с накоплением; другим операндом является операнд памяти данных одиночного доступа. Регистр T также предоставляет операнд для инструкции умножения с параллельной загрузкой или параллельной записью, как например, LD||MAC, LD||MAS, ST||MAC, ST||MAS, и ST||MPY.

Так как биты аккумулятора A (32-16), могут быть поданы на вход в умножитель, то некоторые последовательности, которые требуют сохранения результата одного вычисления в памяти и питают этим результатом множитель могут выполняться быстрее. Для некоторых специализированных инструкций (FIRS, SQDST, ABDST, и POLY), содержимое аккумулятора A может быть вычислено в АЛУ и затем использовано как вход умножителя без накладных расходов. Иначе говоря, блок АЛУ и умножителя в некоторых инструкциях работают параллельно.

19.2.6.2 Инструкции умножения с накоплением

В инструкциях MAC, MAS, и MACSU с адресацией двойного операнда памяти данных, данные могут быть переданы в умножитель в течение каждого цикла с помощью

шин CB и DB, умножены и сложены за один такт. Адреса данных для этих операндов генерируют ARAU0 и ARAU1 – вспомогательные регистры арифметического устройства.

В инструкциях MACD и MACP, данные могут быть переданы в умножитель в течение каждого цикла через DB и PB. DB извлекает данные из памяти данных, а PB извлекает коэффициенты из программной памяти. Когда MACD и MACP используются с инструкциями повторения (RPT и RPTZ), они выполняются в единственном цикле операции MAC с последовательным доступом к данным и коэффициентам. Адреса данных генерируются ARAU0 и программным адресным регистром (PAR). Адрес памяти данных обновляется ARAU0 согласно единственному операнду памяти данных в режиме косвенной адресации; адрес программной памяти инкрементируется в PAGEN.

Регулярная инструкция MACD поддерживает конструкцию фильтрации (нахождение средневзвешенного). Пока вычисляется сумма произведений, экземпляры данных сдвигаются в памяти, чтобы освободить место для следующего экземпляра и выбросить самый старый член последовательности. Инструкции MAC и MACP с циклической адресацией могут также поддержать реализацию фильтра. Инструкция FIRS осуществляет эффективную симметричную структуру для КИХ фильтра, когда используется циклическая адресация.

Инструкции MPYU и MACSU облегчают арифметические действия повышенной точности. Инструкция MPYU выполняет беззнаковое умножение. Беззнаковое содержание регистра T умножается на беззнаковое содержание адресуемой памяти данных и результат размещается в аккумуляторе. Инструкция MACSU выполняет знаковое/беззнаковое умножение и сложение. Беззнаковое содержание одной из ячейки памяти данных умножается на знаковое содержание другой ячейки памяти данных, и результат складывается с аккумулятором. Эта операция допускает операнды более чем в 16 бит, которые нужно разбить в 16- битовые слова и затем обрабатывать отдельно, чтобы сгенерировать произведение более чем в 32 бита.

Инструкция возведения в квадрат и сложения (SQURA) и возведения в квадрат и вычитания (SQURS) передает это значение данных на оба ввода множителя для возведения в квадрат. Результат сложения (SQURA) или вычитания (SQURS) с аккумулятором реализуется на уровне сумматора. Инструкция SQUR возводит в квадрат значение памяти данных или содержание аккумулятора A.

19.2.6.3 *Насыщение в модуле умножения с накоплением*

Когда установлено насыщение в умножении ($SMUL = 1$), инструкция MAC является эквивалентом $MPY + ADD$, когда $OVM = 1$. Эффект в том, что умножение, $8000h \times 8000h$, насыщено к $7FFF\ FFFFh$ в режиме дробных чисел перед выполнением последующего сложения (MAC) или вычитания (MAS).

Когда не установлено насыщение в умножении ($SMUL = 0$), только конечные результаты MAC и MAS насыщаются.

Когда $OVM = 1$ и $FRCT = 1$, бит $SMUL$ в $PMST$ определяет, насыщен ли результат умножения прежде, чем накопление будет выполнено в инструкциях MAC и MAS. Эта характеристика позволяет операциям MAC и MAS соответствовать основным операциям MAC и MAS, определенным в спецификации ETSI GSM (спецификация GSM 6.06, 6.10, и 6.53)

19.2.7 Устройство сравнения, выбора и хранения (CSSU)

Устройство сравнения, выбора и хранения является специализированным аппаратным устройством, предназначенным для операций сложения/сравнения/выбора по Витерби.

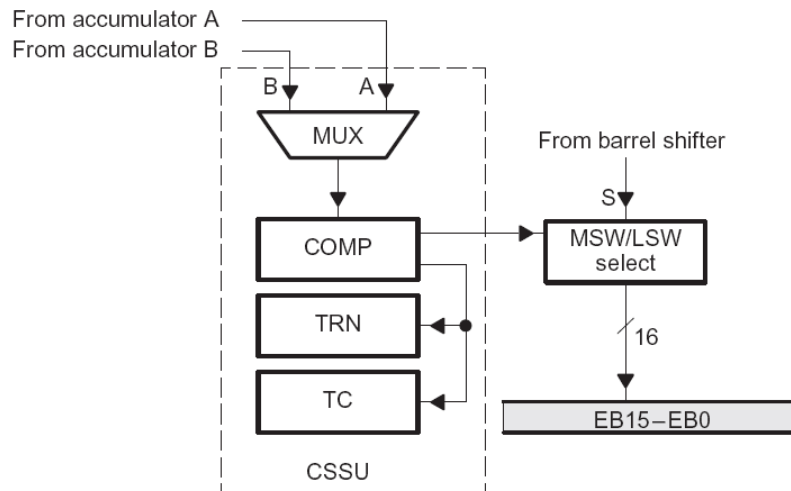


Рисунок 19-6 – Устройство сравнения, выбора и хранения (CSSU)

Это устройство позволяет процессору поддерживать различные вариации алгоритма бабочки, использованные в эквалайзерах и канальных дешифраторах.

Функция сложения оператора Viterbi, выполняется в АЛУ. Эта функция состоит из двойной функции сложения ($Met1 \pm D1$ и $Met2 \pm D2$). Двойное сложение завершается в одном такте, если АЛУ сконфигурировано в двойном 16-разрядном режиме, установкой бита C16 в ST1. С АЛУ, сконфигурированным в двойном 16-разрядном режиме, все длинные слова (32-бит) становятся двойными 16-битовыми арифметическими операндами.

Регистр Т подключен к вводу АЛУ (как двойной 16-битовый операнд) и использован как локальная память для того, чтобы минимизировать обращения к памяти. Таблица 19-2 показывает инструкции, которые выполняют двойные 16-разрядные операции.

Таблица 19-2

Инструкции	Функции (двойной 16-разрядный режим)
DADD Lmem, src [, dst]	$src(31-16) + Lmem(31-16) \rightarrow \square dst(39-16)$ $src(15-0) + Lmem(15-0) \rightarrow \square dst(15-0)$
DADST Lmem, dst	$Lmem(31-16) + T \rightarrow \square dst(39-16)$ $Lmem(15-0) - T \rightarrow \square dst(15-0)$
DRSUB Lmem, src	$Lmem(31-16) - src(31-16) \rightarrow \square src(39-16)$ $Lmem(15-0) - src(15-0) \rightarrow \square src(15-0)$
DSADT Lmem, dst	$Lmem(31-16) - T \rightarrow \square dst(39-16)$ $Lmem(15-0) + T \rightarrow \square dst(15-0)$
DSUB Lmem, src	$src(31-16) - Lmem(31-16) \rightarrow \square src(39-16)$ $src(15-0) - Lmem(15-0) \rightarrow \square src(15-0)$
DSUBT Lmem, dst	$Lmem(31-16) - T \rightarrow \square dst(39-16)$ $Lmem(15-0) + T \rightarrow \square dst(15-0)$

Инструкции	Функции (двойной 16-разрядный режим)
Обозначения: → – сохраняется в Lmem длинное (32 бит) значение памяти данных; src – аккумулятор-источник; dst – аккумулятор назначения; x(n-m) – чтение битов x от m до n	

Устройство сравнения, выбора и хранения осуществляет сравнение и выбирает действие через инструкцию CMPS, компаратор, и 16-битовый регистр перехода (TRN). Эта операция сравнивает две 16-битовых части определенного аккумулятора и сдвигает решение к биту 0 TRN. Это решение также сохраняется в бите TC регистра ST0.

Основываясь на этом решении, соответствующая 16-битовая часть аккумулятора загружается в память данных. Рисунок показывает сравнение и выбор операции, выполненной инструкцией CMPS.

```
CMPS B,*AR3 ; if (B(31-16)>B(15-0)) then
; B(31-16)->(*AR3); TRN<<1; 0->TRN(0);
; 0->TC}
; else B(15-0)->(*AR3); TRN<<1;
; 1->TRN(0); 1->TC;
```

Рисунок 19-7 – Сравнение и выбор операции CMPS

Регистр TRN содержит информацию о решении пути перехода в новые состояния. Эта информация может быть использована для обратной трассировки программы, которая находит оптимальный путь для результатов при декодировании кода.

19.2.8 Шифратор показателя

Шифратор показателя является специализированным аппаратным устройством предназначенным поддерживать инструкции EXP за один цикл. С инструкцией EXP, величина показателя на аккумуляторе может быть сохранена в T как значение в дополнительном коде в пределах диапазона от -8 до 31. Показатель определен как количество старших лишних битов -8, которые соответствуют количеству сдвигов требующихся в аккумуляторе, чтобы устранять незначащие биты знака. Это действие заканчивается отрицательной величиной, когда величина аккумулятора превышает 32 бита.

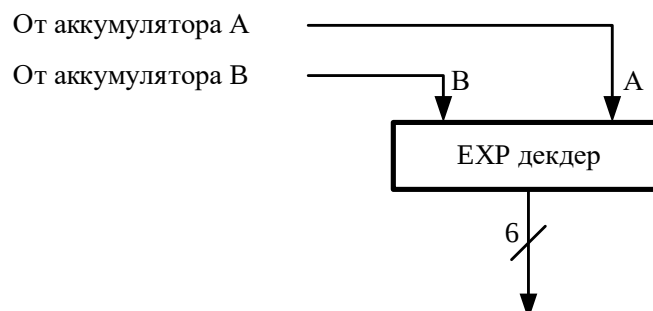


Рисунок 19-8 – Шифратор экспоненты

Инструкции EXP и NORM используют шифратор показателя, чтобы эффективно нормализовать содержание аккумуляторов. Инструкция NORM поддерживает сдвиг величины аккумулятора на число бит, определенных в T за один цикл. Отрицательная величина в T производит перемещение вправо содержания аккумуляторов, которое нормализует любую величину, выходящую за 32-битовым диапазон представлений аккумулятора. Рисунок 19-9 демонстрирует нормализацию аккумулятора A.

;Normalize accumulator A	
EXP A	; (the number of leading bits – 8)-> T.
ST T, EXPONENT	; Store the exponent (T) into data memory
NORM A	; Normalize accumulator A, (A)<<(T)

Рисунок 19-9 – Нормализация аккумулятора A

19.3 Организация конвейера

Реализованный в ядре DSP конвейер имеет следующие особенности:

- Конвейер эластичный, что означает то, что каждый его ярус передает результаты своей работы на следующий ярус при условии готовности к приему следующего яруса. При этом состояние предыдущих ярусов неважно. Перемещение команды по уровням конвейера в процессе исполнения в свою очередь зависит только от состояния этих последующих ярусов. Иначе говоря, операция управления префиксная. Префиксной называется такая операция над векторами, в которой результирующее значение каждого элемента зависит только от тех элементов исходных операндов, которые расположены только справа или только слева от данного элемента, а также элементов данной позиции. Примером префиксной операции обработки является сложение чисел, где значение каждого бита результата зависит только от значений битов слагаемых с меньшим либо тем же весом. В данном случае речь идет об операции управления, в которой вектор составлен из элементов управляющей структуры ярусов конвейера. В таком конвейере «пузыри» (bubble) могут возникать на любом ярусе, по причине того, что тормозится предыдущий ярус, однако это не мешает перемещению данного яруса. Причиной возникновения «пузырей» может быть то, что последующая команда двух или трехсловная или фаза чтения операнда затянулась на два или более такта и т.д. Если таких причин нет, то конвейер остается предельно «плотным» и в идеальном случае в каждом такте синхросигнала все команды в конвейере перемещаются на новый уровень обработки.

- Помехи типа «чтение после записи» (RAW-read after write), возникающие в конвейере разрешаются не программно, а аппаратно. При этом используется как торможение яруса чтения, так и «короткие замыкания» (bypass).

19.3.1 Стадии конвейера

Микропроцессорное ядро имеет конвейер инструкций глубиной в шесть уровней. Шесть стадий конвейера независимы друг от друга и допускают перекрытие в выполнении инструкций. В течение любого цикла, от одной до шести разных инструкций может быть активно одновременно, каждая на своем этапе выполнения.

- **Упреждающая выборка программы.** Шина программного адреса (PAB), загружается адресом следующей инструкции, которую нужно выбрать.
- **Выборка программного кода.** Слово инструкции выбирается из шины команд (PB) и загружается в регистр инструкции (IR). Это завершает последовательность выборки инструкции, которая состоит из этого и предыдущего циклов.
- **Декодирование.** Содержание регистра инструкции (IR) декодируется, чтобы определять тип операций при доступе в память и управляющую последовательность в устройстве генерации адреса данных (DAGEN).
- **Доступ.** DAGEN загружает адреса операндов на шину адреса данных, DAB. Если требуется второй операнд, другая шина адреса данных, CAB, также загружается соответствующим адресом. Вспомогательный регистр в режиме косвенной адресации и указатель стека (SP) также обновляются. Это считается первым этапом двухстадийной последовательности чтения операндов.
- **Чтение.** Операнд или операнды данных для чтения, если они необходимы, читаются из шины данных, DB и CB. Это завершает двухступенчатую последовательность чтения операндов. В то же самое время начинается двухступенчатая последовательность записи операнда. Адрес данных записываемого операнда, если это требуется, загружается на шину адреса записи данных (EAB). Для регистров, отображенных в памяти, операнд данных читается из выбранных регистров.
- **Выполнение.** Последовательность записи операнда завершается записью данных с использованием шины записи данных (EB). Инструкция выполняется в этой фазе.

Рисунок 19-10 демонстрирует этапы работы конвейера. Первые два этапа конвейера – предвыборка и выборка, составляют последовательность выборки инструкции. В первом цикле загружается адрес новой инструкции. В следующем цикле слово инструкции прочитано. В случае многословной инструкции необходимо несколько таких последовательностей выборки инструкции.

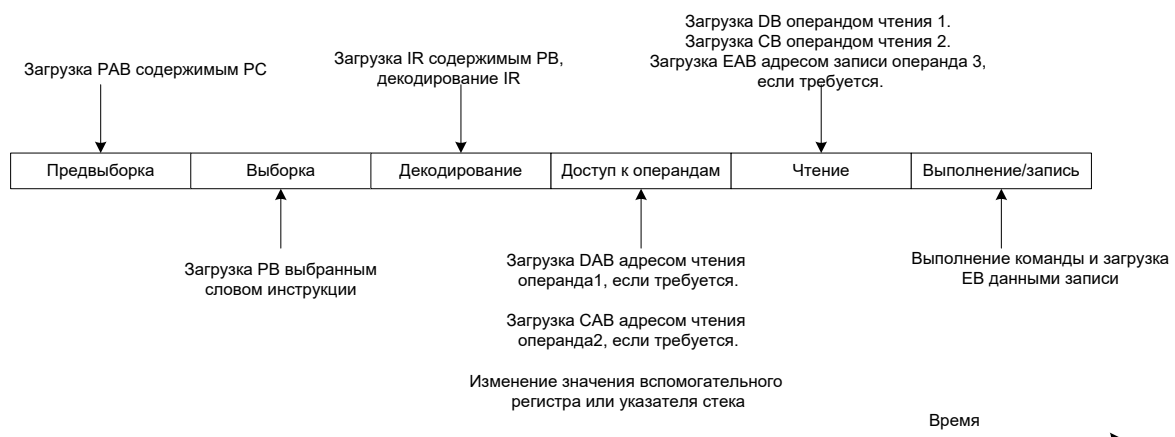


Рисунок 19-10 – Этапы работы конвейера

В течение третьего этапа конвейера – этапа декодирования, выбранная инструкция преобразуется так, чтобы подходящие управляющие последовательности были активизированы для надлежащего исполнения инструкции. Эти управляющие

последовательности являются микрокомандами для отдельных блоков в той или иной последующей стадии конвейера.

Следующие две стадии конвейера, доступа и чтения – последовательность чтения операнда. Если требуется инструкцией, адрес данных одного или двух операндов загружается в фазе доступа и операнд или операнды читаются в следующей фазе чтения.

Любая операция записи распространяется не более чем два этапа конвейера, стадию чтения и выполнения. В течение фазы чтения, адрес данных записываемого операнда сохраняется для последующей загрузки на ЕАВ. В следующем цикле операнд записывается в память, используя ЕВ.

Каждый доступ в память выполняется в две фазы конвейера. В первой фазе, шина адреса загружается адресом памяти. Во второй фазе, данные читаются на соответствующую шину данных или записывается с нее в этот адрес памяти.

Рисунок 19-11 показывает, как различные фазы доступа к памяти выполняются в конвейере. На рисунке допущено, что любой доступ к памяти выполняется в одном цикле, единственным словом инструкции с расположенной на кристалле расслоенной оперативной памятью. Внутренняя память поддерживает двойной доступ за один такт конвейера, если они адресуются в разные банки (слои).

а) выборка слова инструкции (один цикл)

Предвыборка	Выборка	Декодирование	Доступ	Чтение	Выполнение/запись
Загрузка РАВ	Чтение из РВ				

б) выполнение инструкцией чтения одного операнда (например, LD *AR1,A; один цикл)

Предвыборка	Выборка	Декодирование	Доступ	Чтение	Выполнение/запись
			Загрузка DAB	Чтение из DB	

в) инструкция выполняет чтение двух операндов (например, MAC *AR2+,*AR+,A or DLD *AR2,A; один цикл)

Предвыборка	Выборка	Декодирование	Доступ	Чтение	Выполнение/запись
			Загрузка DAB и CAB	Чтение из DB и CB	

д) инструкция выполняет запись одного операнда (например, STH A, *AR1; один цикл)

Предвыборка	Выборка	Декодирование	Доступ	Чтение	Выполнение/запись
				Загрузка ЕАВ	Запись на ЕВ

е) инструкция выполняет запись двух операндов (например, DST A, *AR1; два цикла)

Предвыборка	Выборка	Декодирование	Доступ	Чтение	Выполнение/запись
				Загрузка ЕАВ	Запись в ЕВ

Предвыборка	Выборка	Декодирование	Доступ	Чтение	Выполнение/запись
				Загрузка ЕАВ	Запись в ЕВ

и) инструкция выполняет чтение операнда и запись операнда (например, ST A, *AR2 || LD *AR3,B; один цикл)

Предвыборка	Выборка	Декодирование	Доступ	Чтение	Выполнение/запись
			Загрузка DAB	Чтение из DB; Загрузка ЕАВ	Запись в ЕВ

Рисунок 19-11 – Выполнение в конвейере различных фаз доступа к памяти

294

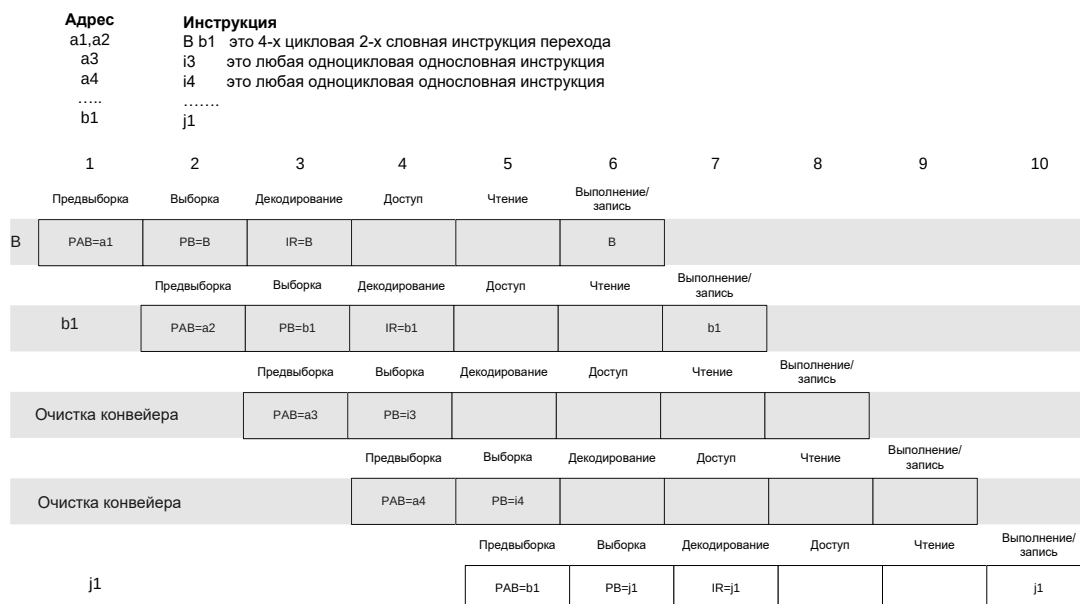


Рисунок 19-13 – Инструкция перехода в конвейере

- **Цикл 1:** РAB загружается адресом инструкции перехода.
- **Циклы 2 и 3:** Выборка двух слов инструкции перехода.
- **Циклы 4 и 5:** Выбираются две дополнительные инструкции i3 и i4. Хотя две инструкции после инструкции перехода, i3 и i4, выбираются ядром, но им не позволено перемещаться в дальнейшем на этап декодирования и, в конечном счете, они будут отвергнуты. После того как второе слово инструкции перехода (представленное как b1 в левой колонке), декодировано, РAB загружается этой новой величиной (в цикле 5).
- **Циклы 6 и 7:** двухсловная инструкция перехода входит в стадию выполнения в конвейере в циклах 6 и 7. Также, выбирается j1 по адресу b1 в цикле 6.
- **Циклы 8 и 9:** эти циклы также заняты той же инструкцией перехода, так как следующим двум инструкциям, i3 и i4, не было позволено завершить выполнение. В этом и заключается причина, по которой инструкция перехода занимает четыре цикла вместо двух.
- **Цикл 10:** j1 завершает выполнение.

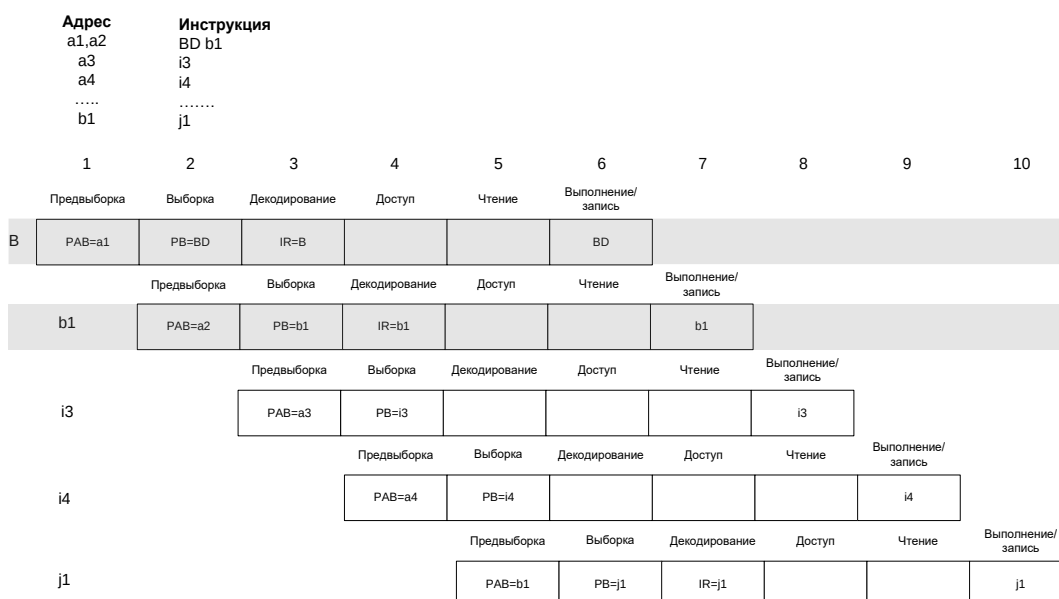


Рисунок 19-14 – Задержанная инструкция перехода в конвейере

В этом случае, конвейер ведет себя также, как при обычной инструкции перехода. Тем не менее, двум инструкциям, следующим за переходом, i3 и i4, позволено завершать выполнение. Следовательно, только циклы 6 и 7 используются задержанной инструкцией перехода, делая задержанную инструкцию перехода двухцикловой.

19.3.3 Инструкции вызова функций в конвейере

Стандартная инструкция вызова требует четыре цикла при выполнении. Хотя стандартный вызов является двухсловной инструкцией и кажется, что нужно только два цикла, он в действительности очищает конвейер в двух циклах.

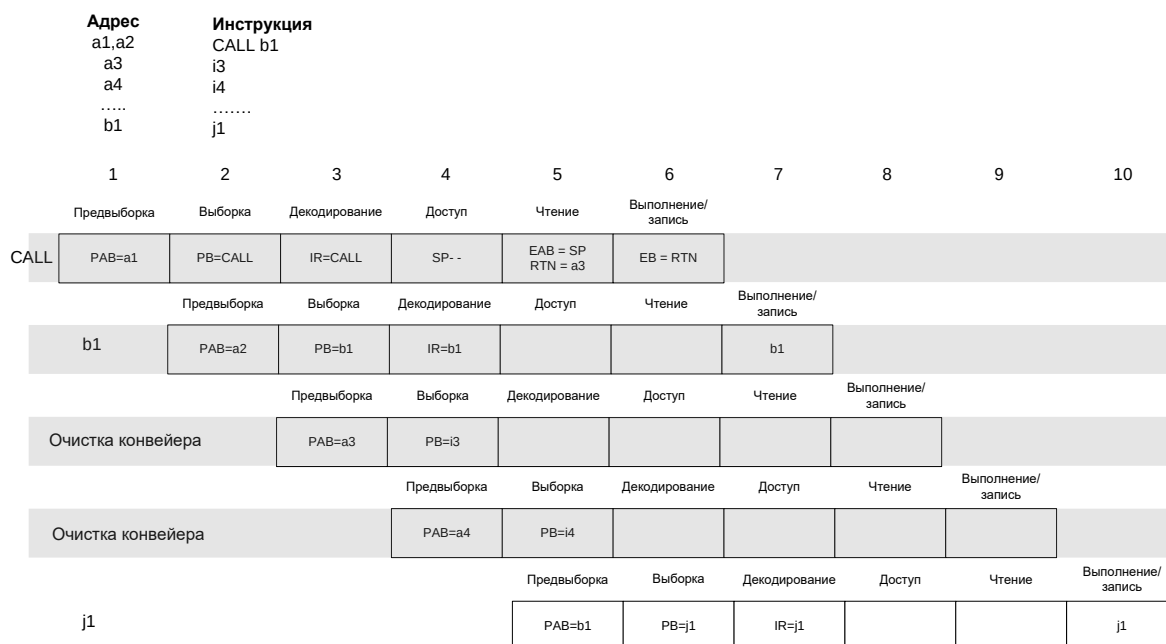


Рисунок 19-15 – Инструкция вызова в конвейере

- **Цикл 1:** PAB загружается адресом инструкции вызова функции.
- **Циклы 2 и 3:** Два слова инструкции вызова выбраны.
- **Цикл 4:** SP декрементируется (представлено как SP --), поскольку адрес возврата записан в стек. Инструкция i3 выбрана; тем не менее, ей не позволено перемещаться далее к фазе декодирования.
- **Цикл 5:** шина адреса записи (EAB) загружается содержимым SP и расположенный на кристалле регистр возврата (RTN) загружается адресом возврата a3. После того как второе слово инструкции вызова (b1) будет декодировано, PAB загружается новой величиной в цикле 5 (показано в колонке j1).
- **Циклы 6 и 7:** содержание RTN записано в стек, используя EB в цикле 6. Инструкция j1 по адресу b1 выбрана в цикле 6. Двухсловная инструкция вызова входит в стадию выполнения в конвейерных циклах 6 и 7.
- **Циклы 8 и 9:** Эти циклы поглощены инструкцией вызова, поскольку выполнение следующих двух инструкций не разрешено.
- **Цикл 10:** j1 завершает выполнение.

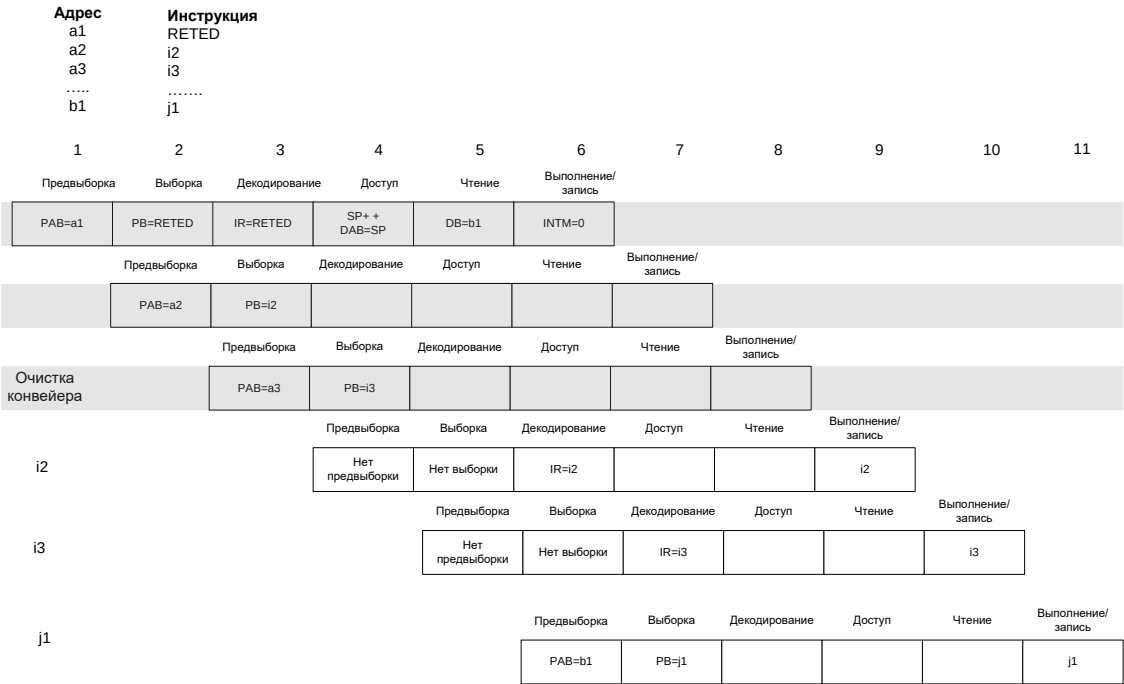


Рисунок 19-21 – Задержанная инструкция возврата с разрешением прерываний в конвейере

Следующие примеры показывают поведение конвейера для инструкции быстрого возврата (RETF) и для задержанной инструкции быстрого возврата (RETFD), соответственно. Инструкция RETF, в отличие от инструкции RETE, не делает чтение адреса возврата из стека. Вместо этого она читает его из регистра RTN. Это позволяет инструкции загружать PAB адресом возврата на два цикла быстрее, чем может инструкция RETE. Как показано в примерах, инструкция RETF использует для выполнения только три цикла; задержанная версия инструкции, RETFD, выполняется в одном цикле.

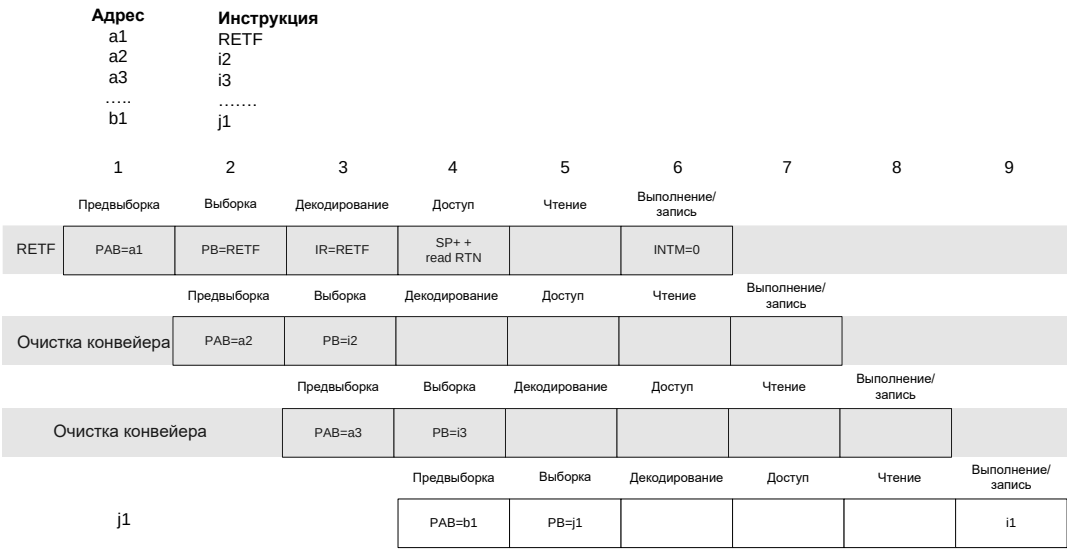


Рисунок 19-22 – Команда быстрого возврата в конвейере

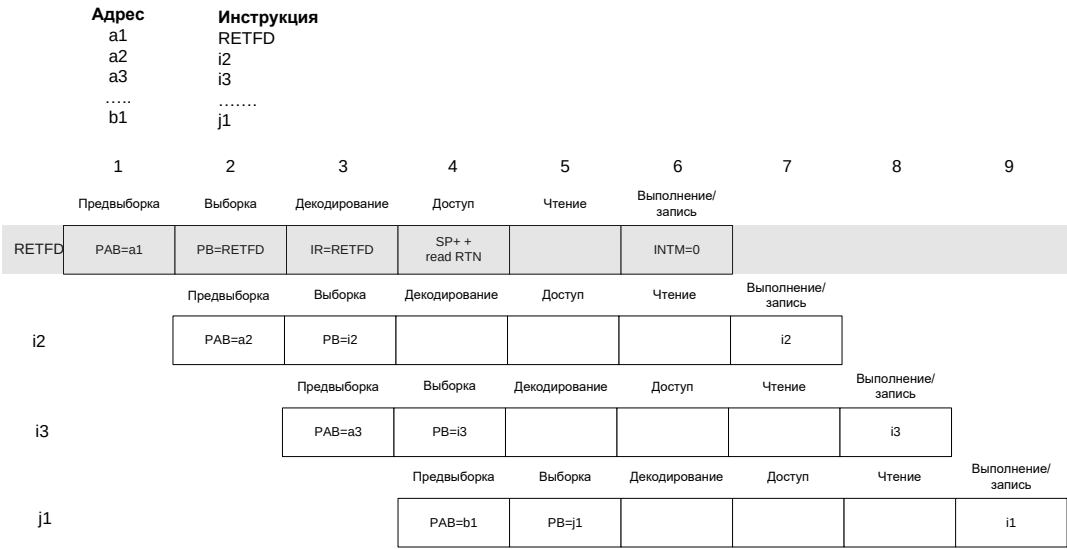


Рисунок 19-23 – Задержанная команда быстрого возврата в конвейере

19.3.5 Условная инструкция в конвейере

Поскольку ХС – однословная инструкция, она использует минимально один цикл для выполнения. Пример показывает конвейерное поведение в течение выполнения ХС.

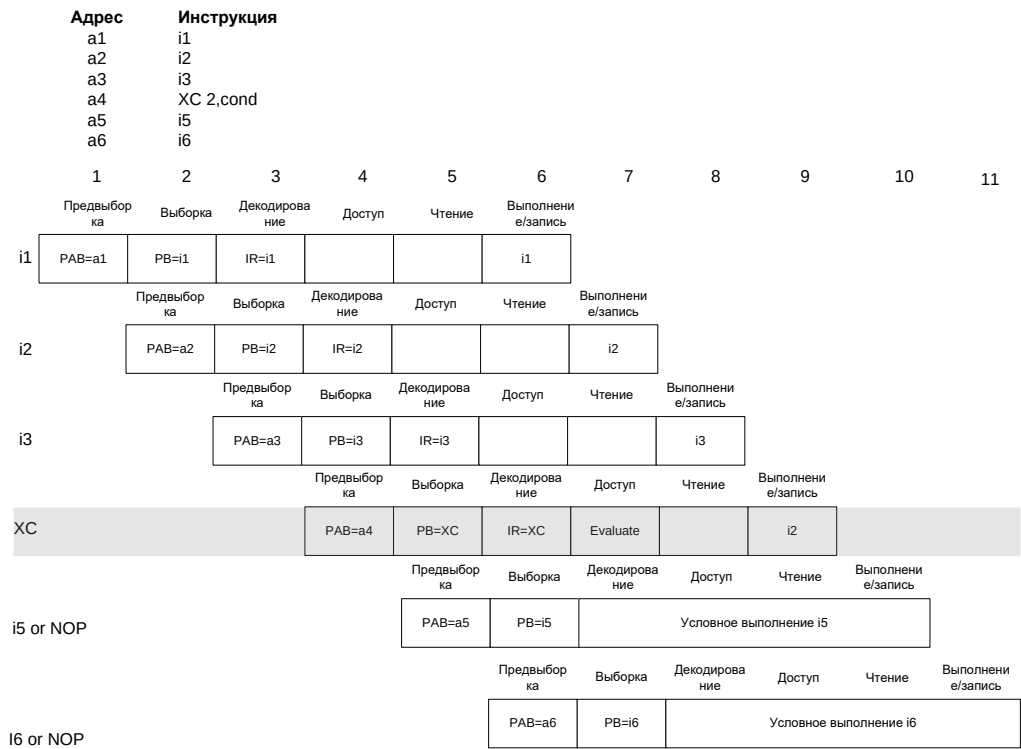


Рисунок 19-24 – Инструкция ХС в конвейере

- **Цикл 4:** PAB загружается адресом инструкции ХС.
- **Цикл 5:** инструкция ХС выбрана.
- **Цикл 6:** инструкция декодирована.
- **Цикл 7:** когда инструкция ХС перемещается на этап доступа к операндам в цикле 7 конвейера, любые условия определенные в инструкции ХС вычислены. Если тестируемые условия истинны, следующие две инструкции i5 и i6, декодированы и допускают выполнение. Тем не менее, если тестируемые условия ложны, i5 и i6 не декодируются.

Чтобы выполнять ХС в одном цикле, ядро оценивает условия тестирования на этапе доступа. Это означает, что две однословные инструкции (или одна 2-х словная) до инструкции ХС не будут полностью выполнены перед их тестированием. Поскольку коды условия изменяются инструкциями только на стадии выполнения, эти две инструкции не окажут эффекта на действия команды ХС.

19.3.6 Команды условного вызова и условного перехода в конвейере

Поскольку инструкция вызова состоит из двух слов, Вы должны ожидать, что потребуется минимум два цикла для ее выполнения. Обычная инструкция условного вызова использует при выполнении пять циклов, если выполняется или три цикла, если вызов не потребовался.

Условный вызов инструкции аналогичен в своем конвейерном поведении инструкции безусловного вызова. Единственное исключение в том, что тестируемые условия для инструкции условного вызова устанавливаются на этапе вычислений конвейера. Когда тестируемые условия определяются в цикле 7, предшествующая инструкция, i1, полностью выполнилась. Кроме того, следующие две инструкции после СС, i4 и i5, также выбраны. Если тестируемые условия оценены как ложные, эти две инструкции продолжают движение через конвейер.

В противном случае, они отвергаются. Упреждающая выборка инструкции в цикле 7 также зависит от оцениваемых условий. Если условия истинны, РАВ загружается адресом вызова (b1); в противном случае, загружается следующим инкрементированным адресом (a6).

Если оцененные условия истинны, i4 и i5 не выполняются в циклах 10 и 11. В этом случае, инструкция СС становится инструкцией 5-ти цикловой. Тем не менее, если оцененные условия ложные, команды i4 и i5 выполняются, делая инструкцию СС трехцикловой.

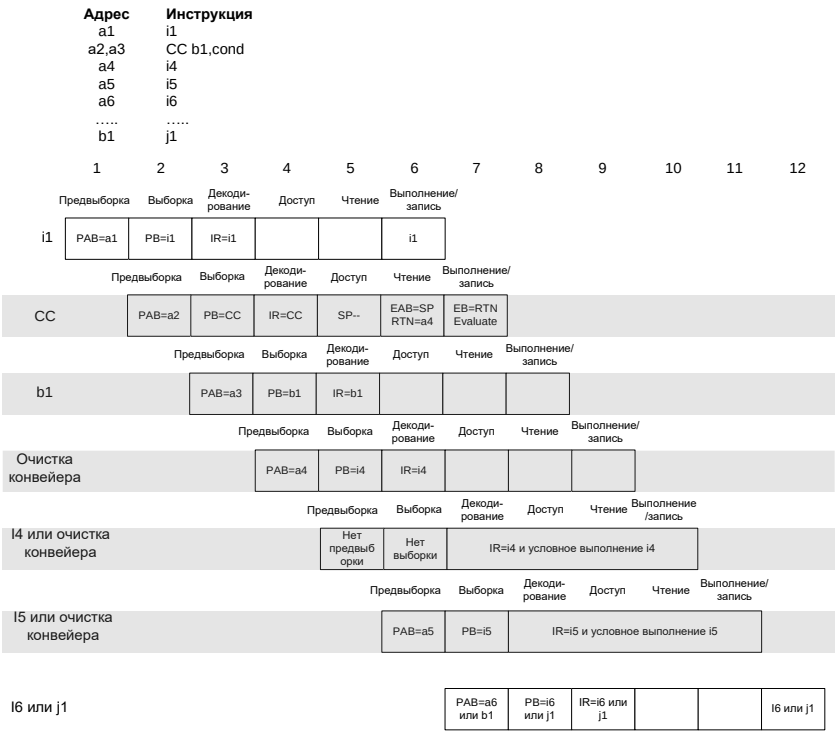


Рисунок 19-25 – Команда условного вызова (CC) в конвейере

Конвейер ведется себя также для инструкции CCD. Тем не менее, следующим двум инструкциям, i3 и i4, позволено завершить выполнение независимо от того, истинны или нет протестированные условия. Только циклы 7, 8, и 9 использованы инструкцией CCD, делая эту инструкцию трехцикловой.

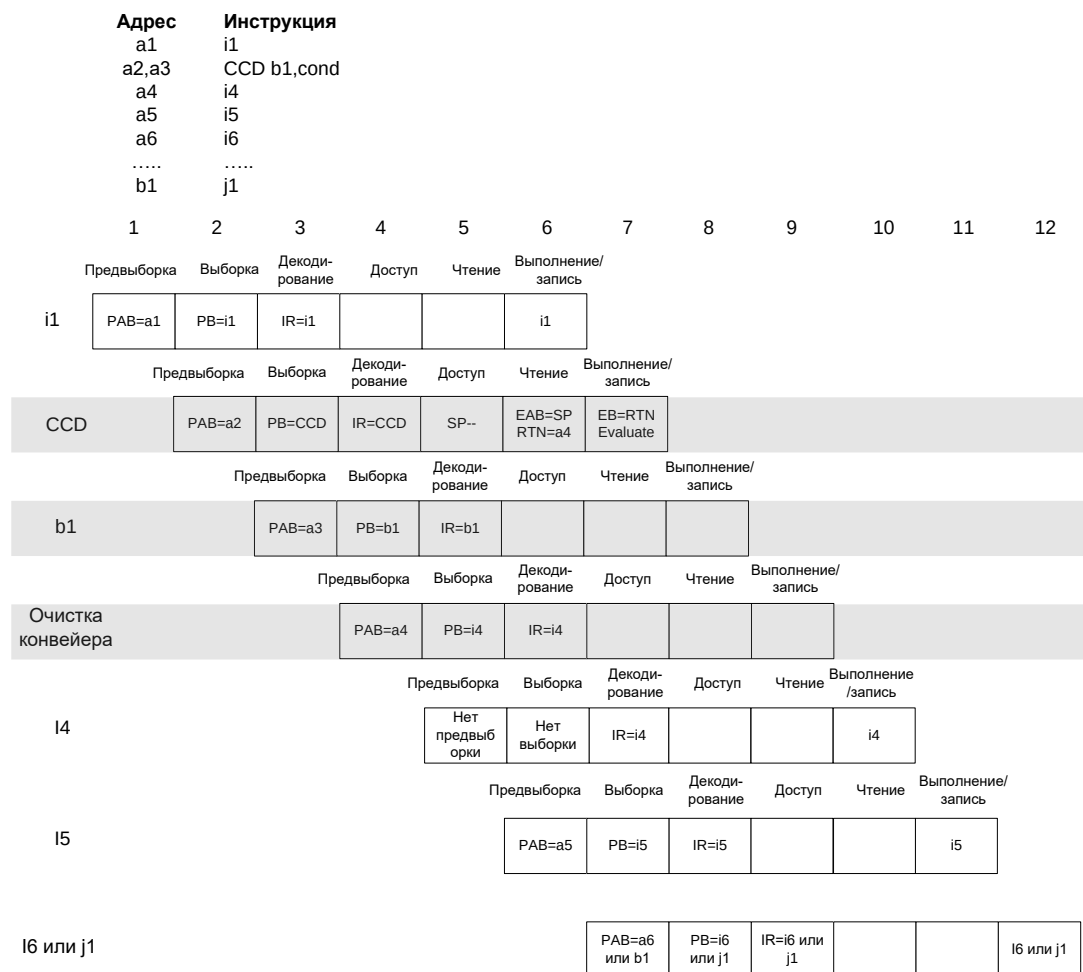


Рисунок 19-26 – Инструкция задержанного условного вызова функции (CCD) в конвейере

Следующие примеры показывают поведение конвейера в течение выполнения инструкции условного перехода (BC) и инструкции задержанного условного перехода (BCD).

Поведение инструкций условного перехода (BC) и задержанного условного перехода (BCD) в конвейере подобно инструкциям CC и CCD, соответственно. Различие в том, что в этом случае адрес возврата не записывается в стек. Инструкция BC использует три или пять циклов при выполнении, в зависимости от того, есть переход или нет. Инструкция BCD выполняется в три цикла.

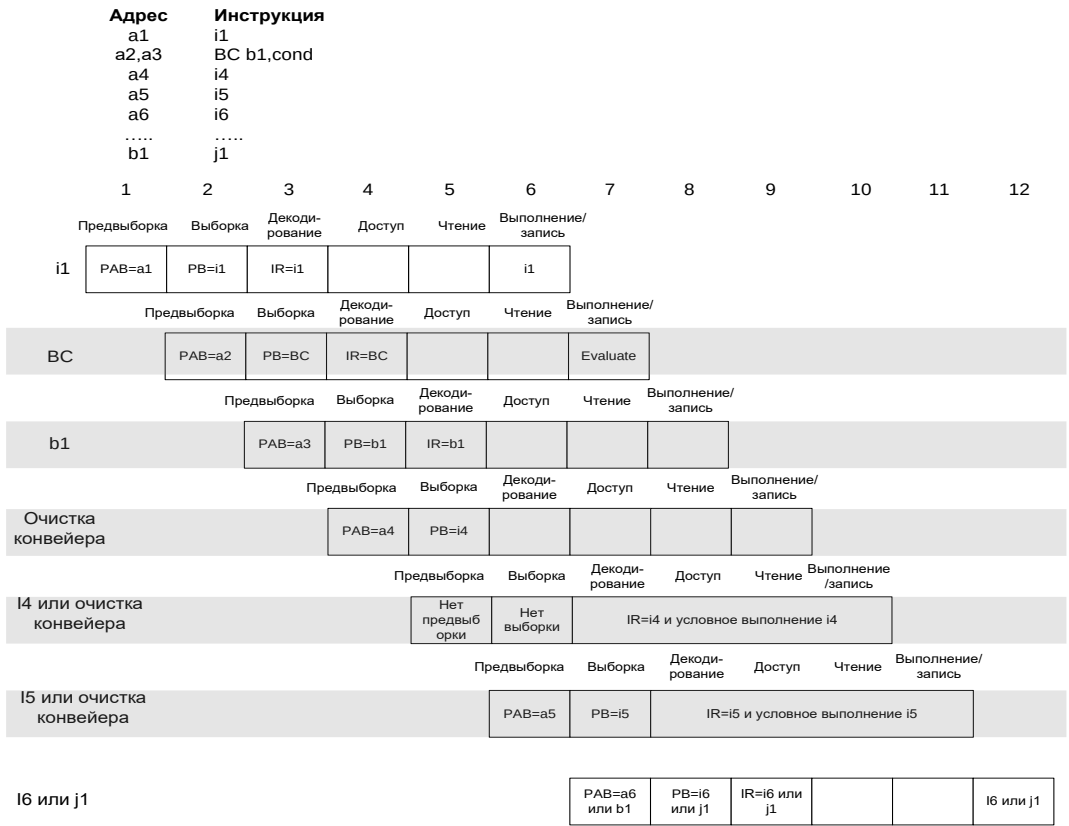


Рисунок 19-27 – Команда условного перехода (BC) в конвейере

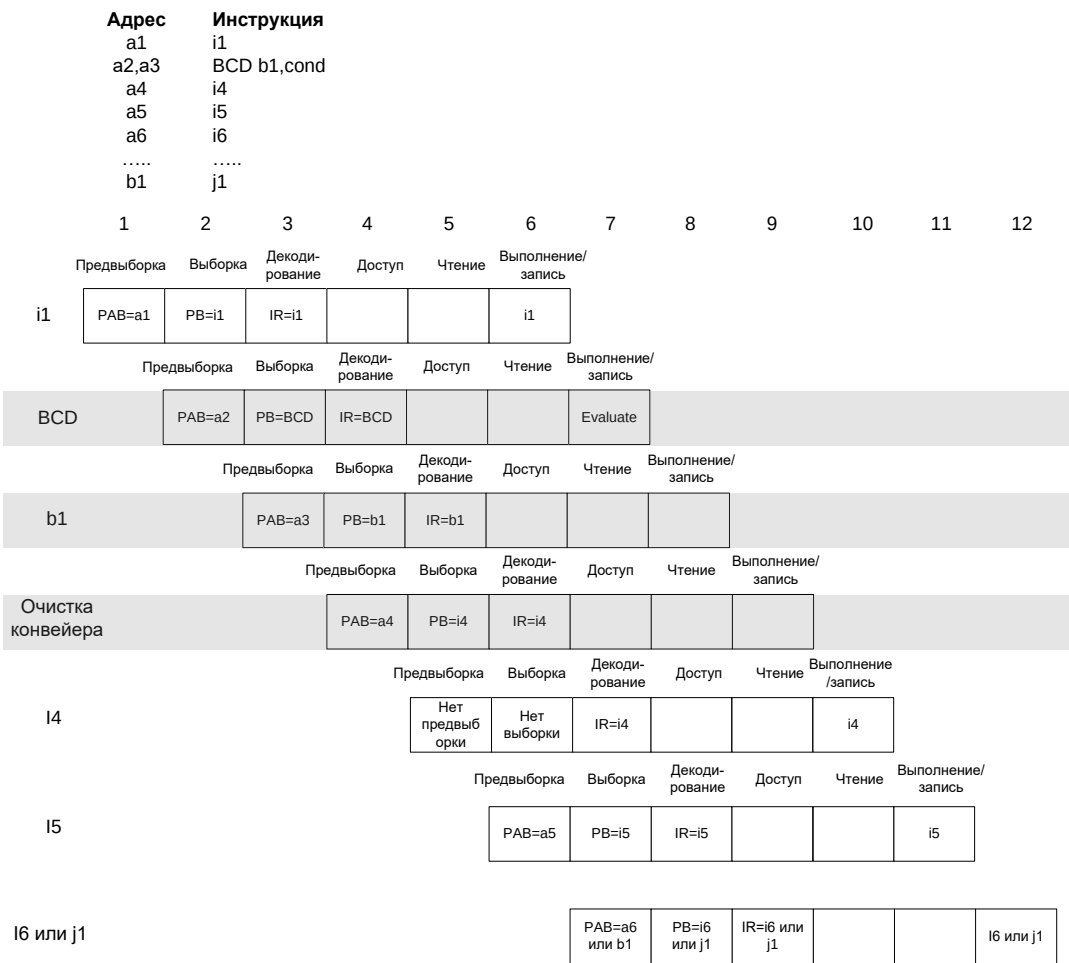


Рисунок 19-28 – Команда задержанного условного перехода (BCD) в конвейере

19.3.7 Прерывания и конвейер

Если прерывание обслужено в конце цикла 3, инструкция INTR автоматически устанавливается в стадии декодирования конвейера в течение следующего цикла (4). Инструкция i2 не декодируется, поскольку инструкция INTR установлена в конвейере на этом этапе. В течение следующих трех циклов, инструкции, которые уже были до этого декодированы, выполняются. Циклы 7, 8, и 9 используются инструкцией INTR. Первая инструкция в ISR, RETFD, выполняется в цикле 10. Циклы 11 и 12 используются двумя однословными инструкциями, которые заполняют щель задержки инструкции RETFD. В следующем цикле, выполняется инструкция i2, завершая возврат из ISR.

Как показано на рисунке, прерывание добавляет только три цикла (количество циклов необходимых для перехода на ISR). Возврат из прерывания использует только один цикл, поскольку RETFD – единственный цикл инструкции. Поскольку только четыре слова зарезервированы для каждого прерывания в таблице векторов прерывания, если ISR требует более, чем четыре слова инструкции, она (программа обработки прерывания) должна быть расположена где-нибудь еще. В этом случае, в таблице векторов надо занести инструкцию типа перехода. Это приводит к немного большим накладным расходам на обработку прерывания.

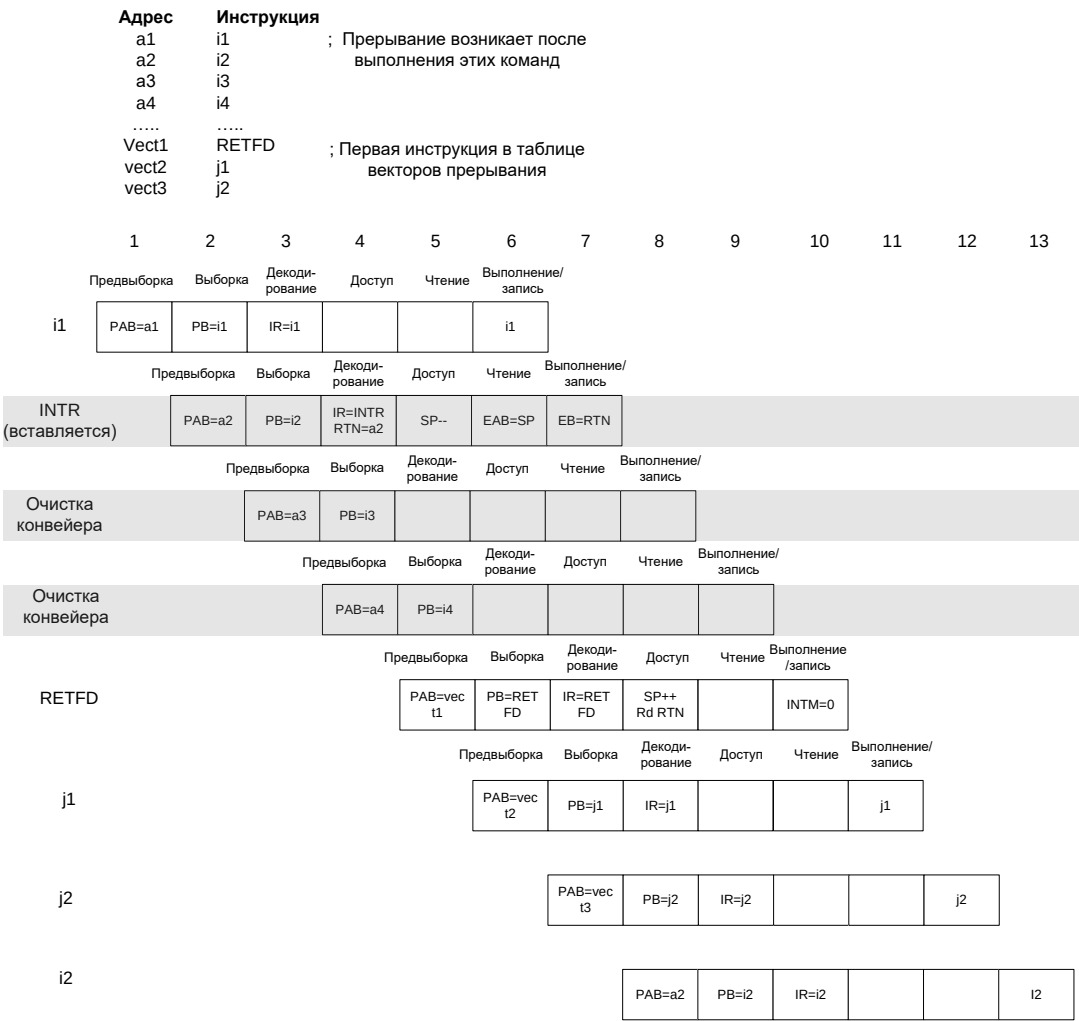


Рисунок 19-29 – Ответ конвейера на прерывание

19.4 Адресация данных

Микропроцессор ЦОС предлагает семь основных способов адресации:

- непосредственная адресация использует фиксированное значение в команде;
- абсолютная адресация использует фиксированный адрес в команде;
- аккумуляторная адресация использует аккумулятор, чтобы обратиться к памяти программ как к данным;
- прямая адресация использует семь бит команды как смещение относительно DP или SP. Смещение плюс значение DP или SP определяет исполнительный адрес в памяти данных;
- косвенная адресация использует вспомогательные регистры, чтобы обратиться к памяти;
- адресация регистра отображенного в память изменяет регистры, отображенные в памяти, не затрагивая текущее значение DP(data pointer) или текущее значение SP (stack pointer);
- стековая адресация управляет добавлением и удалением элементов системного стека.

19.4.1 Непосредственная адресация

В непосредственной адресации, синтаксис команды содержит конкретное значение операнда. Два типа значений могут быть закодированы в команде:

- короткие непосредственные значения могут быть 3, 5, 8, или 9 битовыми по длине;
- 16-разрядные непосредственные значения - всегда 16 битовые.

Непосредственные значения могут быть закодированы в первом или втором слове инструкции. 3-, 5-, 8- или 9-битовые значения закодированы в первом слове команды; 16-разрядные значения закодированы во втором слове команды. Длина непосредственного операнда, закодированного в команде, зависит от типа используемой команды. Таблица 19-3 дает список команд, которые могут закодировать непосредственные операнды в своем слове команды. Таблица также дает число разрядов, которое может быть закодировано в команде.

Таблица 19-3 – Команды, разрешающие непосредственную адресацию

3- и 5-разрядные константы	8-разрядная константа	9-разрядная константа	16-разрядная константа	
LD	FRAME	LD	ADD	ORM
	LD		ADDM	RPT
	RPT		AND	RPTZ
			ANDM	ST
			BITF	STM
			CMPM	SUB
			LD	XOR
			MAC	XORM
			OR	

Синтаксис непосредственной адресации использует знак (#), предшествующий значению или символу, чтобы указать, что это – непосредственное значение. Например, чтобы загрузить сумматор значением 80 в шестнадцатеричной системе счисления, Вы написали бы:

LD #80h, A

Рисунок 19-30 и Рисунок 19-31 используют команду RPT, чтобы показать, как непосредственное значение закодировано в командах, которые используют непосредственную адресацию. Код операции в команде закодирован в старшей половине команды: биты 8-15 1-ого слова инструкции, биты 0-15 старшего слова кодирования с 2 словами. Значение константы находится в остальной части кодового пространства.

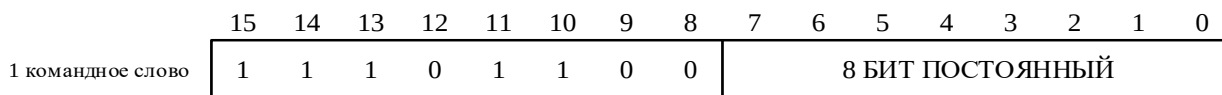


Рисунок 19-30 – Команда RPT с короткой непосредственной адресацией



Рисунок 19-31 – Команда RPT с 16-разрядной непосредственной адресацией

19.4.2 Абсолютная адресация

Это четыре типа абсолютной адресации:

- адресация dmad (data-memory address):
 - MVDK Smem, dmad
 - MVDM dmad, MMR
 - MVKD dmad, Smem
 - MVMD MMR, dmad
- адресация pmad (program-memory address):
 - FIRS Xmem, Ymem, pmad
 - MACD Smem, pmad, src
 - MACP Smem, pmad, src
 - MVDP Smem, pmad
 - MVPD pmad, Smem
- адресация PA (port address):
 - PORTR PA, Smem
 - PORTW Smem, PA
- адресация *(lk).

Адресация dmad

dmad (data-memory address) – Адресация к памяти данных (dmad) использует определенное значение, указывающее адрес в пространстве данных. В синтаксисе для обращения dmad используется символ или число, определяющее адрес в пространстве

данных. Например, чтобы скопировать значение, содержащееся в адресе, обозначенном как SAMPLE в пространстве данных в ячейку памяти в пространстве данных, на которое указывает AR5, Вы должны написать:

MVKD SAMPLE, *AR5

В этом примере адрес, на который ссылается SAMPLE-значение типа dmad.

Адресация pmad

Адресация к памяти программ (pmad-program-memory address) использует значение, определяющее адрес в пространстве программы. В синтаксисе обращения pmad используется символ или число, определяющее адрес в пространстве программы. Например, скопировать значение из программной памяти с меткой TABLE в ячейку памяти в пространстве данных, помеченную как AR7, Вы будете писать:

MVPD TABLE, *AR7-

В этом примере, адрес, на который ссылается TABLE – значение типа pmad.

Адресация PA

Адресация к порту (PA – port address) использует значение, определяющее внешний адрес порта ввода – вывода. Синтаксис для PA адресации, использует символ или число, определяющее адрес порта. Например, чтобы скопировать значение с порта ввода – вывода FIFO в память данных, на которое указывает AR5, Вы написали бы:

PORTR FIFO,*AR5

В этом примере, FIFO – это обращение к адресу порта.

Адресация *(lk)

Адресация * (lk) используется для определения адреса в пространстве данных.

Синтаксически * (lk) адресация определяется использованием символа или числа определяющего адрес в пространстве данных. Например, чтобы загрузить аккумулятор значением, содержавшимся в адресе BUFFER в пространстве данных, Вы бы написали:

LD *(BUFFER), A

Синтаксис для * (lk) адресация таков, что позволяет всем командам, которые используют Smem, осуществить доступ к любому месту в пространстве данных, не изменяя значение регистра DP или инициализации регистра AR. Когда используется эта форма абсолютной адресации, длина команды расширена одним словом. К примеру, однословная инструкция стала бы двухсловной, а двухсловная станет трехсловной. Добавление одного слова к команде затрагивает ее практичность и задержку в тактах.

Примечание – Команды использующие * (lk) форму абсолютной адресации не могут быть использованы с повторением единственной команды (RPT,RPTZ).

19.4.3 Аккумуляторная адресация

Адресация через аккумулятор использует аккумулятор как адрес. Этот способ адресации используется, чтобы использовать память программ как память данных. Две команды позволяют Вам использовать значение аккумулятора как адресного регистра:

- READA Smem
- WRITA Smem

READA перемещает слово памяти программ, определенного аккумулятором А к месту памяти данных, определенному единственным операндом команды (Smem-single data-memory).

WRITA передает слово из памяти данных, определенного операндом Smem команды к месту памяти программы, определенному аккумулятором А. В режиме повторения, может использоваться приращение, чтобы увеличить адрес определенный значением аккумулятора А.

19.4.4 Прямая адресация

В режиме прямой адресации, команда содержит младшие семь бит адреса памяти данных (dma). 7-битовый dma – смещение адреса, которое объединено с базовым адресом, с указателем страницы данных (DP), или с указателем вершины стека (SP), чтобы сформировать 16-разрядный адрес памяти данных. Используя эту форму адресации, Вы можете обратиться к любой из 128 ячеек памяти в произвольном порядке, не изменяя значение регистров DP или SP.

Примечание – Прямая адресация не единственный метод адресации по смещению. Однако, преимущество этого режима в том, что это кодирует каждую команду с адресом в единственном слове. DP или SP могут быть объединены со смещением dma, чтобы генерировать исполнительный адрес. Бит режима компилятора (CPL-compiler mode bit), расположенный в регистре состояния ST1, выбирает, какой метод используется для генерации адреса:

– когда $CPL = 0$, dma поле сочленяется с девятибитовым полем DP, чтобы сформировать 16-разрядный адрес памяти данных.

– когда $CPL = 1$, dma поле складывается (положительное смещение) с SP, чтобы сформировать 16-разрядный адрес памяти данных.

Синтаксис для прямой адресации использует символ или число, чтобы определить значение смещения. Например, чтобы добавить содержание ячейки памяти SAMPLE к аккумулятору В, при условии, что правильный базовый адрес находится в DP ($CPL = 0$) или SP ($CPL = 1$), Вы написали бы:

ADD SAMPLE, В

Младшие семь бит адреса SAMPLE сохранены в слове команды.

Рисунок 19-32 показывает формат кода операции для команд, использующих прямую адресацию.

Таблица 19-4 описывает биты команды прямой адресации.

Рисунок 19-33 иллюстрирует, как сформирован 16-разрядный адрес данных.

15-8	7	6 - 0
Код операции	I = 0	dma

Рисунок 19-32 – Формат команд прямой адресации

Таблица 19-4 – Краткое описание битов команды прямой адресации

Бит	Название	Функция
15 – 8	Код операции	Это восьмиразрядное поле содержит код операции команды
7	I	I=0, режим адресации, используемый командой, является режимом прямой адресации
6 – 0	dma	Это семиразрядное поле содержит адрес смещения памяти данных для команды

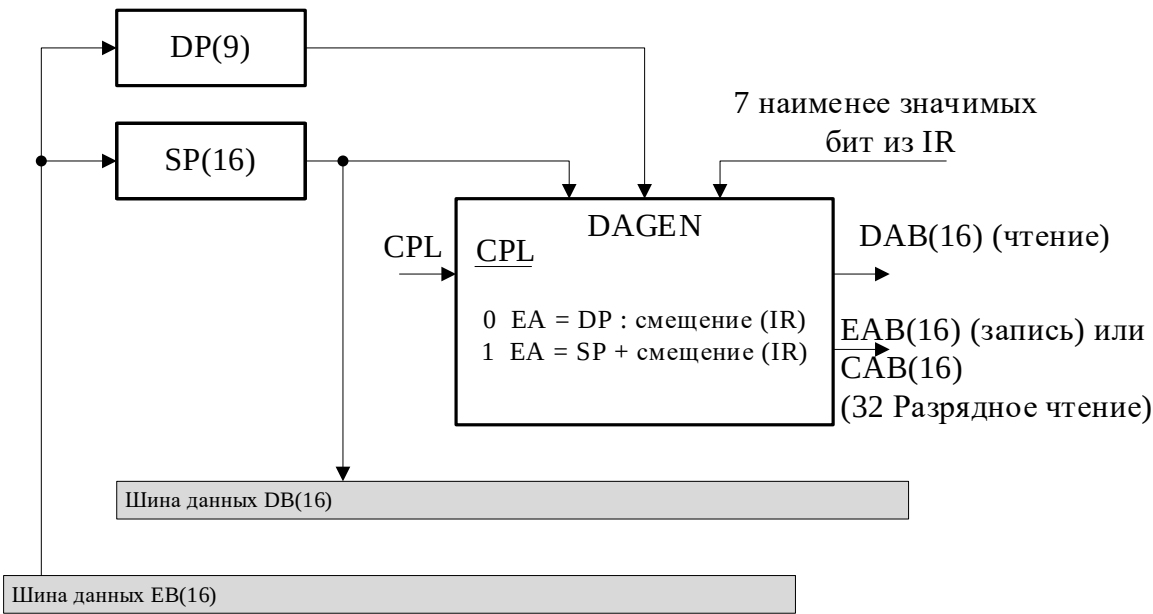


Рисунок 19-33 – Блок-схема прямой адресации

Прямая адресация со ссылкой на DP

В прямой адресации, ссылаемой на DP, семибитовое поле dma в команде сочленяется (конкатенация) с девятибитовым значением в DP, чтобы сформировать адрес. Рисунок 19-34 показывает, как два значения составляют исполнительный адрес.

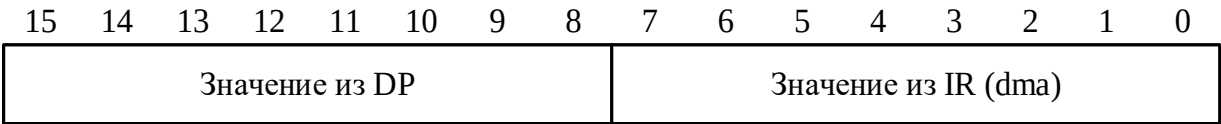


Рисунок 19-34 – Прямая адресация, связанная с DP

Прямая адресация со ссылкой на DP делит память на 512 страниц, потому что диапазон DP от 0 до 511 ($2^9 - 1$). Каждая страница имеет 128 адресов, потому что диапазон dma от 0 до 127 ($2^7 - 1$).

Другими словами, DP указывает на одну из 512 возможных страниц памяти данных с 128 словами; dma указывает на определенное место в пределах этой страницы. Единственное различие между доступом к месту 0 на странице 1 и к месту 0 на странице 2 проявляется в значении DP.

Регистр DP загружается командой LD.

Прямая адресация со ссылкой на SP

В прямой адресации, связанной с SP, семибитовый dma в регистре команды складывается как положительное смещение со значением SP, чтобы сформировать эффективный 16-разрядный адрес памяти данных. Рисунок 19-35 показывает, как два значения объединяются, чтобы сформировать результирующий адрес.



Рисунок 19-35 – Прямая адресация, связанная с SP

SP указывает на любой адрес в памяти. dma указывает на определенное место на странице, разрешая Вам обратиться к непрерывному в 128 слов (27 – 1) блоку в памяти от любого базового адреса. Через регистр SP можно также добавить или удалить элементы из стека. См. пункт 19.4.6 «Адресация регистра отображенного в памяти».

19.4.5 Косвенная адресация

В косвенной адресации, к любому месту в пространстве данных в 64К-слов можно обратиться через 16-разрядный адрес, содержащийся во вспомогательном регистре. Микропроцессор имеет восемь 16-разрядных вспомогательных регистров (AR0-AR7). Косвенная адресация используется главным образом тогда, когда есть потребность доступа к последовательным местам в памяти с фиксированным шагом.

Когда к памяти обращаются с косвенной адресацией, адрес во вспомогательном регистре может быть произвольно изменен декрементом, инкрементом, смещением, или индексом. Специальные режимы предлагают циклическую адресацию и с реверсом разрядов адреса. Регистр размера циклического буфера (BK) используется в циклической адресации. Регистр AR0 используется для индексной адресации и режима с битреверсной адресацией и в дополнение к этому, может быть использован для указания на память, как делают и другие вспомогательные регистры.

Косвенная адресация достаточно гибка не только, чтобы читать или писать единственный 16-разрядный операнд памяти данных в одной команде, но также и для обращения к двум операндам в памяти данных в одной команде. Доступ к двум операндам в памяти данных включает чтения двух независимых ячеек, чтение и запись двух последовательных ячеек, и чтение одной ячейки, объединенной с записью в ячейку памяти.

Адресация к единственному операнду

Рисунок 19-36 показывает формат команды с косвенным обращением для единственного (Smem) операнда. Таблица 19-5 описывает биты команды.

15-8	7	6 - 3	2 - 0
Код операции	I = 0	MOD	ARF

Рисунок 19-36 – Формат команд косвенной адресации для операнда памяти данных одиночного доступа

Таблица 19-5 – Косвенная адресация одного операнда

Биты	Название	Функции
15-8	Opcode	Это восьмибитовое поле содержит код операции
7	I	При единичном значении используется косвенная адресация
6-3	MOD	Это четырех битовое поле определяет тип косвенной адресации (см.далее)
2-0	ARF	Это поле определяет вспомогательный регистр, используемый при адресации. ARF зависит от бита совместимости в статусном регистре 1 (ST1). CMPT = 0 – Стандартный режим. В стандартном режиме ARF всегда определяет вспомогательный регистр, независимо от значения в ARP (auxiliary register pointer – поле в статусном регистре 0). ARP не изменяется. ARP должен всегда быть установлен в нуль, когда DSP – в этой режиме. CMPT = 1 – Режим совместимости. В режиме совместимости, ARP выбирает вспомогательный регистр, если ARF = 0. В противном случае, ARF выбирает вспомогательный регистр и величина ARF загружается в ARP, когда доступ завершен. *AR0 в командах ассемблера указывает вспомогательный регистр, выбранный с помощью ARP в режиме совместимости

Примечание – В некоторых случаях, два операнда данных могут быть выбраны сразу. Это требует других форматов инструкций. Далее описаны эти форматы.

ARAU и операция генерации адресов

Два модуля арифметики над вспомогательными регистрами (ARAU0 и ARAU1) оперируют содержимым вспомогательных регистров. ARAUs выполняют 16-разрядные беззнаковые арифметические операции над вспомогательными регистрами. Некоторые адреса могут быть получены, предварительной модификацией вспомогательных регистров.

Вспомогательные регистры могут быть:

- загружены непосредственным значением, используя команду STM;
- загружены через шину данных, записью во вспомогательный регистр, отображенный в памяти;
- изменены косвенной адресацией любой команды, которая поддерживает косвенную адресацию;
- изменены инструкцией модификации вспомогательного регистра (MAR);
- использованы как счетчики цикла в команде BANZ[D].

Примечание – Как правило, чтобы загрузить вспомогательные регистры, используются команды STM или MVDK. Обе эти команды позволяют следующей команде использовать новое значение в регистре. Другие команды, которые загружают новое значение через AR, вызывают ожидание в конвейере. Информация относительно конвейера и возможных конвейерных конфликтов дана в описании конвейера.

Рисунок 19-37 показывает ARAUs, используемый для генерации адреса в косвенном способе адресации с единственным операндом памяти данных. Как показано на рисунке, основные компоненты, используемые для генерации адресов в косвенной адресации – арифметические модули вспомогательных регистров (ARAU0 и ARAU1) и сами вспомогательные регистры (AR0-AR7).

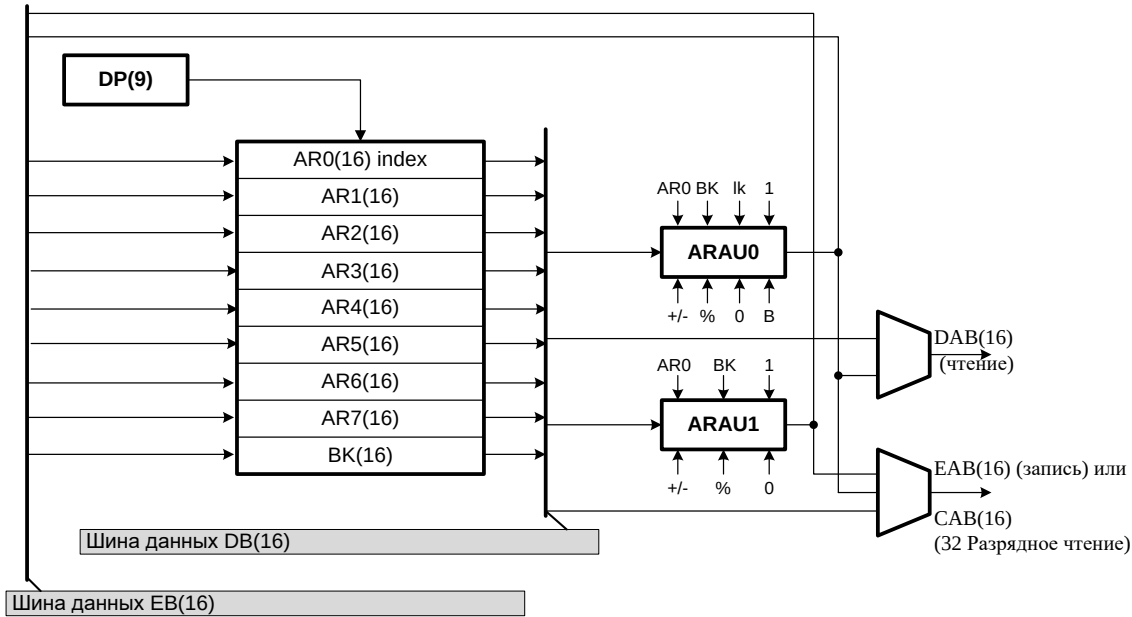


Рисунок 19-37 – Блок-схема косвенной адресации для операнда памяти данных одиночного доступа

Модификация адреса единственного операнда

Вы можете изменить адреса, которые Вы используете в командах прежде или после того, как к ним обратились, или Вы можете оставить их неизменными. Вы можете изменить их, инкрементируя или декрементируя адрес на единицу, добавляя 16-разрядное смещение или индексируя со значением в AR0. Эти три типа действий, предпринятые до или после доступа, плюс способы оставляющие адрес неизменным дают в общей сложности 16 типов адресации, которые определяются значением поля MOD, четырехбитовым полем в коде команды, использующем косвенную адресацию.

Таблица 19-6 дает список типов адресации одного операнда памяти данных, в соответствии со значением поля MD, синтаксис ассемблера и выполняемые действия для каждого типа.

Таблица 19-6 – Типы косвенной адресации

Поле MOD	Синтаксис операнда	Функция	Описание ¹⁾
0000 (0)	*ARx	addr = ARx	ARx содержит адрес памяти данных
0001 (1)	*ARx-	addr = ARx ARx = ARx -1	После доступа адрес в ARx декрементируется ²⁾

Поле MOD	Синтаксис операнда	Функция	Описание ¹⁾
0010 (2)	*ARx+	addr = ARx ARx = ARx + 1	После доступа адрес в ARx инкрементируется ²⁾
0011 (3)	*+ARx	addr = ARx + 1 ARx = ARx + 1	Адрес в ARx инкрементируется перед его использованием ^{2, 3, 5)}
0100 (4)	*ARx-0B	addr = ARx ARx = B(ARx - AR0)	После доступа AR0 вычитается из ARx с реверсным распространением переноса
0101 (5)	*ARx-0	addr = ARx ARx = ARx - AR0	После доступа AR0 вычитается из ARx
0110 (6)	*ARx+0	addr = ARx ARx = ARx + AR0	После доступа AR0 прибавляется к ARx
0111 (7)	*ARx+0B	addr = ARx ARx = B(ARx + AR0)	После доступа AR0 прибавляется к ARx с реверсным распространением переноса
1000 (8)	*ARx-%	addr = ARx ARx = circ(ARx - 1)	После доступа адрес в ARx декрементируется с циклической адресацией ²⁾
1001 (9)	*ARx-0%	addr = ARx ARx = circ(ARx - AR0)	После доступа AR0 вычитается из ARx с циклической адресацией
1010 (10)	*ARx+%	addr = ARx ARx = circ(ARx + 1)	После доступа адрес в ARx инкрементируется с циклической адресацией ²⁾
1011 (11)	*ARx+0%	addr = ARx ARx = circ(ARx + AR0)	После доступа, AR0 складывается с ARx (циклическая адресация)
1100 (12)	*ARx(lk)	addr = ARx + lk ARx = ARx	Сумма ARx и 16-битового длинного смещения используется для доступа к памяти данных. ARx не изменяется
1101 (13)	*+ARx(lk)	addr = ARx + lk ARx = ARx + lk	Адрес в ARx инкрементируется перед его использованием и складывается с 16-битовым знаковым длинным смещением. Результат затем используется для доступа к памяти данных ³⁾
1110 (14)	*+ARx(lk) %	addr = circ(ARx + lk) ARx = circ(ARx + lk)	Адрес в ARx инкрементируется перед его использованием и складывается с 16-битовым знаковым длинным смещением циклически. Результат затем используется для доступа к памяти данных ³⁾
1111 (15)	*(lk)	addr = lk	Беззнаковое 16 битное длинное смещение используется как абсолютный адрес памяти данных ^{3, 4)}

Примечания

¹⁾ ARx используется как адрес памяти данных, если не указан другой способ действия;

²⁾ Значение инкрементируемого и декрементируемого значения равно 1 для 16 битовых адресуемых слов и 2 для 32 битовых;

³⁾ Этот режим невозможен для регистров, отображаемых в памяти;

⁴⁾ Этот режим подробнее обсуждается далее;

⁵⁾ Этот режим позволяет осуществить доступ только для записи

Инкрементная и декрементная адресация (MOD = 0, 1, 2, или 3)

Во время использования AR вы можете изменить его значение, инкрементировать или декрементировать его.

Синтаксис использования AR без модификации, последующего декремента или инкремента, предварительного инкремента показан в таблице для MOD = 0, 1, 2, и 3, соответственно.

Предварительный инкремент (* +ARx) поддержан только в командах, которые обращаются к операндам в операции записи

Адресация по смещению (MD = 12 или 13)

Адресация по смещению – тип косвенной адресации, в которой предопределенное смещение или размер шага, добавлены к содержимому вспомогательного регистра. Есть две опции для адресации смещения. В обоих случаях, 16-разрядное длинное смещение, которое является частью команды, добавлено к значению во вспомогательном регистре, и результат используется для доступа к ячейке в памяти данных. В первом случае, вспомогательный регистр не обновляется. Во втором случае, вспомогательный регистр обновлен новым значением.

Этот тип адресации полезен при доступе к определенному элементу массива или структуры, особенно когда вспомогательный регистр не обновляется. Когда вспомогательный регистр обновляется, этот тип адресации особенно полезен для того, чтобы делать доступ в массиве с фиксированным шагом. Синтаксис для адресации по смещению AR без и с обновлением AR использует смещение так, как показано в таблице при MOD = 12 и 13, соответственно.

Примечания

1 Команда использующая адресацию по смещению не может быть повторена, используя повторение единственной команды.

2 Предварительная модификация 16-разрядным словом смещения (* +ARx (lk)) использует дополнительный цикл, потому что код команды имеет два или три слова. Последнее слово – смещение.

Индексная адресация (MD = 5 или 6)

Индексная адресация – тип косвенной адресации, в которой содержание AR0 складывается или вычитается из любого другого вспомогательного регистра ARx. Индексная адресация отличается от адресации по смещению тем, что индекс или размер шага могут быть определены во время выполнения программы. Поскольку индекс определяется во время выполнения программы, Вы можете легко внести изменения в размер шага. Индексная адресация также имеет преимущество перед адресацией по смещению – это не требует дополнительного слова команды. Синтаксис для этой адресации смотри в таблице для MD = 5 и 6, соответственно.

Циклическая адресация (MD = 8, 9, 10, 11, или 14)

Многие алгоритмы, типа свертки, корреляции и фильтрации на фильтрах с конечной импульсной характеристикой(FIR), требуют реализации кругового (циклического) буфера в памяти. В этих алгоритмах, циклический буфер – подвижное

окно в последовательности данных, содержащее последние данные. По мере прихода новые данные в буфер записываются поверх самых старых данных. Ключом к выполнению круговой адресации буфера является циклическая адресация.

Регистр размера циклического буфера (ВК) определяет размер кругового буфера. Круговой буфер размера R должен начинаться на N -битовой границе (то есть, младшие разряды базового адреса N кругового буфера должны быть нулевыми), где N – наименьшее целое число, которое удовлетворяет условию $2^N > R$. Значение R должно быть загружено в ВК. Например, круговой буфер с 31 словами должен начинаться в адресе, пять младших битов которого – 0 (то есть, XXXX XXXX XXX0 0000₂), и значение 31 должно быть загружено в ВК. Как второй пример, круговой буфер с 32 словами должен начинаться в адресе, шесть младших битов которого – 0 (то есть, XXXX XXXX XX00 0000₂), и значение 32 должно быть загружено в ВК. В некоторых приложениях, однако, может быть возможно использование бит-реверсивной адресации, размещением 2^N буфера на 2^N границе и реализацией круговой адресации.

Эффективный базовый адрес (EFB-effective base address) кругового буфера определяется обнулением младших N бит выбранного пользователем вспомогательного регистра (AR_x). Конец адреса кругового буфера (EOB-end of buffer address) определяется заменой младших N бит AR_x младшими N битами ВК. Индекс кругового буфера – просто младшие биты N AR_x, и шаг – количество, добавляемое или вычитаемое из вспомогательного регистра. Следуйте следующим трем правилам, когда Вы используете круговую адресацию:

- поместите первый (наименьший) адрес кругового буфера на $2N$ границе, где $2N$ больше чем размер кругового буфера.
- используйте шаг, меньше или равный размеру кругового буфера.
- в первый раз, когда обращаетесь к циклической очереди, вспомогательный регистр должен указывать на элемент в круговой очереди.

Алгоритм циклической адресации следующий:

```

If  $0 \leq \text{index} + \text{step} < \text{BK}$ :
    index = index + step.
Else
    if index + step  $\geq$  BK:
        index = index + step - BK.
    Else
        if index + step < 0:
            index = index + step + BK.
```

Круговая адресация может использоваться как для единственного операнда памяти данных так и двух операндов памяти данных. Когда в ВК нулевое значение, циклической модификации адреса не происходит. Это особенно полезно, когда двойной операнд должен выполнить модификацию адреса, эквивалентную AR_x+0.

Рисунок 19-38 иллюстрирует отношения между ВК, вспомогательным регистром (AR_x), началом кругового буфера, его вершиной и индекса в циклическом буфере.

Рисунок 19-39 показывает, как круговой буфер реализуется и каковы отношения между сгенерированными значениями и элементами в циклическом буфере.

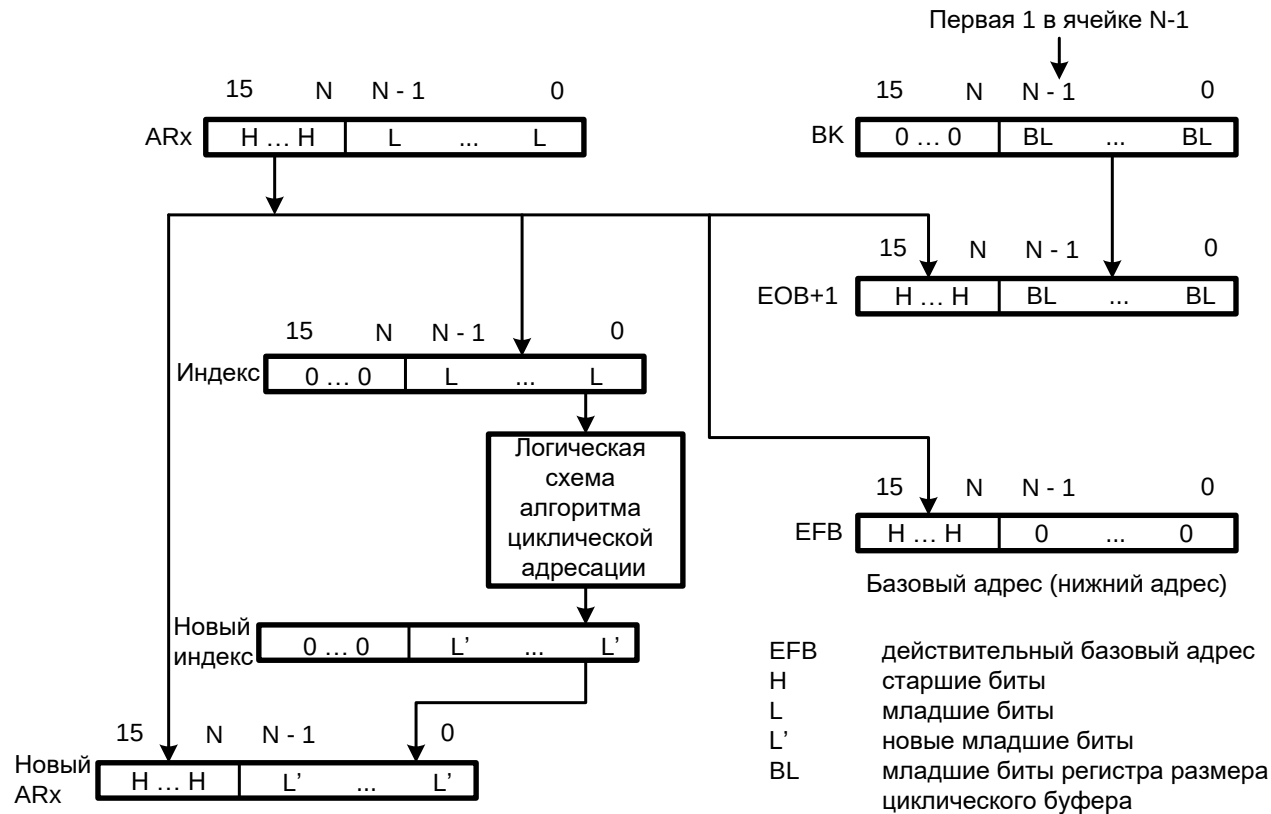


Рисунок 19-38 – Блок-схема циклической адресации

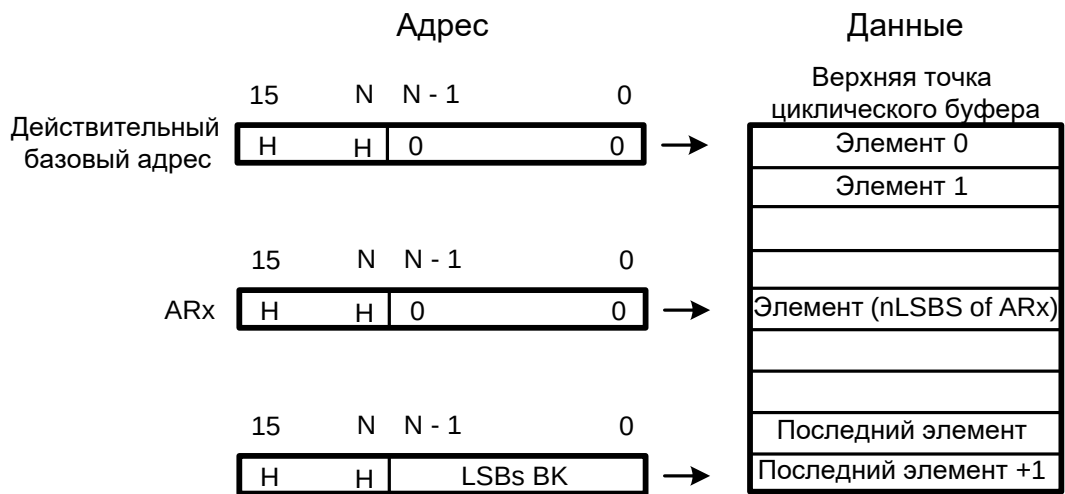


Рисунок 19-39 – Осуществление работы циклического буфера

Циклическая адресация типично использует декремент или инкремент на единицу ($MD = 8$ и 10) или декремент или приращение на величину индекса ($MD = 9$ и 11). Предварительная модификация 16-разрядным словом смещения ($* +ARx(lk) \%$) требует дополнительного кодового слова так, что код команды имеет два или три слова. Последнее слово – смещение. Команда, используя косвенную адресацию по смещению не может быть повторена, с использованием единственной повторяемой операции.

Синтаксис для каждого из пяти типов круговой адресации показан в таблице для $MD = 8, 9, 10, 11$ и 14 .

Бит-реверсивная адресация (MD = 4 или 7)

Бит-реверсивная адресация увеличивает скорость выполнения и память программы для алгоритмов быстрого преобразования Фурье (FFT), которые используют разные основания системы счисления. В этом способе адресации, AR0 определяет одну половину длины последовательности FFT. Значение, содержащееся в AR0 должно быть равным 2^{N-1} , где N – целое число, и размер FFT – 2^N . Вспомогательный регистр указывает на физическое местоположение значения данных. Когда Вы складываете AR0 с вспомогательным регистром, используя бит-реверсивную адресацию, адрес генерируется бит-реверсивным способом, с битом переноса, распространяющимся слева-направо, вместо нормального способа справа-налево. Синтаксис для каждого из двух режимов бит-реверсивной адресации показан в таблице для MD 4 и 7, соответственно. Предположим, что вспомогательные регистры длиной в восемь бит, что AR2 представляет базовый адрес данных в памяти (01100000₂), и что AR0 содержит значение 00001000₂. Ниже приведена последовательность модификаций AR2 и получающихся значений AR2.

Последовательность изменений вспомогательных регистров в адресации с инвертированием разрядов:

*AR2+0B ;AR2 = 0110 0000 (0th значение)
 *AR2+0B ;AR2 = 0110 1000 (1st значение)
 *AR2+0B ;AR2 = 0110 0100 (2nd значение)
 *AR2+0B ;AR2 = 0110 1100 (3rd значение)
 *AR2+0B ;AR2 = 0110 0010 (4th значение)
 *AR2+0B ;AR2 = 0110 1010 (5th значение)
 *AR2+0B ;AR2 = 0110 0110 (6th значение)
 *AR2+0B ;AR2 = 0110 1110 (7th значение)

Таблица 19-7 показывает соотношение битовой комбинации четырех младших бита AR2 в шаговой индексации и при использовании бит-реверсивной адресации.

Таблица 19-7 – Адресация с инвертированием разрядов

Шаг	Конфигурация бит	Конфигурация инвертированных бит	Шаг инвертированных бит
0	0000	0000	0
1	0001	1000	8
2	0010	0100	4
3	0011	1100	12
4	0100	0010	2
5	0101	1010	10
6	0110	0110	6
7	0111	1110	14
8	1000	0001	1
9	1001	1001	9
10	1010	0101	5
11	1011	1101	13
12	1100	0011	3
13	1101	1011	11
14	1110	0111	7
15	1111	1111	15

Адресация двойного операнда

Двойная адресация операнда памяти данных используется для команд, которые выполняют два чтения или единственное чтение и параллельно запись (обозначенный двумя вертикальными чертами, ||) в то же самое время. Эти команды – все длиной в одно слово и работают только в косвенном способе адресации. Два операнда памяти данных представлены Xmem и Ymem:

- Xmem – операнд чтения с доступом через шину D. Команды записи, например STH и STL с операцией сдвига, изменяют Xmem на операнд записи.
- Ymem используется как операнд чтения в командах с двойным чтением (доступ через шину C) или как операнд записи в командах с параллельным сохранением (доступ через шину E).

Если операнд источника и операнд приемника указывают на то же самое место, в командах с параллельной записью (например, ST|| LD), источник читается перед записью. Если команда двойного операнда (например, ADD), указывает на тот же самый вспомогательный регистр с различными способами адресации, определенными для обоих операндов, для адресации используется режим, определенный Xmod.

Рисунок 19-40 показывает формат косвенной адресации команды с двойным операндом памяти данных.

Таблица 19-8 описывает биты команды.

Поскольку только два бита доступны для выбора каждого вспомогательного регистра в этом режиме, только четыре из вспомогательных регистров могут использоваться, AR2 – AR5.

Таблица 19-9 показывает, какие Xar и Yar выбираются в качестве вспомогательных регистров.

15-8	7	6	5	4	3	2	1	0
Код операции	Xmod	Xar	Ymod	Yar				

Рисунок 19-40 – Формат команды косвенной адресации
для операндов памяти данных двойного доступа

Таблица 19-8 – Краткое описание битов команды адресации двойного операнда

Биты	Имя	Функция
15-8	Opcode	Это восьмиразрядное поле содержит код операции команды
7-6	Xmod	Это двухразрядное поле определяет тип косвенной адресации для доступа к Xmem операнду
5-4	Xar	Двухразрядное поле определяет вспомогательный регистр, который содержит адрес Xmem
3-2	Ymod	Это двухразрядное поле определяет тип косвенной адресации для доступа к Ymem операнду
1-0	Yar	Двухразрядное поле определяет вспомогательный регистр, который содержит адрес Ymem

Таблица 19-9 – Выбор вспомогательных регистров

Хар или Yар	Вспомогательный регистр
00	AR2
01	AR3
10	AR4
11	AR5

Рисунок 19-41 показывает, как при генерации адреса используется двойная адресация памяти данных.

Адресация двойного операнда памяти данных использует четыре вспомогательных регистра (AR2-AR5). ARAUs совместно с этими регистрами, обеспечивают возможность обратиться к двум операндам в единственном цикле.

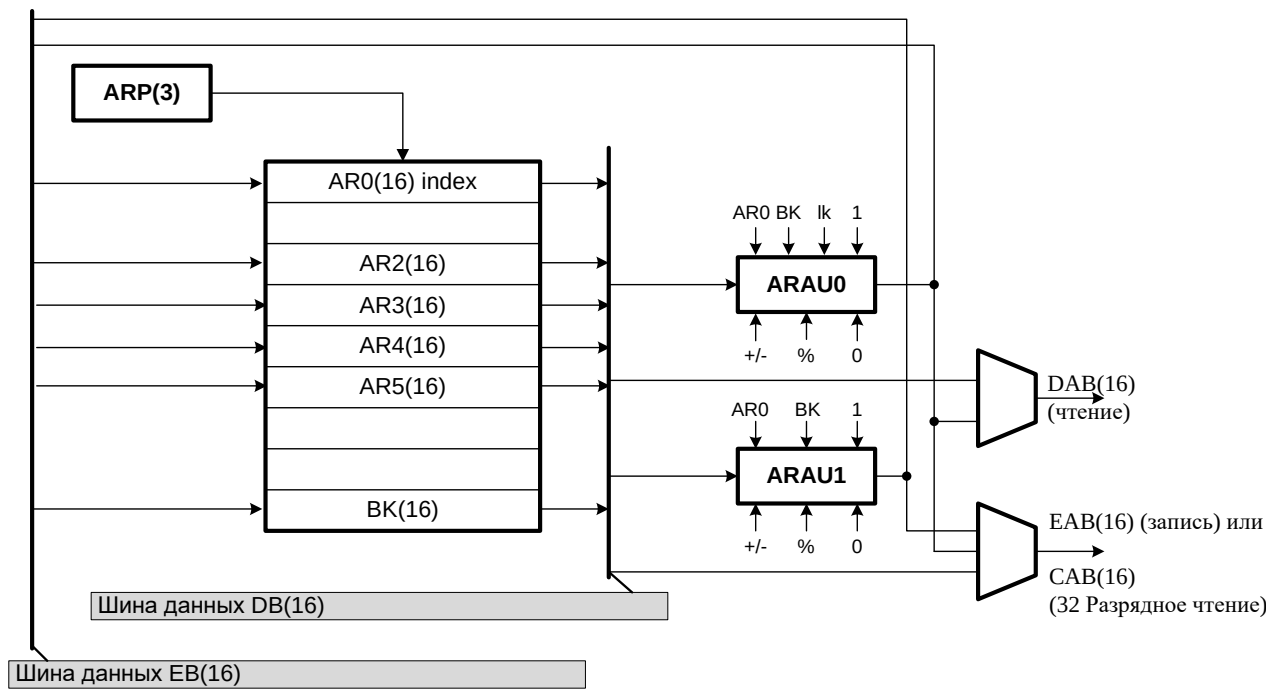


Рисунок 19-41 – Блок схема косвенной адресации для операндов памяти данных двойного доступа

Таблица 19-10 перечисляет типы двойной адресации операнда памяти данных, наряду со значением поля модификации (Xmod или Ymod), синтаксисом ассемблера и функцией для каждого типа.

Таблица 19-10 – Косвенная адресация с двойным операндом памяти данных

Xmod или Ymod	Синтаксис операнда	Функция	Описание ¹⁾
00	*ARx	addr = ARx	ARx является адресом памяти данных
01	*ARx-	addr = ARx ARx = ARx – 1	После доступа адрес ARx декрементируется
10	*ARx+	addr = ARx ARx = ARx + 1	После доступа адрес ARx инкрементируется
11	*ARx+0%	addr = ARx ARx = circ(ARx + AR0)	После доступа AR0 складывается с ARx циклически ²⁾

Xmod или Ymod	Синтаксис операнда	Функция	Описание ¹⁾
¹⁾ ARx используется как адрес памяти данных, если не указан другой способ действия ²⁾ Размер циклического буфера определен в регистре размера ВК			

В каждом случае, содержание вспомогательного регистра используется как операнд памяти данных. После использования адреса во вспомогательном регистре, ARAUs выполняют указанную математическую операцию. Отключая круговые модификации адреса, можно выполнить индексную адресацию или эквивалентно *ARx+0. Очистка ВК отключает циклическую адресацию.

В командах, которые выполняют чтение двойного операнда, если вспомогательный регистр определен полем Yar, при обращении к одному из регистров отображенных в памяти, прочитанное значение не будет представлять содержание регистра.

Инкремент и декремент адреса для двойного операнда (Xmod или Ymod = 0, 1 или 2)

Вы можете изменить AR, увеличиваясь или уменьшая его значение. Когда Xmod или Ymod = 0, ARx используется как адрес памяти данных без инкрементирования или декрементирования. Когда Xmod или Ymod = 1, ARx - декрементируется после того, как доступ осуществлен. Когда Xmod или Ymod = 2, ARx увеличивается после доступа.

Индексная адресация для двойного операнда (Xmod или Ymod = 3 и ВК = 0)

Когда Xmod или Ymod = 3 и ВК = 0, AR0 складывается с ARx после каждого доступа. Иначе говоря, индексная адресация двойного операнда точно такая же, как описана в ранее.

Циклическая адресация для двойного операнда (Xmod или Ymod = 3 и ВК ≠ 0)

Когда Xmod или Ymod = 3 и ВК ≠ 0, AR0 складывается с ARx используя циклическую адресацию после каждого доступа. Иначе говоря, циклическая адресация двойного операнда точно такая же, как описана ранее.

Команды с одним операндом, которые используют формат двойного операнда

Некоторые команды с одним операндом памяти данных используют адресацию двойных операндов памяти данных так, чтобы они размещались в одном слове для выполнения в одном цикле. В этих командах, доступен только Xmem, и поля Xmod и Xar определяют способ адресации для операнда. Четыре команды с одним операндом могут выполняться в одном цикле:

- BIT Xmem, BITC
- SACCD src, Xmem, cond
- SRCCD Xmem, cond
- STRCD Xmem, cond

- ADD Xmem, SHFT, src
- LD Xmem, SHFT, dst
- STH src, SHFT, Xmem
- STL src, SHFT, Xmem
- SUB Xmem, SHFT, src

ARP может использоваться в косвенной адресации. Это позволяет определить AR с помощью ARP для простой трансляции кода 'C2x/C20x/C24x/C5x устройств. С CMPT = 1 и ARF = 0, ARP используется, чтобы определить, какое AR используется для обращения к памяти. Рисунок 19-42 показывает, как APR индексирует вспомогательные регистры. При использовании ARP, микропроцессор отличается от 'C5-x в том, что когда микропроцессор использует AR, на которое указывает ARP, микропроцессор не обновляет ARP в той же самой командой. Таблица 19-11 показывает синтаксис ассемблера для 'C2x/C20x/C24x/C5x сравнивая его с данным микропроцессором.

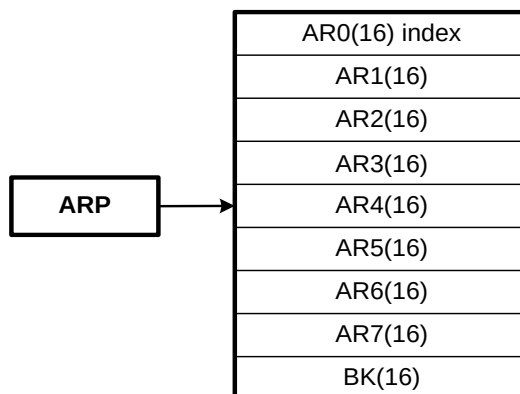


Таблица 19-11 – Сравнение синтаксисов ассемблера для устройств TMS320C2х/C20х/C24х/C5х и ‘54х

Синтаксис для ‘C2x/C20x/C24x/C5x	Синтаксис для ‘54x	Синтаксис для ‘C2x/C20x/C24x/C5x	Синтаксис для ‘54x
*	*AR0	*0-	*AR0-0
*_	*AR0-	*0+	*AR0+0
*+	*AR0+	*BR0-	*AR0-0B
		*BR+	*AR0+0B

15-8	7	6 - 3	2 - 0
Код операции	I = 1	MOD	ARF=000

322

Таблица 19-12 описывает биты команды в режиме ARP.

Таблица 19-12 – Инструкции с косвенной адресацией – совместимый режим

Биты	Имя	Функция
15-8	Opcode	Это восьми битовое поле содержит код операции
7	I	При единичном значении используется косвенная адресация
6-3	MOD	Это четырехбитовое поле определяет тип косвенной адресации (см. ранее определение 16-ти путей организации адресации)
2-0	ARF	Это поле определяет вспомогательный регистр, используемый при адресации. ARF зависит от бита совместимости в статусном регистре 1. CMPT = 0 – Стандартный режим. В стандартном режиме ARF всегда определяет вспомогательный регистр, независимо от значения в ARP (auxiliary register pointer – поле в статусном регистре 0). ARP не изменяется. ARP должен всегда быть установлен в нуль, когда DSP в этой режиме. CMPT = 1 – Режим совместимости. В режиме совместимости, ARP выбирает вспомогательный регистр, если ARF = 0. В противном случае, ARF выбирает вспомогательный регистр и величина ARF загружается в ARP когда доступ завершен. *AR0 в командах ассемблера указывает вспомогательный регистр выбранный с помощью ARP в режиме совместимости
Примечания 1 ARP должен всегда быть установлен в 0, когда DSP – в стандартном режиме (CMPT = 0); 2 При сбросе как ARP так и CMPT устанавливаются в 0 принудительно		

19.4.6 Адресация регистра отображенного в памяти

Адресация регистра отображенного в памяти используется, чтобы изменить регистры отображенные в памяти, не затрагивая или текущее значение указателя страницы данных (DP) или текущее значение указателя вершины стека (SP). Поскольку DP и SP не должны быть изменены в этом режиме, дополнительные расходы для записи в регистр минимальны. Регистр отображенный в памяти используется как в прямой так и косвенной адресации.

Рисунок 19-44 показывает, как сгенерированы адреса отображенные в памяти. Адреса генерируется посредством:

- Установкой в нуль девяти наиболее значительных битов (MSBs) адреса памяти данных к 0, независимо от текущего значения DP или SP, когда используется прямая адресация.
- Используя семь младших битов текущего значения вспомогательного регистра, когда используется косвенная адресация.

Примечание – В косвенной адресации, девять старших разрядов вспомогательного регистра установлены в 0 после операции.

Например, если AR1 используется для указания на регистр отображенный в памяти в способе адресации регистров отображенных в памяти, и он содержит значение FF25h, тогда AR1 указывает на регистр периода таймера (PRD), так как семь младших битов AR1 хранят 25 в шестнадцатеричной системе счисления, и адрес PRD как раз и равен 0025h. После выполнения, значение, остающееся в AR1 является 0025h.



Рисунок 19-44 – Блок-схема адресации отображаемого в памяти регистра

Примечание – В дополнение к регистрам, любая сверхоперативная память, расположенная на странице 0 данных может быть изменена при использовании адресации регистра отображенного в памяти.

Только восемь команд могут использовать адресацию регистра с отображенной памятью:

- LDM MMR, dst
- MVDM dmad, MMR
- MVMD MMR, dmad
- MVMM MMRx, MMRy
- POPM MMR
- PSHM MMR
- STLM src, MMR
- STM # lk, MMR

Примечание – Следующие режимы косвенной адресации недоступны для адресации регистров отображенных в памяти:

- *ARx(lk)
- *+ARx(lk)
- *+ARx(lk)%
- *(lk)

В этих случаях ассемблер выдает предупреждение.

19.4.7 Стековая адресация

Системный стек используется, чтобы автоматически сохранять счетчик команд в течение прерываний и вызовов подпрограмм. Он может также использоваться по вашему усмотрению, чтобы хранить дополнительные элементы контекста или передать значения данных. Стек заполняется от самого большого к самому малому адресу памяти. Процессор использует 16-разрядный регистр отображенный в памяти именуемый указателем вершины стека (SP), для адресации к стеку. SP всегда указывает на последний элемент, сохраненный в стеке.

Четыре команды обращаются к стеку, используя способ стековую адресацию:

- PSHD помещает значение памяти данных в стек;
- PSHM помещает регистр, отображенный в памяти в стек;
- POPD выталкивает значение памяти данных из стека;
- POPM выталкивает регистр, отображенный в памяти из стека.

Запись предекрементирует и выталкивание постинкрементирует адрес в SP. Рисунок 19-45 показывает пример состояния стека и SP до и после помещения X2 в стек (PSHD X2).

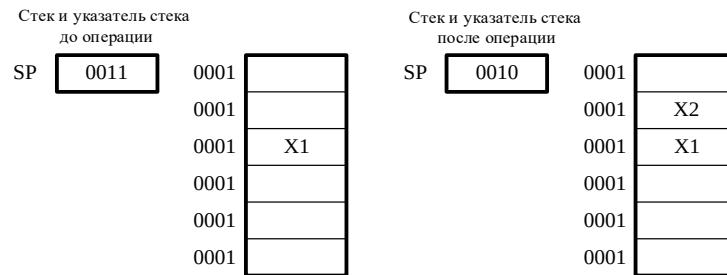


Рисунок 19-45 – Стек и указатель стека до и после операции записи в стек

Другие действия также влияют на стек и указатель стека. Стек используется в течение прерываний и вызова подпрограмм, чтобы сохранять и восстанавливать содержание PC. Когда произошел вызов подпрограммы или возникло прерывание, адрес возврата автоматически сохраняется в стеке, используя вталкивание в вершину. Инструкции использующие вызов подпрограмм и прерывания – CALA[D], CALL[D], CC[D], INTR, и TRAP.

Когда происходит возврат из подпрограммы, адрес возврата извлекается из стека, используя действие выталкивания и загружается в PC. Инструкции использующие возврат из подпрограмм - RET[D], RETE[D], RETEF[D], и RC[D].

Инструкция FRAME также влияет на стек. Эта инструкция добавляет короткий непосредственный операнд сдвига к указателю стека. Стек также используется в прямой адресации со ссылкой на SP (смотри ранее).

19.4.8 Типы Данных

Есть два основных типа данных для доступа к памяти в микропроцессоре: 16-бит и 32-бит. Большинство инструкций могут иметь доступ к 16-битовым данным. Доступ к 32-битовым данным, тем не менее, требует использование специальных инструкций, которые указаны в Таблица 19-13.

Таблица 19-13 – Инструкции, оперирующие с 32-битовыми словами

Команда	Описание
DADD	Сложение двойной точности/двойное 16-битное сложение с аккумулятором
DADST	Загрузка двойной точности и сложение или вычитание с T
DLD	Загрузка длинного числа в аккумулятор
DRSUB	Вычитание двойной точности из длинного числа
DSADT	Загрузка длинного числа и сложение или вычитание с T
DST	Запись аккумулятора как длинного слова
DSUB	Вычитание числа двойной точности из аккумулятора
DSUBT	Вычитание числа двойной точности из T

Для доступа к 16-битовому операнду, 16-битовое слово читается из памяти данных через шину D и записывается в память данных через шину E. Для доступа к 32-битовому операнду, шина C (для старшей части слова) и шина D (для младшей части слова) используются для чтения совместно. Тем не менее, поскольку только шина E используется для записи, операция записи (инструкция DST) выполняется в два цикла.

При доступе к 32-битовым словам, первое доступное слово рассматривается как наиболее значимое (старшая часть – MSW), тогда как второе доступное слово как младшая часть (LSW). Если исходный адрес доступа четный, то затем следует доступ ко второму слову – по нечетному адресу(большему на 1). Если сначала доступ по нечетному адресу, то затем доступ ко второму слову будет по меньшему четному адресу. Рисунок 19-46 показывает этот эффект.

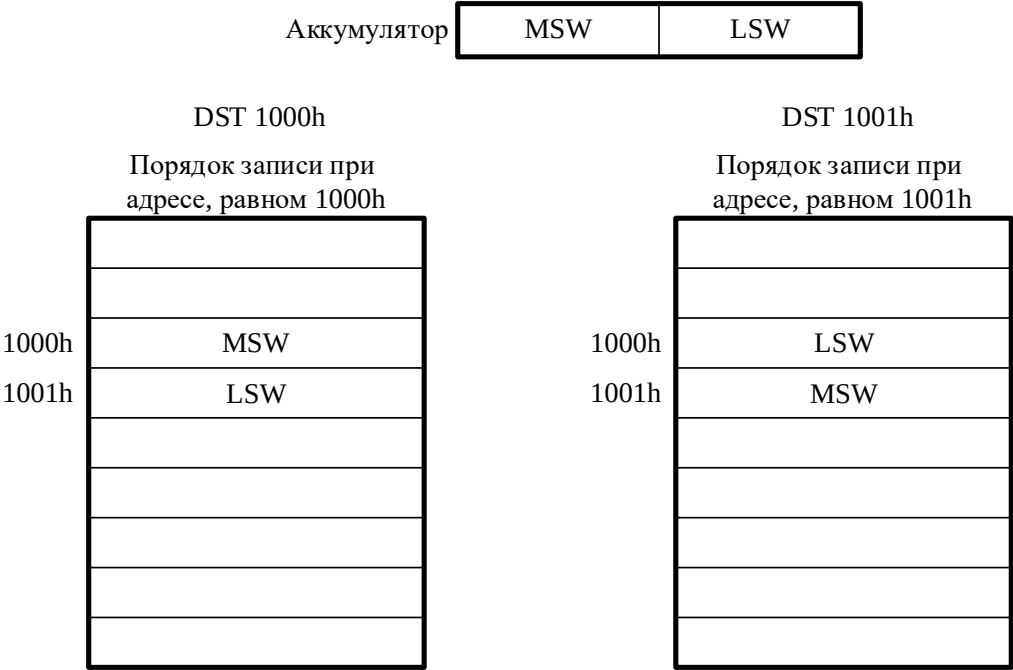


Рисунок 19-46 – Порядок слов в памяти

19.5 Адресация программ

В этой главе обсуждается, как генерируются адреса программной памяти, и какие адреса загружаются в счетчик команд (PC). Эта глава также описывает действия по управлению выполнением программ, которые влияют на загрузки в PC:

- Переходы;
- Вызовы подпрограмм;
- Возвраты из подпрограмм;
- Условные операции;
- Повторение инструкций или группы инструкций;
- Сброс аппаратуры;
- Прерывания.

Эти действия могут возникнуть в произвольной последовательности и произвести загрузку PC.

Описание действий в конвейере, а также прерываний, полезно в понимании действия с программным счетчиком микропроцессора.

Режим выключения питания останавливает выполнение программ.

Вектора сброса, прерывания и ловушек расположены в пространстве программы. Эти векторы подразумеваются программно заданными, чтобы процессор, попав в ловушку, загружал счетчик программы (РС) адресом захвата и выполнял код, определяемый вектором. Четыре слова зарезервированы по месту каждого вектора, чтобы разместить или задержанную (отсроченную) команду перехода, две однословные операции или одну двухсловную операцию, которые позволяют выполнять переход к соответствующей программе обработки прерывания.

При сбросе устройства, вектора сброса, прерывания и ловушек отображаются к адресам FF80h в пространстве программы. Однако, эти векторы могут быть повторно отображены к началу любой страницы с 128 словами в пространстве программы после сброса устройства. Это делается загрузкой указателя вектора прерывания (IPTR) в регистр PMST с соответствующим адресом границы страницы в 128 слов. После загрузки IPTR любой вектор пользовательского прерывания или ловушки отображается к новой странице в 128 слов. Например:

STM #05800h,PMST; Повторно отображение векторов, начиная с 5800 адреса.

Этот пример перемещения векторов прерывания в пространство программ с 05800-ого адреса. Любое последующее прерывание (за исключением сброса устройства) выбирает его вектор прерывания от этого нового местоположения. Например, если, после загрузки IPTR, возникает INT2, вектор программы обработки прерывания выбирается, начиная с адреса 5848 в пространстве программы в противоположность местоположению FFC8h. Эта особенность облегчает перемещение необходимых векторов из аппаратного загрузчика и затем удаления ПЗУ из карты памяти. Как только системный код загружен в систему от резидентного загрузчика в ПЗУ, приложение перезагружает IPTR значения, указывая на новые вектора. В предыдущем примере, команда STM используется, чтобы изменить PMST. Отметим, что команда STM изменяет не только IPTR, но и другие биты состояния в регистре PMST.

В случае сброса ядра средствами DSP бит RST_DSP или RST_DSP_CPU регистр IPTR загружается значением 0xFFFF. Вектор сброса всегда выбирается с адреса FF80h в пространстве памяти программ. Кроме того, для микропроцессора, 128 слов зарезервированы в ПЗУ на кристалле в целях тестирования. Прикладной код, написанный для реализации на ПЗУ в кристалле должен резервировать эти 128 слов в адресах FF00h-FF7Fh в пространстве программы.

19.5.1 Генерация адреса программной памяти

Программная память содержит коды приложений, таблицы коэффициентов и непосредственные операнды. Микропроцессор может адресовать до 64K слов программной памяти, используя шину адресов программы (PAB).

Логика генерации программного адреса (PAGEN) генерирует адрес, используемый для доступа к инструкциям, таблицам коэффициентов, 16-битовые непосредственным операндам или любой другой информации загружаемой в программную память и устанавливает этот адрес на шине PAB.

PAGEN состоит из пяти регистров (см. Рисунок 19-47):

- счетчик программы (PC-programm counter);
- счетчик повторения (RC-repeat counter);
- счетчик повторения блоков (BRC-block-repeat counter);
- регистр стартового адреса повторения блока (RSA-block-repeat start address register);
- регистр конечного адреса повторения блока (REA-block-repeat end address register).

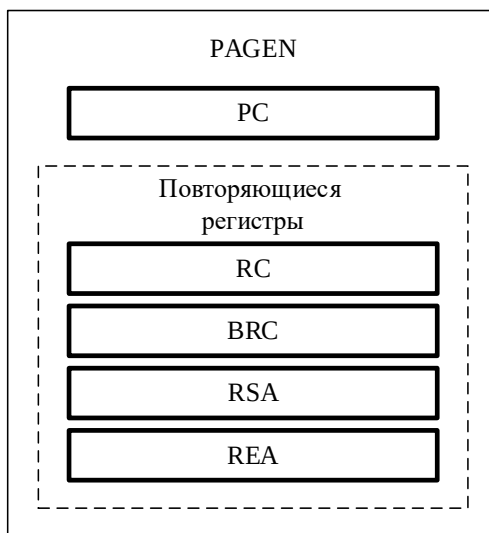


Рисунок 19-47 – Регистры логической схемы генерации адреса программ

Микропроцессор выбирает инструкции, устанавливая величину PC на PAB и читая соответствующее место в памяти. Когда команда прочитана из памяти, PC инкрементируется для следующей выборки. Если происходит прерывание, программы (например, переход, вызов подпрограммы, возврат из подпрограммы, прерывание или блочное повторение), соответствующий адрес загружается в PC. Инструкция, адресованная через PAB – затем загружается в регистр инструкции (IR).

Для того, чтобы улучшать исполнение определенных инструкций, устройство генерации программного адреса также используется для выборки операндов из программной памяти. Операнды выбираются из программной памяти, когда устройство читает из нее, или записываются в таблицу коэффициента, или передаются данные между пространством памяти программ и пространством памяти данных.

Некоторые инструкции, как например, FIRS, MACD, и MACP, используют программную шину, чтобы выбирать второй сомножитель.

19.5.2 Программный Счетчик (PC)

PC – 16-битный регистр, который содержит адрес внутренний или внешний программной памяти используемый, когда выбирается инструкция или когда 16-битовый непосредственный операнд или коэффициент таблицы доступен в программной памяти. Для адресации программной памяти адрес из PC помещается на PAB.

РС может быть установлен несколькими способами. Таблица 19-14 показывает, что загрузка в РС зависит от того, какое действие выполняется.

Таблица 19-14 – Загрузка адреса в РС

Операция	Адрес, загружаемый в РС
Сброс	РС загружается кодом FF80h
Последовательное выполнение	РС загружается значением РС + 1
Переход	РС загружается 16-битным непосредственным значением следующим за инструкцией
Переход из аккумулятора	РС загружается младшими 16-ти битами аккумулятора А или В
Цикл повторения блока	РС загружается начальным адресом повторения(RSA), когда РС+1 эквивалентно конечному адресу повторения (REA), при условии того, что BRAF = 1
Вызов подпрограммы	РС+2 заталкивается в стек, РС загружается 16 битным операндом, непосредственно следующим после инструкции. Инструкция возврата выталкивает вершину стека обратно на РС для возврата к прерванной последовательности
Вызов подпрограммы из аккумулятора	РС+1 вталкивается в стек, РС загружается младшими 16 битами аккумулятора А или В. Инструкция возврата выталкивает вершину стека обратно на РС для возврата к прерванной последовательности
Аппаратное прерывание, программное прерывание или ловушка	РС+1 вталкивается в стек, РС загружается адресом соответствующим вектору ловушки. Инструкция возврата выталкивает вершину стека обратно на РС для возврата к прерванной последовательности

19.5.3 Ветвления

Ветвления ломают последовательный поток инструкций, передавая управление в другую позицию в программной памяти. Следовательно, ветвления влияют на программный адрес сгенерированный и сохраненный в РС. Микропроцессор выполняет как безусловные, так и условные ветвления, и оба этих типа могут быть как с задержками, так и без задержек.

Безусловные переходы

Безусловный переход всегда выполняется, когда он возник. Во время выполнения, РС загружается определенным адресом программной памяти, и выполнение новой секции кода начинается с этого адреса. Адресом, загружаемым в РС, является второе слово инструкции перехода или младшие 16 бит аккумулятора (аккумулятор А или аккумулятор В).

Когда инструкция перехода достигает фазы выполнения в конвейере, следующие два слова инструкции уже выбраны. Как эти два слова инструкции используются, зависит частично от того, задержанный переход или нет:

– Не задержанный: Два слова инструкции удаляются из конвейера, чтобы они не были выполнены, и затем выполнение передается по адресу перехода.

– Задержанный: Одна двухсловная инструкция или две однословные инструкции, следующих за инструкцией перехода выполняются. Это позволяет Вам избегать опустошения конвейера, который требует дополнительных циклов.

Примечание – Два слова, следующих за задержанной инструкцией не могут быть командами, которые вызывают нарушение последовательности РС (переход, вызов подпрограммы, возврат или программное прерывание).

Таблица 19-15 показывает безусловные инструкции перехода в микропроцессоре и количестве циклов необходимых для выполнения этих инструкций (как не задержанных, так и задержанных). Задержанные инструкции используют на два цикла меньше, чем соответствующие не задержанные инструкции, поскольку они не сбрасывают конвейер.

Таблица 19-15 – Безусловный переход

Команда	Описание	Число циклов Не задерж./задерж.
B[D]	Загрузка РС адресом, определенным инструкцией	4/2
BACC[D]	Загрузка РС адресом, определяемым 16-тью младшими разрядами аккумулятора	6/4

Условные переходы

Условные переходы выполняются подобно безусловным переходам, но они выполняются только тогда, когда одно или более определенных программистом условий выполнены. Возможные условия даны в Таблица 19-16. Если все условия выполнены, РС загружается вторым словом инструкции перехода, которая содержит адрес перехода и выполнение продолжается с этого адреса.

Когда условия тестируются, два слова инструкции, следующих за условной инструкцией уже выбраны и находятся в конвейере. То, как эти два слова инструкции будут использованы, частично зависит от того задержанный или не задержанный переход:

– Не задержанный: Если все условия выполнены, эти два слова инструкции удаляются из конвейера и выполнение передается на адрес перехода. Если условия не выполнены, два слова инструкции выполняются вместо перехода.

– Задержанный: Одна 2-словная инструкция или две 1-словные инструкции, следующих за инструкцией перехода выполняются. Это позволяет Вам избежать сброса конвейера, что требует дополнительных циклов. Тестируемые условия не оказывают влияния на инструкции, следующие за задержанным переходом.

Примечание – Два слова, следующих за задержанной инструкцией не могут быть инструкциями, которые вызывают нарушение последовательности РС (переход, вызов подпрограммы, возврат или программное прерывание).

Таблица 19-16 показывает условные инструкции перехода и количество циклов необходимых для выполнения этих инструкций. Поскольку условные переходы используют условия, определяемые выполнением предшествующих инструкций,

условная инструкция перехода, BC[D], требует на один цикл больше, чем безусловный переход.

Таблица 19-16 – Команды условного перехода

Инструкция	Описание	Количество циклов	
		Условие выполнено/не выполнено	
		Не Задержанная	Задержанная
BC[D]	Загрузка PC адресом, определенным инструкцией, если условие, определенное в команде, выполнено	5/3	3/3
BANZ[D]	Загрузка PC адресом, определенным командой, если текущее значение вспомогательного регистра не эквивалентно нулю (полезно для организации циклов)	4/2	2/2

Таблица 19-17 – Команды длинных переходов

Команды	Описание	Количество циклов
		Не задержанная / Задержанная
FB[D]	Загрузка PC и XPC адресом, определенным в инструкции	4/2
FBACC[D]	Загрузка PC и XPC адресом, определенным в 23 разрядах соответствующего аккумулятора	6/4

19.5.4 Вызов процедур

Подобно переходам, вызов ломает последовательный поток инструкций, передавая управление в некоторую другую позицию в программной памяти. Тем не менее, в отличие от переходов, эта передача предполагается быть временной. При вызове подпрограммы или функции, адрес инструкции, следующей за вызовом, сохраняется в стеке. Этот адрес используется для возврата на прерванную программу и завершение ее выполнения.

Микропроцессор выполняет как безусловные, так и условные вызовы, и оба этих типа могут быть как не задержанными, так и задержанными.

Безусловные вызовы процедур

Безусловный вызов всегда выполняется, когда он возник. Когда вызов выполняется, PC загружается определенным адресом программной памяти и выполнение вызванной программы начинается по этому адресу. Адрес, загруженный в PC, может происходить из второго слова инструкции вызова или младших 16 битов аккумулятора (аккумулятор A или аккумулятор B). Прежде, чем PC будет загружен, адрес возврата сохраняется в стеке. После того, как подпрограмма или функция будут выполнены, инструкция возврата загружает PC адресом возврата из стека, и выполнение продолжается с инструкции, следующей за инструкцией вызова.

Когда безусловная инструкция вызова достигает фазы выполнения в конвейере, следующие два слова инструкции уже выбраны. То, как эти два слова инструкции будут использованы, зависит частично от того, задержанный или нет вызов процедуры:

– Не задержанный: Два слова инструкции удаляются из конвейера, чтобы они не были выполнены, адрес возврата сохраняется в стеке и затем выполнение продолжается с начала вызванной функции.

– Задержанный: Одна двухсловная или две однословные инструкции, следующих за инструкцией вызова выполняются. Это позволяет избежать сброса конвейера, что требует дополнительных циклов.

Примечание – Два слова, следующих за задержанной инструкцией не могут быть инструкциями нарушающими непрерывность значений РС (переход, вызов процедуры, возврат или программное прерывание).

Таблица 19-18 показывает безусловные инструкции вызова процедур в микропроцессоре (как не задержанных так и задержанных) и количество циклов необходимых для выполнения этих инструкций.

Задержанным инструкциям нужно на два цикла меньше, чем соответствующим не задержанным инструкциям, поскольку они не опустошают конвейер.

Таблица 19-18 – Команды безусловного перехода

Команда	Описание	Количество циклов выполнения
		Не задержанная / Задержанная
CALL[D]	В стеке размещается адрес возврата и затем в РС загружается адрес, определенный в инструкции	4/2
CALA[D]	В стеке размещается адрес возврата и затем в РС загружается адрес, определенный значением соответствующего аккумулятора	6/4

Условные вызовы

Условные вызовы действуют подобно безусловным вызовам, но они выполняют только когда один или многочисленные условия выполнены. Возможные условия даны в Таблица 19-19. Если все условия выполнены, РС загружается вторым словом инструкции вызова, который содержит стартовый адрес вызываемой функции. Перед переходом в указанную функцию, процессор сохраняет адрес инструкции, следующей за инструкцией вызова в стеке.

Функция должна закончиться вызовом инструкции, адрес которой берется из стека и загружается в РС, позволяя процессору продолжать выполнение прерванной программы.

Когда признаки условной инструкции вызова протестированы, два слова инструкции, следующих за инструкцией вызова уже выбраны в конвейер. То как эти два слова инструкции будут использованы, зависит частично от того, задержанный или нет вызов:

– Не задержанный: Если все условия выполнены, эти два слова инструкции удаляются из конвейера чтобы не быть выполненными, и затем управление передается в

начало вызванной функции. Если условия не выполнены, эти две инструкции выполняются вместо вызова.

– **Задержанный:** одна двухсловная или две однословные инструкции, следующие за инструкцией вызова всегда выполняются. Это позволяет избежать сброса конвейера, что требует дополнительных циклов. Условия тестирования не оказывают влияния на инструкциями, следующие за задержанным вызовом. Если условия не выполнены, процессор выполняет эти два слова инструкции вместо вызова.

Примечание – Два слова, следующих за задержанной инструкцией не могут быть инструкциями, которые вызывают нарушение последовательности РС (переход, вызов из подпрограммы, возврат или программное прерывание).

Таблица 19-19 показывает команду условного вызова и количество циклов необходимых на его выполнение. Поскольку есть цикл ожидания для установления условий, условная инструкция вызова, CC[D], требует на один цикл больше, чем безусловный вызов.

Таблица 19-19 – Условные вызовы

Команда	Описание	Количество циклов	
		Не задержанная	Задержанная
CC[D]	Размещает адрес возврата в стеке и затем загружает в РС адрес определенный в инструкции, если условия обозначенные в команде выполнены	5/3	3/3

19.5.5 Возвраты

Инструкции возврата обеспечивают способ продолжения последовательности инструкций обработки, которые были прерваны вызовом в другую функцию или подпрограмму обработки прерывания. Когда вызванная функция или подпрограмма обработки прерывания завершила свое выполнение, необходимо продолжить обработку с точки, следующей за вызовом или точки, в которой произошло прерывание. Инструкции возврата выполняют эти действия, выталкивая верхушку стека, которая содержит адрес следующей инструкции, которую необходимо выполнить, в программный счетчик (РС).

Микропроцессор выполняет как безусловный, так и условный возврат, и оба этих типа могут быть задержанными или не задержанными.

Безусловный возврат

Безусловный возврат всегда выполняется, когда он возникает. Когда возврат выполняется, РС загружается адресом возврата из стека и выполнение продолжается с инструкции, следующей за инструкцией, после которой произошел вызов функции или в точке, где произошло прерывание.

Когда безусловная инструкция возврата достигает фазу выполнения в конвейере, следующие два слова инструкции уже выбраны. То, как эти два слова инструкции используются, будет зависеть частично от того, не задержанная или задержанная инструкция возврата:

– **Не задержанная:** Два слова инструкции удаляются из конвейера, чтобы они не были выполнены, адрес возврата взят из стека, и затем выполнение продолжается с этого адреса.

– Задержанная: одна двухсловная или две однословные инструкции, следующие за инструкцией возврата выполняются. Это позволяет избежать сброса конвейера, что требует дополнительных циклов. Адрес возврата берется из стека.

Примечание – Два слова, следующих за задержанной инструкцией не могут быть инструкциями, которые вызывают нарушение последовательности PC (переход, вызов из подпрограммы, возврат или программное прерывание).

Таблица 19-20 показывает безусловные инструкции возврата в микропроцессоре (не задержанные и задержанные) и количество нужных циклов для выполнения этих инструкций. Задержанным инструкциям нужно на два цикла меньше, чем соответствующим не задержанным инструкциям.

Таблица 19-20 – Инструкции безусловного возврата

Инструкция	Описание	Количество циклов
		Не задержанная / Задержанная
RET[D]	Загрузка PC адресом возврата с вершины стека	5/3
RETE[D]	Загрузка PC адресом возврата с вершины стека и разрешение маскируемых прерываний	5/3
RETF[D]	Загрузка PC адресом возврата из RTN регистра и разрешение маскируемых прерываний	3/1

Разрешение прерываний в инструкциях RETE и RETF гарантирует, что возврат выполнится прежде, чем другое прерывание начнет обрабатываться.

Условные возвраты

Используя инструкцию условного возврата (RC-conditional return), Вы можете дать функции или программе обработки прерывания (ISR- interrupt service routine) более чем один из возможных путей возврата. Путь выбора зависит от обрабатываемых данных. Кроме того, Вы можете использовать условный возврат, чтобы избежать условного перехода на инструкцию возврата в конце функции или программы обработки прерывания.

Условные возвраты действуют подобно безусловным возвратам, но они выполняют только тогда, когда одно или более условий выполнены. Возможные условия даны в Таблица 19-21. Если все условия выполнены, процессор загружает адрес возврата из стека в PC и продолжает выполнение прерванной программы.

Условный возврат является однословной инструкцией; тем не менее, из-за возможности прерывания последовательности значений PC, он выполняется с тем же эффективным временем, что и условный переход или вызов программы.

Когда условия инструкции условного возврата протестированы, два слова инструкции, следующих за инструкцией возврата уже выбраны в конвейер. То, как эти два слова инструкции будут использованы, зависит частично от того задержанный возврат или нет:

– Не задержанный: Если все условия выполнены, эти два слова инструкции удаляются из конвейера, чтобы они не выполнялись, и тогда остается только выполнение вызова прерванной программы. Если условия не выполнены, две инструкции выполняются вместо возврата.

– Задержанный: Процессор выполняет две инструкции, которые следуют за инструкцией возврата. Это позволяет избежать сброса конвейера, что требует дополнительных циклов. Значения тестируемых условий не влияют на инструкции, следующие за задержанным возвратом.

Примечание – Два слова, следующих за задержанной инструкцией не могут быть инструкциями, которые вызывают нарушение последовательности PC (переход, вызов из подпрограммы, возврат или программное прерывание).

Таблица 19-21 показывает условную инструкцию возврата и количество циклов необходимых для выполнения этой инструкции.

Таблица 19-21 – Команды условного возврата

Инструкция	Описание	Количество циклов	
		Условия выполнились / не выполнились	
		Не задержанный	Задержанный
RC[D]	PC загружается адресом возврата из вершины стека, если условия определенные в команде выполнены	5/3	3/3

19.5.6 Условные операции

Микропроцессор включает инструкции, которые выполняются только тогда, когда одно или более условий выполнены. Таблица 19-22 перечисляет условия, которые Вы можете использовать с этими инструкциями и соответствующие символы операнда.

Таблица 19-22 – Условия для условных операций

Условие	Описание	Операнд
A = 0	Аккумулятор A равен нулю	AEQ
B = 0	Аккумулятор B равен нулю	BEQ
A ≠ 0	Аккумулятор A не равен нулю	ANEQ
B ≠ 0	Аккумулятор B не равен нулю	BNEQ
A < 0	Аккумулятор A меньше нуля	ALT
B < 0	Аккумулятор B меньше нуля	BLT
A ≤ 0	Аккумулятор A меньше или равен нулю	ALEQ
B ≤ 0	Аккумулятор B меньше или равен нулю	BLEQ
A > 0	Аккумулятор A больше нуля	AGT
B > 0	Аккумулятор B больше нуля	BGT
A ≥ 0	Аккумулятор A больше или равен нулю	AGEQ
B ≥ 0	Аккумулятор B больше или равен нулю	BGEQ
AOV = 1	Аккумулятор A переполнен	AOV
BOV = 1	Аккумулятор B переполнен	BOV
AOV = 0	Аккумулятор A не переполнен	ANOV
BOV = 0	Аккумулятор B не переполнен	BNOV
C = 1	Бит переноса C равен единице	C
C = 0	Бит переноса C равен нулю	NC
TC = 1	Флаг тест/управление равен единице	TC
TC = 0	Флаг тест/управление равен нулю	NTC
BIO\ low	BIO\ сигнал в низком состоянии	BIO
BIO\ high	BIO\ сигнал в высоком состоянии	NBIO

Условие	Описание	Операнд
none	Безусловная операция	UNC

Использование множественных условий

Множественные условия могут быть указаны как операнды условных инструкций.

Если указаны множественные условия, то все они должны быть выполнены для того чтобы инструкция выполнялась. Только определенные комбинации условий приемлемы (Таблица 19-23). Для каждой комбинации, условия должны быть выбраны из группы 1 или группа 2 следующим образом:

– Группа 1: Вы можете выбрать одно условие из категории А и одно условие из категории В. Два условия не могут быть из одной и той же категории. Например, Вы можете протестировать EQ и OV одновременно, но Вы не можете одновременно протестировать GT и NEQ. Аккумулятор должен быть тем же для обоих условий; Вы не можете протестировать условия для обоих аккумуляторов в одной и той же инструкции. Например, Вы можете протестировать AGT и AOV одновременно, но Вы не можете одновременно протестировать AGT и BOV.

– Группа 2: Вы можете выбрать одно условие из каждой из трех категорий (А, В, и С). Никакие два условия могут быть из одной и той же категории. Например, Вы можете одновременно протестировать ТС, С, и ВЮ, но Вы не можете протестировать NTC, С, и NC в одно и то же время.

Таблица 19-23 – Группирование множественных условий инструкций

Группа 1		Группа 2		
Категория А	Категория В	Категория А	Категория В	Категория С
EQ	OV	TC	C	BIO
NEQ	NOV	NTC	NC	NBIO
LT				
LEQ				
GT				
GEQ				

Условно выполняемая инструкция (XC)

Там, где есть условные фрагменты кода из одного или двух слов программы, Вы можете заменить переход одноцикловой условно выполняемой инструкцией (XC). Есть две формы для инструкции XC. Одна форма – условное выполнение однословной инструкции (XC 1, условие). Вторая форма – условное выполнение одной двухсловной инструкции или двух однословных инструкций (XC 2, условие). Условия для XC такие же, как и условия для условных переходов, вызовов и возвратов.

Примечание – Условие должно быть стабильным к моменту решения о том, следует ли исполнять одну или две команды следующие за XC. В оригинальном микропроцессоре это гарантировалось таким программированием, что если некоторое условие может измениться предыдущими командами, то эти команды должны быть на таком удалении от команды XC,

чтобы к моменту решения об исполнении анализируемые условия не менялись. О возможных ошибках программирования сообщал ассемблер.

В данном микропроцессоре эти коллизии разрешаются не программным способом, а аппаратно. Если команды, предшествующие ХС, изменяют некоторые переменные, а те влияют на выполнение инструкции ХС, то решение о выполнении притормаживается до того момента когда условия будут окончательно сформированы.

Инструкции условного хранения

Некоторые регистры CPU могут условно быть загружены в память данных, используя условные инструкции загрузки, перечисленные в Таблица 19-24. Условия, использованные условными инструкциями загрузки, указаны в Таблица 19-25.

В условной инструкции загрузки, адрес модифицирован и операнд памяти прочитан независимо от условия. Если условие выполнено, соответствующий регистр загружается в память данных. Если условие не выполнено, операнд записывается в ту же позицию памяти, из которой он был прочитан, так что, значение этих позиций памяти остается тем же.

Условные инструкции хранения являются однооперандными инструкциями, но они используют двухпортовую память операндов в режиме косвенной адресации, чтобы разместить инструкцию в одно 16-битовое слово. Следовательно, эти инструкции выполняются в одном цикле.

Условное сохранение счетчика повторения блока (BRC) позволяет Вам сохранить индекс в цикле повторения блока.

Таблица 19-24 – Инструкции условного сохранения (условной записи)

Инструкция	Регистр процессора
SACCD	Аккумулятор А или В
STRCD	Временный регистр
SRCCD	Счетчик повторения блока

Таблица 19-25 – Условия для команд условного хранения

Операнд	Условие	Описание
AEQ	$A = 0$	Аккумулятор А равен нулю
BEQ	$B = 0$	Аккумулятор В равен нулю
ANEQ	$A \neq 0$	Аккумулятор А не равен нулю
BNEQ	$B \neq 0$	Аккумулятор В не равен нулю
ALT	$A < 0$	Аккумулятор А меньше нуля
BLT	$B < 0$	Аккумулятор В меньше нуля
ALEQ	$A \leq 0$	Аккумулятор А меньше или равен нулю
BLEQ	$B \leq 0$	Аккумулятор В меньше или равен нулю
AGT	$A > 0$	Аккумулятор А больше нуля
BGT	$B > 0$	Аккумулятор В больше нуля
AGEQ	$A \geq 0$	Аккумулятор А больше или равен нулю
BGEQ	$B \geq 0$	Аккумулятор В больше или равен нулю

19.5.6.1 Повторение одной инструкции

Микропроцессор включает две инструкции, RPT и RPTZ, которые вызывают повторение следующей инструкции. Количество повторений инструкции получается из операнда инструкции и равняется этому операнду + 1.

Эта величина сохраняется в 16-битовом счетчике повторения (RC). Вы не можете запрограммировать величину в регистре RC, он загружается только инструкциями повторения (RPT или RPTZ). Максимальное количество выполнений определяемых инструкциями – 65 536. Абсолютный адрес программы или данных автоматически увеличивается, когда использовано свойство повторения единственной команды.

Как только инструкция повторения будет декодирована, все прерывания, включая NMI\, но не RS\, запрещены до завершения цикла повторения. Тем не менее, микропроцессор реагирует на сигнал HOLD\ когда выполняется цикл RPT/RPTZ, ответ зависит от величины бита НМ регистра ST1.

Функция повторения может быть использована некоторыми инструкциями, как например, умножение с накоплением и перемещение блока, увеличивая скорость выполнения этих инструкций. Эти многоцикловые инструкции (см. Таблица 19-26) эффективно становятся одноцикловыми инструкциями после первой итерации инструкции повторения.

Таблица 19-26 – Многоцикловые инструкции, которые становятся одноцикловыми при повторении

Команда	Описание	# Циклы *
FIRS	Симметричный фильтр с конечной импульсной характеристикой (FIR)	3
MACD	Умножение и посылка результата в аккумулятор с задержкой	3
MACP	Умножение и посылка результата в аккумулятор	3
MVDK	Пересылка из области данных в область данных	2
MVDM	Пересылка данных в MMR	2
MVDP	Пересылка данных в программную область	4
MVKD	Пересылка из области данных в область данных	2
MVMD	Пересылка из MMR в область данных	2
MVPD	Пересылка из области программ в область данных	3
READA	Пересылка из области программ в область данных	5
WRITE	Пересылка данных в программную область	5
* Количество циклов, когда инструкция не повторяется		

Единственный операнд памяти данных в инструкции не может быть повторен, если использованы режим длинного смещения или абсолютный адрес (например, *ARn(lk), *+ARn(lk), *+ARn(lk)% и *(lk)). Инструкции, указанные в таблице ниже (Таблица 19-27), не могут быть повторены использованием RPT.

Таблица 19-27 – Неповторяемые инструкции

Команда	Описание
ADDM	Сложение длинной константы с ячейкой памяти данных
ANDM	Операция AND ячейки памяти данных с длинной константой
B[D]	Безусловный переход
BACC[D]	Переход по адресу в аккумуляторе
BANZ[D]	Переход, если вспомогательный регистр не равен нулю
BC[D]	Условный переход
CALA[D]	Вызов функции по адресу в аккумуляторе
CALL[D]	Безусловный вызов функции
CC[D]	Условный вызов функции
CMPR	Сравнение со вспомогательным регистром
DST	Сохранение длинного (32 бита) слова
FRETE[D]	Разрешение прерываний и возврат из подпрограммы
IDLE	Инструкция перевода процессора в нерабочий режим
INTR	Ловушка прерывания
LD ARP	Загрузка значения в указатель вспомогательного регистра (ARP)
LD DP	Загрузка указателя страницы памяти (DP)
MVMM	Перемещение одного регистра в другой (MMR - отображенный в памяти)
ORM	Операция OR ячейки памяти данных с длинной константой
RC[D]	Условный возврат
RESET	Программный сброс
RET[D]	Безусловный возврат
RETE[D]	Возврат из обработки прерываний
RETF[D]	Быстрый возврат из прерываний
RND	Округление аккумулятора
RPT	Повторение следующей инструкции
RPTB[D]	Повторение блока инструкций
RPTZ	Повторение следующей инструкции и очистка аккумулятора
RSBX	Сброс бита статусного регистра
SSBX	Установка бита статусного регистра
TRAP	Программная ловушка
XC	Условное выполнение
XORM	Операция XOR ячейки памяти данных с длинной константой

19.5.6.2 Повторение блока инструкций

Повторение блока инструкций используется для того, чтобы повторять блок кода N + 1 раз, где N – некоторая величина, загруженная в регистр-счетчик повторения блоков (BRC). Этот блок кода может содержать одну или более инструкций. В отличие от повторения единственного действия, которое выводит из строя все маскируемые прерывания, действие повторения блока может быть прервано.

Инструкции, использованные для этих действий – RPTB и RPTBD (задержанная инструкция). Инструкция RPTB выполняется в четыре цикла. RPTBD позволяет

выполнение одной двухсловной инструкции или двух однословных инструкций, следующих за инструкцией RPTBD вместо очистки конвейера; таким образом, RPTBD эффективно выполняется в два цикла.

Характеристика повторения блока обеспечивает выполнение цикла более чем нуль раз. Не нулевое выполнение цикла управляется – флагом активности повторения блока (BRAf) в ST1 и следующими регистрами, отображенными в памяти:

- BRC содержит величину N, которая на единицу меньше, чем количество повторений блока.
- Регистр стартового адреса повторения блока (RSA), содержит адрес первой инструкции блока кода, который нужно повторять.
- Регистр конечного адреса повторения блока (REA), содержит адрес последнего слова инструкции блока кода, который нужно повторять.

BRAf устанавливается в 1, чтобы активизировать блочное повторение. Свойство повторения блока может активизироваться, только если количество итераций больше, чем 0. Цикл начинается со следующих шагов:

- **Шаг 1:** Загружается BRC числом циклов в диапазоне от 0 до 65 535.
- **Шаг 2:** Инструкция загружает адрес первой инструкции, которая должна повторяться. Эта инструкция – одна немедленно следующая за RPTB или вторая инструкция, следующая за RPTBD. Инструкция повторения блока (RPTB) или инструкция повторения блока с задержкой (RPTBD) автоматически загружают RSA адресом инструкции, следующей за инструкцией RPTB, или адресом второй инструкции, следующей за инструкцией RPTBD.
- **Шаг 3:** Инструкция загружает REA адресом, следующим за последним словом последней инструкции, которая должна повторяться в блоке, являющаяся также длинным непосредственным операндом, определенным в инструкции. Это действие также устанавливает BRAf. REA загружается 16-битовым непосредственным операндом инструкции RPTB или RPTBD, и бит BRAf устанавливается. Значение для 16-битового непосредственного операнда RPTB или RPTBD равно L-1, где L – адрес инструкции, следующей за последним словом последней инструкции в цикле.

Каждый раз PC обновляется в течение выполнения цикла, REA сравнивается с величиной PC. Если величины равны, BRC - декрементируется. Если BRC больше или равно 0, RSA загружается в PC, чтобы перезапустить цикл. Если нет, BRAf сбрасывается в 0 и процессор продолжает выполнение инструкции следующей за концом цикла.

BRC декрементируется в течение фазы декодирования последней инструкции в блоке повторения. По этой причине, будьте осторожными при использовании инструкции SRCCD в пределах цикла. Для того, чтобы сохранять текущее значение счетчика цикла (предекрементированное BRC), инструкция SRCCD должна быть размещена минимум за три инструкции до конца цикла.

Есть только одна установка регистров повторения блока, так что многочисленный блоки повторения не могут быть вложены, сохраняя при этом контекст вне циклов. Самый простой путь создания вложенных циклов в том, чтобы использовать RPTB[D] инструкцию только для внутреннего цикла и использовать BANZ[D] для всех внешних циклов.

19.6 Функционирование ядра DSP после сброса

Сброс – немаскируемое внешнее прерывание, которое может быть использовано в любое время, чтобы устанавливать микропроцессор в известное состояние. В данной реализации производится путем записи соответствующих бит регистра DSP_CONTROL_STATUS со стороны RISC. Для правильной системной операции после подачи синхросигнала, в течении нескольких тактов бит RST_DSP_MEM должен находиться в исходном состоянии. Пять тактов синхросигнала снятия сброса процессорное ядро DSP выбирает инструкцию по адресу FF80h и начинает выполнять код.

Следующие действия происходят в течение операции сброса:

- IPTR устанавливается в 1FFh;
- PC устанавливается в FF80h;
- Адрес FF80h выдается на шину адреса;
- Шина данных переходит в высокоимпедансное состояние;
- Управляющие сигналы становятся не активными;
- Генерируется сигнал IACK\;
- INTM устанавливается в «1», чтобы запретить все маскируемые прерывания;
- IFR сбрасывается, чтобы сбросить флаги прерывания;
- Счетчик повторения одной команды (RC) сброшен;
- Следующие биты статуса установлены в их начальное состояние:

- ARP = 0	- CLKOFF = 0	- HM = 0	- SXM = 1
- ASM = 0	- CMPT = 0	- INTM = 1	- TC = 1
- AVIS = 0	- CPL = 0	- OVA = 0	- XF = 1
- BRAF = 0	- DP = 0	- OVB = 0	
- C = 1	- DROM = 0	- OVLY = 0	
- C16 = 0	- FRCT = 0	- OVM = 0	

Примечания

1 Остальные биты статуса не инициализируются – Ваш код должен их инициализировать;

2 Сброс не инициализирует указатель стека (SP). Ваш код должен инициализировать его;

3 Если MP/MC = 0, устройство начинает выполнять код из встроенного ROM. В противном случае, начинает выполняться код из внешней памяти.

19.7 Прерывания DSP

Прерывания являются аппаратно и программно задаваемыми сигналами, которые, возникнув в микропроцессоре, приостанавливают основную программу и выполняют другую функцию, называемую программой обработки прерывания (ISR – interrupt service routine). Обычно, прерывания генерируются аппаратурой, которой нужно давать данные или брать из нее данные (например, АЦП, ЦАП, и другие процессоры). Прерывания могут также быть использованы, чтобы сигнализировать, что произошло конкретное событие (например, таймер закончил счет).

Микропроцессор поддерживает как программные, так и аппаратные прерывания:

- Программное прерывание, вызываемое инструкцией (INTR, TRAP, или сброс – RESET).

- Аппаратное прерывание, вызываемое сигналом с физического устройства (регистра DIRQ).
- Когда многочисленные аппаратные прерывания инициируются в одно и то же время, микропроцессор обслуживает их согласно приоритету, в котором прерывание ранга 1 имеет самый высший приоритет.

Каждое прерывание в микропроцессоре может быть маскируемым или немаскируемым:

- Маскируемые прерывания. Эти – аппаратные или программные прерывания, которые могут быть заблокированы (замаскированы) или разрешены (размаскированы) по программе. Микропроцессор поддерживает до 16 маскируемых пользователем прерываний (SINT15 – SINT0). Обычно использует подмножество этих 16 прерываний. Некоторые из них имеют два имени, поскольку они могут быть введены программным обеспечением или аппаратными средствами;
- Немаскируемые прерывания. Эти прерывания не могут быть заблокированы. Микропроцессор всегда признает этот тип прерывания и переходит от основной программы к ISR. Немаскируемые прерывания включают все программные прерывания и два прерывания от RISC: RS (регистр DSP_CONTROL_STATUS) и NMI (регистр AIRQ).

Микропроцессор обрабатывает прерывания в три фазы:

1 Получение требования на прерывание. Требование на останов основной программы требуется через программное обеспечение (программный код) или аппаратные средства. Если источник прерывания просит маскируемое прерывание, соответствующий бит в регистре флага прерывания (IFR) устанавливается при возникновении прерывания.

2 Подтверждение прерывания. Микропроцессор должен подтвердить прием прерывания. Если прерывание является маскируемым, преопределенные условиями должно быть выполнены для этого подтверждения. Для немаскируемых аппаратных прерываний и для программных прерываний, подтверждение безусловное.

3 Выполнение программы обработки прерывания (ISR – interrupt service routine). Как только прерывание подтверждено, микропроцессор выполняет команду перехода на предопределенный адрес (позицию вектора прерывания) и выполняют ISR.

19.7.1 Регистры управления прерываниями ядра

Таблица 19-28 – Регистр флага прерывания (IFR-interrupt flag register)

7	6	5	4	3	2	1	0
XINT2	RINT2	XINT1	RINT1	TINT	INT2	INT1	INT0

15	14	13	12	11	10	9	8
-	-	DMA_INT	CRINT	CDINT	XINT3	RINT3	INT3

Таблица 19-29 – Назначение битов регистра

Биты	Наименование	Назначение
15, 14	-	Зарезервировано
13	DMA_INT	Прерывание DMA (завершение обработки одного из запросов): 0 – не активно; 1 – активно
12	CRINT	Прерывание Криптомодуля: 0 – не активно; 1 – активно
11	CDINT	Прерывание Аудиокодека: 0 – не активно; 1 – активно
10	XINT3	Прерывание передатчика McBSP3: 0 – не активно; 1 – активно
9	RINT3	Прерывание приемника McBSP3: 0 – не активно; 1 – активно
8	INT3	Прерывание RISC3: 0 – не активно; 1 – активно
7	XINT2	Прерывание передатчика McBSP2: 0 – не активно; 1 – активно
6	RINT2	Прерывание приемника McBSP2: 0 – не активно; 1 – активно
5	XINT1	Прерывание передатчика McBSP1: 0 – не активно; 1 – активно
4	RINT1	Прерывание приемника McBSP1: 0 – не активно; 1 – активно
3	TINT	Прерывание Таймера: 0 – не активно; 1 – активно
2	INT2	Прерывание RISC2: 0 – не активно; 1 – активно
1	INT1	Прерывание RISC1: 0 – не активно; 1 – активно
0	INT0	Прерывание RISC0: 0 – не активно; 1 – активно

IFR – регистр, отображенный в памяти процессора, который идентифицирует и определяет активные прерывания. Прерывание устанавливает соответствующий флаг прерывания в IFR, пока оно не будет признано ядром DSP. Любое из следующих четырех событий сбрасывает флаг прерывания в регистре IFR (но не в регистре AIRQ):

- Сброс ядра DSP.
- Ловушка прерывания захвачена.
- Записана «1» в соответствующий бит IFR.
- Инstrukция INTR выполнена, используя соответствующий номер прерывания.

«1» на любом бите IFR указывает на незаконченное прерывание. Для того, чтобы сбросить прерывание, запишите единицу в бит соответствующий биту в IFR. Все незавершенные прерывания могут быть сброшены записью текущего значения IFR снова в IFR.

19.7.2 Регистр Маски Прерывания (IMR-interrupt mask register)

Таблица 19-30 показывает, как микропроцессор использует отображенный в памяти регистр IMR для маскирования внешних и внутренних прерываний. Если INTM = 0 в ST1, «1» на любом бите IMR допускает соответствующее прерывание. Немаскируемое прерывание от RISC не включены в IMR.

Таблица 19-30 – Регистр маски прерывания

7	6	5	4	3	2	1	0
XINT2M	RINT2M	XINT1M	RINT1M	TINTM	INT2M	INT1M	INT0M

15	14	13	12	11	10	9	8
-	-	DMA_INTM	CRINTM	CDINTM	XINT3M	RINT3M	INT3M

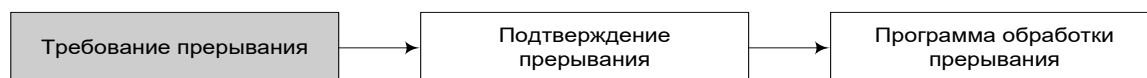
Таблица 19-31 – Назначение битов регистра

Биты	Наименование	Назначение
15, 14	-	Зарезервировано
13	DMA_INTM	Маска прерывания DMA (завершение обработки одного из запросов): 0 – запрещено; 1 – разрешено
12	CRINTM	Маска прерывания Криптомодуля: 0 – запрещено; 1 – разрешено
11	CDINTM	Маска прерывания Аудиокодека: 0 – запрещено; 1 – разрешено

Биты	Наименование	Назначение
10	XINT3M	Маска прерывания передатчика McBSP3: 0 – запрещено; 1 – разрешено
9	RINT3M	Маска прерывания приемника McBSP3: 0 – запрещено; 1 – разрешено
8	INT3M	Маска прерывания RISC3: 0 – запрещено; 1 – разрешено
7	XINT2M	Маска прерывания передатчика McBSP2: 0 – запрещено; 1 – разрешено
6	RINT2M	Маска прерывания приемника McBSP2: 0 – запрещено; 1 – разрешено
5	XINT1M	Маска прерывания передатчика McBSP1: 0 – запрещено; 1 – разрешено
4	RINT1M	Маска прерывания приемника McBSP1: 0 – запрещено; 1 – разрешено
3	TINTM	Маска прерывания Таймера: 0 – запрещено; 1 – разрешено
2	INT2M	Маска прерывания RISC2: 0 – запрещено; 1 – разрешено
1	INT1M	Маска прерывания RISC1: 0 – запрещено; 1 – разрешено
0	INT0M	Маска прерывания RISC0: 0 – запрещено; 1 – разрешено

19.7.3 Обработка прерываний

19.7.3.1 Фаза 1: Прием требования на прерывание



Требование прерывание возникает от аппаратных устройств (в том числе и от ядра RISC) или от команды программы.

Когда возникает требование прерывания, соответствующий флаг (если он имеется), активируется в IFR.

Этот флаг активируется, если даже прерывание позже не подтверждено процессором. Флаг автоматически очищается, когда соответствующее прерывание захвачено.

Требования аппаратных прерываний. Внешние аппаратные прерывания вызываются сигналами с внешних портов прерывания и внутренними аппаратными прерывания с встроенных периферийных устройств. Например, на данном микропроцессоре аппаратные прерывания могут потребоваться от следующих источников:

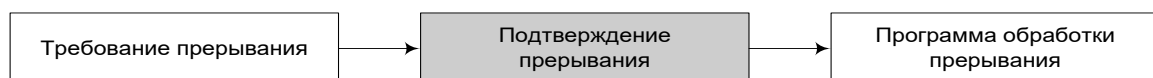
- Прерывания от RISC-ядра (регистр AIRQ) INT0...INT3, NMI;
- Сброс ядра DSP (регистр DSP_CONTROL_STATUS подсистемы RISC);
- Прерывания от последовательных портов (RINT0 и XINT0 , RINT1 и XINT1, RINT2 и XINT2, RINT3 и XINT3);
- Прерывание от таймера (TINT);
- Прерывание от криптомодуля (CRINT);
- Прерывание от аудиокодека (CDINT).

Требования программного прерывания. Программное прерывание вызывается одной из следующих инструкций программы:

- INTR. Эта инструкция позволяет Вам выполнить любую программу обработки прерывания.
- Операнд инструкции (K) указывает значение вектора прерывания CPU. Когда прерывание INTR подтверждено, бит режима прерывания (INTM) в ST1 устанавливается в «1», чтобы запретить маскируемые прерывания.
- TRAP. Эта инструкция выполняется подобно инструкции INTR, но без установки бита INTM.
- RESET. Эта инструкция выполняет немаскируемый программный сброс, что может быть использовано всякий раз для установки микропроцессора в известное состояние. Инструкция RESET влияет на ST0 и ST1, но не влияет на PMST. Для общего описания воздействия на биты и регистры, смотри описание инструкции RESET.

Когда инструкция RESET подтверждена, INTM устанавливается в 1, чтобы запретить маскируемые прерывания. Инициализация IPTR и периферийных регистров отличается от инициализации сделанной аппаратным сбросом.

19.7.3.2 Фаза 2: Подтверждение прерывания



После возникновения требования прерывания от аппаратных или программных средств, CPU должно предложить ответ. Выполнение программ прерывается для немаскируемых аппаратных прерываний немедленно. Маскируемые аппаратные прерывания признаются только после того, как определенные условия будут выполнены:

- *Приоритет самый высокий.* Когда более, чем одно аппаратное прерывание требуется в одно и то же время, микропроцессор обслуживает их согласно приоритету, причем приоритет в наборе ранга 1 указывает самый высокий приоритет.
- *INTM бит в 0.* Бит способа прерывания (INTM), который в ST1, разрешает или запрещает все маскируемые прерывания:
 - Когда INTM = 0, все незамаскированные прерывания разрешены.
 - Когда INTM = 1, все незамаскированные прерывания запрещены.

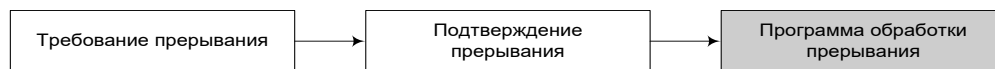
INTM устанавливается в «1» автоматически, когда прерывание захвачено. Если программа обработки прерывания (ISR) заканчивается, используя инструкцию RETE (возврат из прерывания с автоматическим разрешением), INTM восстанавливается (очистка бита).

INTM также может быть установлен сбросом ядра DSP или выполнением инструкции SSBX INTM (запрет прерывания). INTM сбрасывается при выполнении инструкции RSBX INTM (разрешить прерывание). INTM не модифицирует IMR или IFR.

– *Бит маски IMR в 1.* Каждое маскируемое прерывание имеет собственный бит маски в IMR. Для того, чтобы разрешить прерывание, установите соответствующий бит маски в 1.

CPU отвечает на маскируемое аппаратное прерывание и выдает на шину инструкций инструкцию INTR. Эта инструкция переводит PC в соответствующий адрес и выбирает программный вектор.

19.7.3.3 Фаза 3: Выполнение программы обработки прерывания (ISR-interrupt service routine)



После того, как прерывание подтверждено, центральный процессор выполняет следующие действия:

- 1 Загружает значение программного счетчика (PC) (адрес возврата) на вершину стека памяти данных.
- 2 Загружается PC адресом вектора прерывания.
- 3 Выбирает инструкцию, расположенную по адресу вектора. (Если переход задержанный, и Вы также загрузили одну двухсловную инструкцию или две однословные инструкции, CPU также выбирает эти слова).
- 4 Выполняется переход, который лидирует по адресу вашего ISR. (Если переход задержанный, дополнительная инструкция (инструкции) выполняются перед переходом).
- 5 Выполняется ISR, пока инструкция возврата не подведет итог ISR .
- 6 В соответствии с указателем верхушки стека (SP) выталкивает адресу возврата в PC .
- 7 Продолжает выполнять основную программу.

Для того, чтобы определять, что какой векторный адрес назначен каждому из прерываний, смотрите на таблицу векторов. Адреса прерываний расположены отдельно, разделенные шагом в четыре адреса, так что в этих позициях может быть размещена задержанная инструкция перехода и две 1-словные инструкции или одна 2-словная инструкция.

19.7.3.4 Сохранение контекста при прерываниях

Когда выполняется программа обработки прерывания, определенные регистры должно быть сохранены в стеке. Когда программа возвращается из ISR (RC[D], RETE[D], или RETF[D]), ваш программный код должен восстановить содержание этих регистров. Вы можете управлять хранением в стеке до тех пор, пока стек не превысит пространство памяти. Этот стек также используется для подпрограммных вызовов; Микропроцессор

поддерживает подпрограммные вызовы в пределах ISR. Поскольку регистры DSP и периферийные регистры отображены в памяти, команды PSHM и POPM могут передать эти регистры из стека и в стек. Кроме того, инструкции PSHD и POPD могут передать значения из памяти данных в стек и наоборот.

Есть ряд правил, которыми необходимо руководствоваться при записи контекста и восстановлении его:

1 Восстановление данных из стека должно происходить в точном обратном порядке его сохранению.

2 BRC должен быть восстановлен до восстановления бита BRAF в ST1. Если не придерживаться этого правила, бит BRAF будет очищен, если $BRC = 0$ прежде, чем BRC будет восстановлен.

19.7.3.5 *Время перехода к обработке прерывания*

Микропроцессор завершает все инструкции в конвейере, кроме предварительно выбранных инструкций, так что максимальное время ожидания обработки прерывания зависит от содержания конвейера. Смотри описание обработки прерываний в работе конвейера в главе 7. Инструкции, время исполнения которых расширено состояниями ожидания доступа для медленной памяти, и повторяющиеся инструкции требуют дополнительное время для обработки прерываний.

Инструкция повторения одной инструкций (RPT и RPTZ) требует, чтобы выполнение следующей инструкции было полностью завершено перед разрешением прерывания, чтобы не нарушить контекст повторяющейся инструкции. Эта защита необходима, поскольку эти инструкции запускают параллельные действия в конвейер, и контекст этих действий не может быть сохранен в ISR.

Прерывания не могут быть обработаны между инструкцией RSBX INTM (RSBX – сброс бита статусного регистра, INTM (Interrupt mode – общее разрешение прерываний) и следующей инструкцией в программной последовательности. Если прерывание происходит в течение фазы декодирования RSBX INTM, CPU всегда завершает RSBX INTM, а также следующую инструкцию прежде, чем незаконченное прерывание будет обработано. При ожидании завершения этих инструкций проверяется, что возврат (RET) может быть выполнен в ISR прежде, чем следующее прерывание будет обработано, чтобы защититься от переполнения стека.

Если команда ISR заканчивается инструкцией RETE (возврат из ISR с разрешением прерываний), инструкция RSBX INTM необязательная. Подобно сказанному инструкции RSBX INTM, инструкция SSBX INTM и инструкция, следующая за ней, не могут быть прерваны.

Примечание – Сброс, не задерживается в случае многоцикловых инструкций.

19.7.3.6 *Таблица векторов прерываний*

Векторы прерывания могут быть перераспределены в начало любой 128-словной страницы в программной памяти за исключением резервных областей. Адрес вектора прерывания генерируется конкатенацией поля указателя прерываний (IPTR) регистра PMST (Processor Mode Status Register) с номером вектора (0 – 31) сдвинутым влево на две позиции. Смотрите в качестве примера Рисунок 19-48: если возник INT0 и $IPTR = 0001h$, вектором прерывания выбирается 00C0h. Вектор прерывания для INT0 – 16 или 10h.

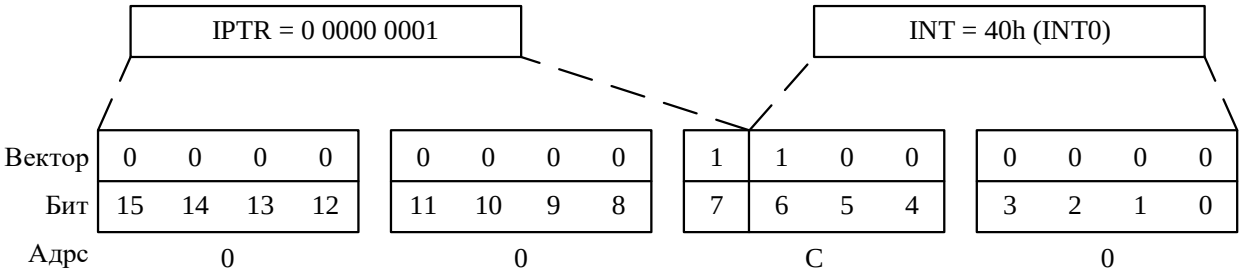


Рисунок 19-48 – Генерация адреса вестора прерываний

При сбросе, биты IPTR устанавливаются в 1 (IPTR = 1FFh); эта величина отображает векторы на страницу 511 в пространстве памяти программ. Следовательно, вектор аппаратного сброса всегда расположен на позиции 0FF80h., векторы прерывания могут быть отображены в другое место загрузкой IPTR другой, нежели 1FFh, величиной. Например, векторы прерывания могут начинаться с адреса 0080h загрузкой IPTR величиной 0001h.

Примечание – Вектор аппаратного сброса (RS) не может быть перемещен, поскольку аппаратный сброс загружает IPTR единицами. Следовательно, вектор аппаратного сброса всегда расположен по адресу FF80h в программном пространстве.

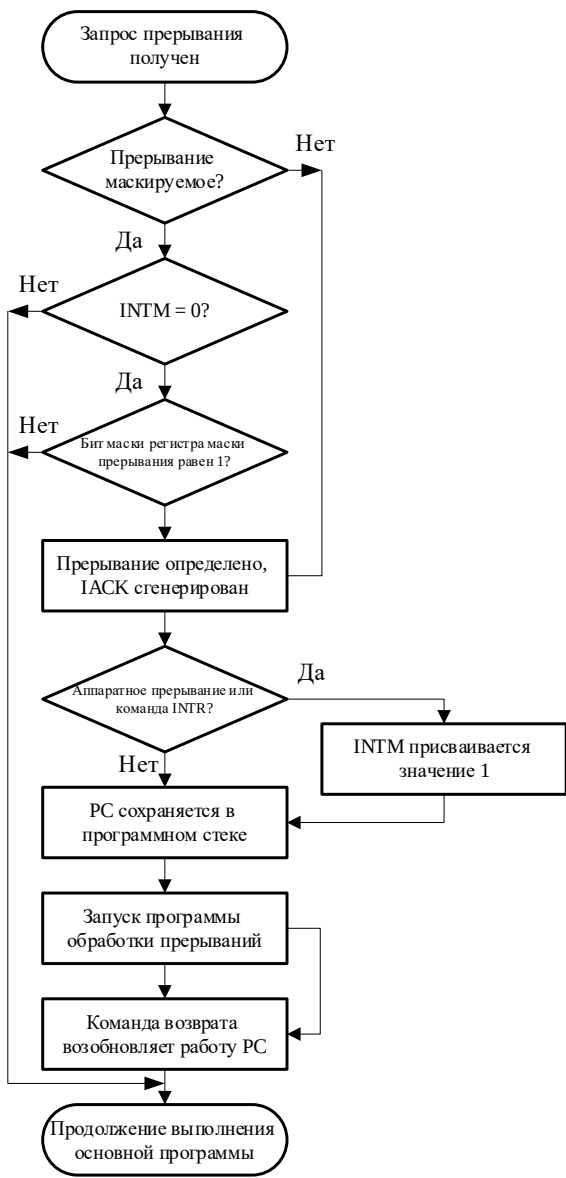


Рисунок 19-49 – Структурная схема процесса прерывания

Таблица 19-32 показывает номер прерывания, приоритет и размещение для микропроцессора.

Таблица 19-32 – Расположение и приоритеты прерываний ‘546

TRAP/INTR номер (К)	Приоритет	Название	Расположение (hex)	Функция
0	1	$\overline{RS}$, SINTR	0	Сброс (Аппаратный И Программный Сброс)
1	2	$\overline{NMI}$, SINT16	4	Немаскируемое Прерывание От RISC (Регистр AIRQ)
2	-	SINT17	8	Программное Прерывание #17
3	-	SINT18	C	Программное Прерывание #18
4	-	SINT19	10	Программное Прерывание #19
5	-	SINT20	14	Программное Прерывание #20
6	-	SINT21	18	Программное Прерывание #21
7	-	SINT22	1C	Программное Прерывание #22
8	-	SINT23	20	Программное Прерывание #23
9	-	SINT24	24	Программное Прерывание #24
10	-	SINT25	28	Программное Прерывание #25
11	-	SINT26	2C	Программное Прерывание #26
12	-	SINT27	30	Программное Прерывание #27
13	-	SINT28	34	Программное Прерывание #28
14	-	SINT29	38	Программное Прерывание #29
15	-	SINT30	3C	Программное Прерывание #30
16	3	$\overline{INT0}$, SINT0	40	Прерывание От RISC 0 (Регистр AIRQ)
17	4	$\overline{INT1}$, SINT1	44	Прерывание От RISC 1 (Регистр AIRQ)
18	5	$\overline{INT2}$, SINT2	48	Прерывание От RISC 2 (Регистр AIRQ)
19	6	TINT, SINT3	4C	Прерывание Таймера
20	7	RINT1, SINT4	50	Прерывание Приема BSP 1
21	8	XINT1, SINT5	54	Прерывание Передачи BSP1
22	9	RINT2, SINT6	58	Прерывание Приема BSP2
23	10	XINT2, SINT7	5C	Прерывание Передачи BSP2
24	11	$\overline{INT3}$, SINT8	60	Прерывание От RISC 3 (Регистр AIRQ)
25	12	RINT3,	64	Прерывание Приема BSP 3
26	13	XINT3,	68	Прерывание Передачи BSP 3
27	14	CDINT	6C	Прерывание Аудиокодека
28	15	CRINT	70	Прерывание Криптомодуля
29	16	DMAINT	74	Прерывание DMA (DSP)
30-31	-		78-7C	Зарезервировано

19.8 Регистры управления и состояния ядра

Микропроцессор имеет три регистра статуса и управления:

- Регистр состояния 0 (ST0);
- Регистр состояния 1 (ST1);
- Регистр состояния режима процессора (PMST).

ST0 и ST1 содержит состояние различных условий и режимов; PMST содержит состояние установок памяти и управляющую информацию. Поскольку эти регистры отображены в память, они могут быть загружены и выгружены из памяти данных; статус процессора может быть сохранен и восстановлен подпрограммами и программами обработки прерываний (ISRs).

19.8.1 Регистры Статуса (ST0 и ST1)

Индивидуальные биты регистров ST0 и ST1 могут быть установлены или сброшены инструкциями SSBX и RSBX. Например, режим расширения знака устанавливается инструкциями SSBX 1, SXM, а сбрасывается RSBX 1, SXM. Битовые поля ARP, DP и ASM могут быть загружены с использованием инструкции LD с коротким непосредственно заданным операндом. Поля ASM и DP могут также быть загружены как значения в памяти, используя инструкцию LD.

Рисунок 19-50 и Таблица 19-33 описывают биты ST0. Рисунок 19-51 и Таблица 19-34 описывают биты ST1.

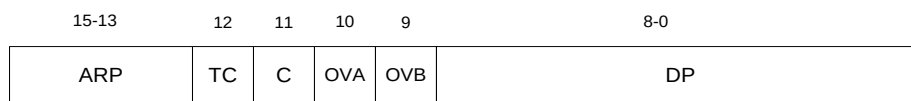


Рисунок 19-50 – Регистр состояния 0 (ST0)

Таблица 19-33 – Назначение битов регистра состояния 0 (ST0)

Биты	Обозначение	Значение по сбросу	Функция
15-13	ARP	0	Указатель вспомогательного регистра. Это трехбитовое поле выбирает вспомогательный регистр, используемый в совместимом режиме косвенной адресации одного операнда. ARP должен быть всегда установлен в нуль, когда DSP в стандартной режиме (CMPT=0)
12	TC	1	Флаг тестирования/управления. TC хранит результаты тестирования арифметико-логическим устройством (ALU). TC формируется командами BIT, BITF, BITT, CMPM, CMPR, CMPS, и SFTC. Состояние (установленное или сброшенное) TC определяется тогда, когда выполняется команда условного перехода, вызов процедуры, выполнения и возврата из подпрограммы. TC = 1, если следующие условия являются истиной: Бит, протестированный командой BIT или BITT равен 1. Условие сравнения, протестированное командами CMPM, CMPR, или CMPS верно (CMPM-равенство

Биты	Обозначение	Значение по сбросу	Функция
			между значением в памяти данных и непосредственным операндом, CMPR-сравнение AR0 и другого вспомогательного регистра, CMPS - сравнение старшего и младшего слова аккумулятора). Бит 31 и бит 30 аккумулятора, протестированных командой SFTC, имеют значения различающиеся друг от друга.
11	C	1	Перенос установлен в 1, если результат сложения генерирует перенос; сброшен в 0, если результат вычитания генерирует заем. В противном случае, сбрасывается после сложения и устанавливается после вычитания, за исключением ADD или SUB с 16- битовым сдвигом. Только в этих случаях, командой ADD может быть установлен и в командой SUB сброшен бит переноса, но на него ничто не может повлиять иное. Перенос и заем определены в 32-й битовой позиции и формируются только на уровне ALU. Инструкции сдвига и циклического сдвига (ROR, ROL, SFTA, и SFTL), и MIN, MAX, ABS, и NEG инструкции также влияют на этот бит.
10	OVA	0	Флаг переполнения для аккумулятора А. OVA устанавливается в 1, когда происходит переполнение в ALU или сумматоре умножителя и расположение для результата - аккумулятор А. Как только переполнение произойдет, OVA остается установленным до сброса, и инструкций BC[D], CC[D], RC[D], или XC, выполненных с использованием условия AOV и условия ANOV. Инструкция RSBX может также очистить этот бит.
9	OVB	0	Флаг переполнения для аккумулятора В. OVB устанавливается в 1, когда происходит переполнение в ALU или сумматоре умножителя и расположение для результата – аккумулятор В. Как только переполнение произойдет, OVB остается установленным до сброса, и инструкций BC[D], CC[D], RC[D], или XC выполненных с использованием условия BOV и условия BNOV. Инструкция RSBX может также очистить этот бит.
8-0	DP	0	Указатель страницы памяти данных. Эта 9-битовая область конкатенируется с семью младшими словами инструкции, чтобы сформировать прямой адрес памяти в 16 бит для адресации единственного операнда памяти. Эта операция выполняется, если бит режима компиляции в ST1 (CPL) = 0. Поле DP может быть загружено инструкцией LD с коротким непосредственно заданным операндом или из памяти данных.

15	14	13	12	11	10	9	8	7	6	5	4-0
BRAF	CPL	XF	HM	INTM	O	OVM	SXM	C16	FRCT	CMPT	ASM

Рисунок 19-51 – Регистр состояния 1 (ST1)

Таблица 19-34 – Назначение битов регистра состояния 1 (ST1)

Биты	Обозначение	Значение по сбросу	Функция
15	BRAF	0	Флаг активности повторения блока. BRAF показывает активность в настоящее время повторения блока. BRAF = 0 – блочное повторение деактивировано. BRAF сбрасывается, когда счетчик повторения блока (BRC) декрементирован ниже 0. BRAF = 1 – повторение блока активно. BRAF автоматически устанавливается, когда выполнена инструкция RPTB
14	CPL	0	Режим компилятора. CPL показывает, какой указатель использован в относительной прямой адресации: CPL = 0 – режим относительной прямой адресации, использовавший выбор страничного указателя данных (DP). CPL = 1 – режим относительной прямой адресации, использовавший выбор указателя стека (SP)
13	XF	1	Состояние XF. XF показывает статус штырька внешнего флага (XF), который является выходным штырьком общего назначения. Инструкция SSBX может установить XF и инструкция RSBX может сбросить XF
12	HM	0	Способ захвата. HM показывает, какой из двух источников программы продолжает выполняться внутри процессором, опрашивая активность сигнала HOLD: HM = 0 – процессор продолжает выполнение из внутренней программной памяти, при этом устанавливая свой внешний интерфейс в высокоимпедансное состояние. HM = 1 – процессор останавливает внутреннее выполнение
11	INTM	1	Режим прерывания. INTM маскирует или разрешает глобально все прерывания. INTM = 0 – все незамаскированные прерывания разрешены. INTM = 1 – все маскируемые прерывания запрещены. Инструкция SSBX устанавливает INTM и инструкция RSBX сбрасывает INTM. INTM устанавливается в 1

Биты	Обозначение	Значение по сбросу	Функция
			сбросом или когда принята маскируемая ловушка прерывания (INTR или внешние прерывания). INTM сбрасывается в 0, когда выполняется инструкция RETE или RETF (возврат из прерывания). INTM не влияет на немаскируемые прерывания (RS\ и NMI\). INTM не может быть установлен действиями записи в память
10	0	0	Всегда читается как нуль
9	OVM	0	Режим переполнения. OVM определяет, что загружается на аккумулятор назначения, когда происходит переполнение: OVM = 0 – переполнение результата из ALU или сумматора умножителя обычно переполняется и на аккумуляторе назначения. OVM = 1 – аккумуляторе назначения установлен на или наиболее положительную величину (00 7FFF FFFFh) или наиболее отрицательную величину (FF 8000 0000h) при столкновении с переполнением. Инструкции SSBX и RSBX устанавливают и сбрасывают OVM, соответственно
8	SXM	1	Режим расширения знака. SXM определяет, как выполняется расширение знака: SXM = 0 – расширение знака подавлено. SXM = 1 – данные расширяются знаком прежде, чем будут использованы в ALU. SXM не влияет на определения некоторых инструкций: инструкции ADDS, LDU и SUB подавляют расширение знака независимо от величины SXM. Инструкции SSBX и RSBX устанавливают и сбрасывают SXM, соответственно
7	C16	0	Режим двойной 16-битовой или спаренной точности арифметических операций. C16 определяет режим арифметических действий в ALU: C16 = 0 – ALU работает в режиме двойной точности арифметических действий. C16 = 1 – ALU работает в режиме спаренных арифметических действий
6	FRCT	0	Режим дробности. Когда FRCT в 1, выход множителя сдвигается влево на один бит, чтобы компенсировать дополнительный бит знака
5	CMPT	0	Режим совместимости. CMPT определяет способ совместимости для ARP: CMPT = 0 – ARP не обновляется в режиме косвенной адресации с единственным операндом в памяти

Биты	Обозначение	Значение по сбросу	Функция
			данных. ARP должен быть всегда установлен в 0, когда DSP – в этом способе. CMPT = 1 – ARP обновляется в режиме косвенной адресации с единственным операндом в памяти данных, кроме тех случаев, когда инструкция выбирает вспомогательный регистр 0 (AR0)
4-0	ASM	0	Режим сдвига аккумулятора. пятибитовая поле ASM определяет величину перемещения в пределах от -16 до 15 и закодирована как величина в дополнительном коде. Инструкции с параллельным хранением, а также STH, STL, ADD, SUB и LD, используют эту возможность сдвига. ASM может быть загружен из памяти данных или же инструкцией LD, использовавшей короткий непосредственный операнд

19.8.1.1 Регистр состояния режима процессора (PMST)

Регистр PMST загружается инструкциями работающими с регистрами, отображенными в памяти, как например, STM. Рисунок 19-52 и Таблица 19-35 описывают биты PMST.

15-7	6	4	3	1	0		
IPTR	MP/ MC\	OVLY	AVIS	DROM	CLKOFF	SMUL*	SST*

* В данной модели микропроцессора эти биты не поддерживаются.

Рисунок 19-52 – Регистр состояния режима процессора (PMST)

Таблица 19-35 – Назначение битов регистра PMST

Биты	Обозначение	Значение по сбросу	Функция
15-7	IPTR	1FFh	Указатель вектора прерывания. 9-битовое поле IPTR указывает на 128-словную программную страницу, где находятся векторы прерывания. Вы можете перенаправить векторы прерывания при загрузке. При сбросе, эти биты все устанавливаются в 1; при сбросе вектора всегда расположены по адресу FF80h в пространстве программной памяти. Инструкция RESET не влияет на эту поле
6	MP/MC\	MP/MC\ контакт	Режим микропроцессор/микрокомпьютер. MP/MC\ разрешает/запрещает встроенное ПЗУ, при адресации в пространстве программной памяти. MP/MC = 0 – встроенное ПЗУ разрешен и адресуем. MP/MC = 1 – встроенное ПЗУ не доступно. MP/MC устанавливается в значение, соответствующее логическому уровню на контакте MP/MC\, когда он семплируется при сбросе. Этот штырек - не семплируется

Биты	Обозначение	Значение по сбросу	Функция
			снова до следующего сброса. Инструкция RESET не влияет на этот бит. Этот бит может быть также установлен или сброшен по программе
5	OVLY	0	Оверлей ОЗУ. OVLY разрешает отобразить встроенное ОЗУ в программное пространство. Величины для OVLY бита: OVLY = 0 – встроенное ОЗУ адресуется в пространстве данных, но не в программном пространстве. OVLY = 1 – встроенное ОЗУ отображено в программном пространстве и пространстве данных. Страница данных 0 (адреса от 0h до 7Fh), тем не менее, не отображаются в программное пространство
4	AVIS	0	Режим видимости адреса. AVIS разрешает/запрещает видимость внутреннего программного адреса на контактах адреса. AVIS = 0 – внешние линии адреса не изменяются с внутренним программным адресом. Управляющие сигналы и сигналы данных не воздействуют на шины адреса, передавая последний адрес в шину. AVIS = 1 – этот режим позволяет внутреннему программному адресу появляться на контактах микропроцессора, так чтобы внутренний программный адрес мог быть прослежен. Также, это позволяет вектору прерывания быть декодированным вместе с IACK\, когда векторы прерывания находятся во встроенной памяти
3	DROM	0	ПЗУ данных. DROM разрешает отображать встроенное ПЗУ в пространство данных. Значения для бита DROM: DROM = 0 – встроенное ПЗУ не отображено в пространство данных. DROM = 1 – часть встроенного ПЗУ отображена в пространство данных. Смотри главу 3, Память, относительно деталей.
2	CLKOFF	0	CLOCKOUT OFF. Когда бит CLKOFF в 1, выход CLKOUT запрещен и остается в высокоуровневом состоянии
1	SMUL*	N/A	Насыщение в умножении. Когда SMUL = 1, насыщение результата умножения происходит перед выполнением накопления в MAC или MAS инструкции. Бит SMUL применяется только тогда, когда OVM = 1 и FRCT = 1. БИТ SMUL позволяет операциям MAC и MAS соответствовать базовым операциям MAC и MAS определенным в спецификации ETSI GSM (спецификация GSM 6.06,6.10, 6.53). Эффект – в том, что результат 8000h * 8000h насыщается к 7FF FFFFh в случае

Биты	Обозначение	Значение по сбросу	Функция
			дробных чисел прежде, чем последующее сложение/вычитание потребуется в MAC или MAS инструкциях. В этом режиме инструкция MAC является эквивалентом MPY + ADD, когда OVM=1. Если способ не установлен и OVM = 1, результат умножения не насыщен перед выполнением дополнения/ вычитания, только результаты MAC и инструкции MAS насыщены. См. пример 4-1 насыщения в действиях умножения
0	SST*	N/A	Насыщение при хранении. Когда SST в 1, насыщение данных с аккумулятора разрешено перед хранением в памяти. Насыщение выполняется после операции сдвига. Насыщение в загрузке происходит со следующими инструкциями: STH, STL, STLM, DST, ST ADD, ST LD, ST MACR[R], ST MAS[R], ST MPY, и ST SUB. Следующие шаги выполняются, когда используется насыщение при сохранении: 1 40-Битовая величина данных сдвигается (право или влево) в зависимости от инструкции. Сдвиг такой же, как и описанный в инструкции SFTA и зависит от бита SXM. 2 40-Битовая величина данных насыщается к 32-битовому значению; насыщение зависит от бита SXM (предполагается, что число всегда положительное). Если SXM = 0, генерируется следующая 32-битовая величина: – 7FFF FFFFh, если величина больше, чем 7FFF FFFFh. Если SXM = 1, генерируется следующая 32-битовая величина: – 7FFF FFFFh, если величина больше, чем 7FFF FFFFh ; – 8000 0000h, если величина – меньше чем 8000 0000h . 3 Данные загружены в память в зависимости от инструкции. 4 Содержание аккумулятора остается неизменным в течение операции. См. пример 4-2 насыщения на операциях загрузки (хранения)

20 Система команд DSP

20.1 Обзор набора команд

Команды ядра DSP можно распределить по четырем типам операций:

- 1 Арифметические операции;
- 2 Логические операции;
- 3 Операции управления командой;
- 4 Операции загрузки с запоминанием.

В этом разделе каждый из типов операций подразделяется на небольшие группы команд с одинаковыми функциями. Вместе с каждым списком команд вы найдете наиболее подходящее число слов и циклов, а также класс команды. Здесь же размещена информация о повторяющихся и неповторяющихся командах.

20.1.1 Арифметические операции

Этот раздел включает команды арифметических операций. С таблиц ниже приведены команды следующих функциональных групп:

- 1 Команды сложения (Таблица 20-1);
- 2 Команды вычитания (Таблица 20-2);
- 3 Команды умножения (Таблица 20-3);
- 4 Команды умножения со сложением (Таблица 20-4);
- 5 Команды умножения с вычитанием (Таблица 20-4);
- 6 Команды с двойным операндом (32 бита) (Таблица 20-5);
- 7 Проблемно-ориентированные команды (Таблица 20-6).

Таблица 20-1 – Команды сложения

Синтаксис	Выражение (операция)	W *	C *	Класс
ADD Smem, src	src = src + Smem	1	1	3A, 3B
ADD Smem, TS, src	src = src + Smem << TS	1	1	3A, 3B
ADD Smem, 16, src [, dst]	dst = src + Smem << 16	1	1	3A, 3B
ADD Smem [, SHIFT], src [, dst]	dst = src + Smem << SHIFT	2	2	4A, 4B
ADD Xmem, SHFT, src	src = src + Xmem << SHFT	1	1	3A
ADD Xmem, Ymem, dst	dst = Xmem << 16 + Ymem << 16	1	1	7
ADD #lk [, SHFT], src [, dst]	dst = src + #lk << SHFT	2	2	2
ADD #lk, 16, src [, dst]	dst = src + #lk << 16	2	2	2
ADD src [, SHIFT] [, dst]	dst = dst + src << SHIFT	1	1	1
ADD src, ASM [, dst]	dst = dst + src << ASM	1	1	1
ADDC Smem, src	src = src + Smem + C	1	1	3A, 3B
ADDM #lk, Smem	Smem = Smem + #lk	2	2	18A, 18B
ADDS Smem, src	src = src + uns(Smem)	1	1	3A, 3B
* Количество слов (W) и циклов (Ц) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных. Добавляется одно слово и один цикл при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem				

Таблица 20-2 – Команды вычитания

Синтаксис	Выражение (операция)	W *	C *	Класс
SUB Smem, src	src = src – Smem	1	1	3А, 3В
SUB Smem, TS, src	src = src – Smem << TS	1	1	3А, 3В
SUB Smem, 16, src [, dst]	dst = src – Smem << 16	1	1	3А, 3В
SUB Smem [, SHIFT], src [, dst]	dst = src – Smem << SHIFT	2	2	4А, 4В
SUB Xmem, SHFT, src	src = src – Xmem << SHFT	1	1	3А
SUB Xmem, Ymem, dst	dst = Xmem << 16 – Ymem << 16	1	1	7
SUB #lk [, SHFT], src [, dst]	dst = src – #lk << SHFT	2	2	2
SUB #lk, 16, src [, dst]	dst = src – #lk <<16	2	2	2
SUB src[, SHIFT] [, dst]	dst = dst – src << SHIFT	1	1	1
SUB src, ASM [, dst]	dst = dst – src << ASM	1	1	1
SUBB Smem, src	src = src – Smem – C	1	1	3А, 3В
SUBC Smem, src	If (src – Smem << 15) _ 0 src = (src – Smem << 15) << 1 + 1 Else src = src << 1	1	1	3А, 3В
SUBS Smem, src	src = src – uns(Smem)	1	1	3А, 3В
* Количество слов (W) и циклов (C) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных. Добавляется одно слово и один цикл при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem				

Таблица 20-3 – Команды умножения

Синтаксис	Выражение (операция)	W *	C *	Класс
MPY Smem, dst	dst = T * Smem	1	1	3А, 3В
MPYR Smem, dst	dst = rnd(T * Smem)	1	1	3А, 3В
MPY Xmem, Ymem, dst	dst = Xmem * Ymem, T = Xmem	1	1	7
MPY Smem, #lk, dst	dst = Smem * #lk , T = Smem	2	2	6А, 6В
MPY #lk, dst	dst = T * #lk	2	2	2
MPYA dst	dst = T * A(32–16)	1	1	1
MPYA Smem	B = Smem * A(32–16), T = Smem	1	1	3А, 3В
MPYU Smem, dst	dst = uns(T) * uns(Smem)	1	1	3А, 3В
SQUR Smem, dst	dst = Smem * Smem, T = Smem	1	1	3А, 3В
SQUR A, dst	dst = A(32–16) * A(32–16)	1	1	1
* Количество слов (W) и циклов (C) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных. Добавляется одно слово и один цикл при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem				

Таблица 20-4 – Команды умножения со сложением и умножения с вычитанием

Синтаксис	Выражение (операция)	W *	C *	Класс
MAC Smem, src	$src = src + T * Smem$	1	1	3A, 3B
MAC Xmem, Ymem, src [, dst]	$dst = src + Xmem * Ymem,$ $T = Xmem$	1	1	7
MAC #lk, src [, dst]	$dst = src + T * \#lk$	2	2	2
MAC Smem, #lk, src [, dst]	$dst = src + Smem * \#lk,$ $T = Smem$	2	2	6A, 6B
MACR Smem, src	$src = rnd(src + T * Smem)$	1	1	3A, 3B
MACR Xmem, Ymem, src [, dst]	$dst = rnd(src + Xmem * Ymem),$ $T = Xmem$	1	1	7
MACA Smem [, B]	$B = B + Smem * A(32-16),$ $T = Smem$	1	1	3A, 3B
MACA T, src [, dst]	$dst = src + T * A(32-16)$	1	1	1
MACAR Smem [, B]	$B = rnd(B + Smem * A(32-16)),$ $T = Smem$	1	1	3A, 3B
MACAR T, src [, dst]	$dst = rnd(src + T * A(32-16))$	1	1	1
MACD Smem, pmad, src	$src = src + Smem * pmad,$ $T = Smem, (Smem + 1) = Smem$	2	3	23A, 23B
MACP Smem, pmad, src	$src = src + Smem * pmad,$ $T = Smem$	2	3	22A, 22B
MACSU Xmem, Ymem, src	$src = src + uns(Xmem) * Ymem,$ $T = Xmem$	1	1	7
MAS Smem, src	$src = src - T * Smem$	1	1	3A, 3B
MASR Smem, src	$src = rnd(src - T * Smem)$	1	1	3A, 3B
MAS Xmem, Ymem, src [, dst]	$dst = src - Xmem * Ymem,$ $T = Xmem$	1	1	7
MASR Xmem, Ymem, src [, dst]	$dst = rnd(src - Xmem * Ymem),$ $T = Xmem$	1	1	7
MASA Smem [, B]	$B = B - Smem * A(32-16),$ $T = Smem$	1	1	3A, 3B
MASA T, src [, dst]	$dst = src - T * A(32-16)$	1	1	1
MASAR T, src [, dst]	$dst = rnd(src - T * A(32-16))$	1	1	1
SQURA Smem, src	$src = src + Smem * Smem,$ $T = Smem$	1	1	3A, 3B
SQURS Smem, src	$src = src - Smem * Smem,$ $T = Smem$	1	1	3A, 3B
* Количество слов (W) и циклов (C) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных. Добавляется одно слово и один цикл при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem				

Таблица 20-5 – Двухоперандные команды (32 бита)

Синтаксис	Выражение (операция)	W *	C *	Класс
DADD Lmem, src [, dst]	If C16 = 0 dst = Lmem + src If C16 = 1 dst(39–16) = Lmem(31–16) + src(31–16) dst(15–0) = Lmem(15–0) + src(15–0)	1	1	9A, 9B
DADST Lmem, dst	If C16 = 0 dst = Lmem + (T << 16 + T) If C16 = 1 dst(39–16) = Lmem(31–16) + T dst(15–0) = Lmem(15–0) – T	1	1	9A, 9B
DRSUB Lmem, src	If C16 = 0 src = Lmem – src If C16 = 1 src(39–16) = Lmem(31–16) – src(31–16) src(15–0) = Lmem(15–0) – src(15–0)	1	1	9A, 9B
DSADT Lmem, dst	If C16 = 0 dst = Lmem – (T << 16 + T) If C16 = 1 dst(39–16) = Lmem(31–16) – T dst(15–0) = Lmem(15–0) + T	1	1	9A, 9B
DSUB Lmem, src	If C16 = 0 src = src – Lmem If C16 = 1 src (39–16) = src(31–16) – Lmem(31–16) src (15–0) = src(15–0) – Lmem(15–0)	1	1	9A, 9B
DSUBT Lmem, dst	If C16 = 0 dst = Lmem – (T << 16 + T) If C16 = 1 dst(39–16) = Lmem(31–16) – T dst(15–0) = Lmem(15–0) – T	1	1	9A, 9B
* Количество слов (W) и циклов (C) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных. Добавляется одно слово и один цикл t при использовании длинного смещения в косвенной адресации или абсолютной адресации Lmem				

Таблица 20-6 – Проблемно-ориентированные команды

Синтаксис	Выражение (операция)	W *	C *	Класс
ABDST Xmem, Ymem	B = B + A(32–16) A = (Xmem – Ymem) << 16	1	1	7
ABS src [, dst]	dst = src	1	1	1
CMPL src [, dst]	dst = ~src	1	1	1
DELAY Smem	(Smem + 1) = Smem	1	1	24A, 24B

Синтаксис	Выражение (операция)	W *	С *	Класс
EXP src	$T = \text{number of sign bits (src)} - 8$	1	1	1
FIRS Xmem, Ymem, pmad	$B = B + A * \text{pmad}$ $A = (X\text{mem} + Y\text{mem}) \ll 16$	2	3	8
LMS Xmem, Ymem	$B = B + X\text{mem} * Y\text{mem}$ $A = A + X\text{mem} \ll 16 + 215$	1	1	7
MAX dst	$\text{dst} = \max(A, B)$	1	1	1
MIN dst	$\text{dst} = \min(A, B)$	1	1	1
NEG src [, dst]	$\text{dst} = -\text{src}$	1	1	1
NORM src [, dst]	$\text{dst} = \text{src} \ll TS$ $\text{dst} = \text{norm}(\text{src}, TS)$	1	1	1
POLY Smem	$B = S\text{mem} \ll 16$ $A = \text{rnd}(A(32-16) * T + B)$	1	1	3A, 3B
RND src [, dst]	$\text{dst} = \text{src} + 215$	1	1	1
SAT src	$\text{saturate}(\text{src})$	1	1	1
SQDST Xmem, Ymem	$B = B + A(32-16) * A(32-16)$ $A = (X\text{mem} - Y\text{mem}) \ll 16$	1	1	7
* Количество слов (W) и циклов (C) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных. Добавляется одно слово и один цикл при использовании длинного смещения в косвенной адресации или абсолютной адресации Lmem				

20.1.2 Логические операции

Этот раздел включает команды логических операций. Таблицы ниже содержат команды следующих функциональных групп:

- 1 Команды И (AND) (Таблица 20-7);
- 2 Команды ИЛИ (OR) (Таблица 20-8);
- 3 Команды, исключающие ИЛИ (XOR) (Таблица 20-9);
- 4 Команды сдвига (Таблица 20-10);
- 5 Команды тестирования (Таблица 20-11).

Таблица 20-7 – Команды AND

Синтаксис	Выражение (операция)	W *	С *	Класс
AND Smem, src	$\text{src} = \text{src} \& S\text{mem}$	1	1	3A, 3B
AND #lk [, SHFT], src [, dst]	$\text{dst} = \text{src} \& \#lk \ll \text{SHFT}$	2	2	2
AND #lk, 16, src [, dst]	$\text{dst} = \text{src} \& \#lk \ll 16$	2	2	2
AND src [, SHIFT] [, dst]	$\text{dst} = \text{dst} \& \text{src} \ll \text{SHIFT}$	1	1	1
ANDM #lk, Smem	$S\text{mem} = S\text{mem} \& \#lk$	2	2	18A, 18B
* Количество слов (W) и циклов (C) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных. Добавляется одно слово и один цикл при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem				

Таблица 20-8 – Команды OR

Синтаксис	Выражение (операция)	W *	С *	Класс
OR Smem, src	src = src Smem	1	1	3А, 3В
OR #lk [, SHFT], src [, dst]	dst = src #lk << SHFT	2	2	2
OR #lk, 16, src [, dst]	dst = src #lk << 16	2	2	2
OR src [, SHIFT] [, dst]	dst = dst src << SHIFT	1	1	1
ORM #lk, Smem	Smem = Smem #lk	2	2	18А, 18В
* Количество слов (W) и циклов (C) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных. Добавляется одно слово и один цикл при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem				

Таблица 20-9 – Команды XOR

Синтаксис	Выражение (операция)	W *	С *	Класс
XOR Smem, src	src = src ^ Smem	1	1	3А, 3В
XOR #lk [, SHFT], src [, dst]	dst = src ^ #lk << SHFT	2	2	2
XOR #lk, 16, src [, dst]	dst = src ^ #lk << 16	2	2	2
XOR src [, SHIFT] [, dst]	dst = dst ^ src << SHIFT	1	1	1
XORM #lk, Smem	Smem = Smem ^ #lk	2	2	18А, 18В
* Количество слов (W) и циклов (C) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных. Добавляется одно слово и один цикл при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem				

Таблица 20-10 – Команды сдвига

Синтаксис	Выражение (операция)	W *	С *	Класс
ROL src	циклический сдвиг влево через перенос	1	1	1
ROLTC src	циклический сдвиг влево через ТС	1	1	1
ROR src	циклический сдвиг вправо через перенос	1	1	1
SFTA src, SHIFT [, dst]	dst = src << SHIFT {арифметический сдвиг}	1	1	1
SFTC src	если src(31) = src(30) то src = src << 1	1	1	1
SFTL src, SHIFT [, dst]	dst = src << SHIFT {логический}	1	1	1
* Количество слов (W) и циклов (C) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных				

Таблица 20-11 – Команды тестирования

Синтаксис	Выражение (операция)	W *	C *	Класс
BIT Xmem, BITC	TC = Xmem(15 – BITC)	1	1	3A
BITF Smem, #lk	TC = (Smem && #lk)	2	2	6A, 6B
BITT Smem	TC = Smem(15 – T(3–0))	1	1	3A, 3B
CMPM Smem, #lk	TC = (Smem == #lk)	2	2	6A, 6B
CMPR CC, ARx	сравнивает ARx с AR0	1	1	1
* Количество слов (W) и циклов (C) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных. Добавляется одно слово и один цикл при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem				

20.1.3 Команды управления программой

Этот раздел включает команды управления ходом программы. Таблицы ниже содержат команды следующих функциональных групп:

- 1 Команды перехода (Таблица 20-12);
- 2 Команды вызова (Таблица 20-13);
- 3 Команды прерывания (Таблица 20-14);
- 4 Команды возврата (Таблица 20-15);
- 5 Команды повторения (Таблица 20-16);
- 6 Команды управления стеком (Таблица 20-17);
- 7 Разные команды управления программой (Таблица 20-18).

Таблица 20-12 – Команды перехода

Синтаксис	Выражение (операция)	W ¹⁾	C ¹⁾	Класс
B[D] pmad	PC = pmad(15–0)	2	4/[2 ⁴]	29A
BACC[D] src	PC = src(15–0)	1	6/[4 ⁴]	30A
BANZ[D] pmad, Sind	if (Sind _ 0) then PC = pmad(15–0)	2	4 ² /2 ³ /[2 ⁴]	29A
BC[D] pmad, cond [, cond [, cond]]	if (cond(s)) then PC = pmad(15–0)	2	5 ² /3 ³ /[3 ⁴]	31A
FB[D] extpmad (в данной модели не реализовано)	PC = pmad(15–0), XPC = pmad(22–16)	2	4/[2 ⁴]	29A
FBACC[D] src (в данной модели не реализовано)	PC = src(15–0), XPC = src(22–16)	1	6/[4 ⁴]	30A
¹⁾ Количество слов (W) и циклов (C) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных. ²⁾ Условие истинно. ³⁾ Условие ложно. ⁴⁾ Задержанная команда				

Таблица 20-13 – Команды вызова

Синтаксис	Выражение (операция)	W ¹⁾	C ¹⁾	Класс
CALA[D] src	--SP, PC + 1[3 ⁴⁾] = TOS, PC = src(15-0)	1	6/[4 ⁴⁾]	30B
CALL[D] pmad	--SP, PC + 2[4 ⁴⁾] = TOS, PC = pmad(15-0)	2	4/[2 ³⁾]	29B
CC[D] pmad, cond [, cond [, cond]]	if (cond(s)) then --SP, PC + 2[4 ⁴⁾] = TOS, PC = pmad(15-0)	2	5 ^{2)/3^{3)/[3⁴⁾]}}	31B
FCALA[D] src (в данной модели не реализовано)	--SP, PC + 1 [3 ⁴⁾] = TOS, PC = src(15-0), XPC = src(22-16)	1	6/[4 ⁴⁾]	30B
FCALL[D] extpmad (в данной модели не реализовано)	--SP, PC + 2[4 ⁴⁾] = TOS, PC = pmad(15-0), XPC = pmad(22-16)	2	4/[2 ⁴⁾]	29B
¹⁾ Количество слов (W) и циклов (C) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных. ²⁾ Условие истинно. ³⁾ Условие ложно. ⁴⁾ Задержанная команда				

Таблица 20-14 – Команды прерывания

Синтаксис	Выражение (операция)	W *	C *	Класс
INTR K	--SP, ++ PC = TOS, PC = IPTR(15-7) + K << 2, INTM = 1	1	3	35
TRAP K	--SP, ++ PC = TOS, PC = IPTR(15-7) + K << 2	1	3	35
* Количество слов (W) и циклов (C) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных				

Таблица 20-15 – Команды возврата

Синтаксис	Выражение (операция)	W ¹⁾	C ¹⁾	Класс
FRET[D] (в данной модели не реализовано)	XPC = TOS, ++ SP, PC = TOS, ++SP	1	6/[4 ⁴⁾]	34
FRETE[D] (в данной модели не реализовано)	XPC = TOS, ++ SP, PC = TOS, ++SP, INTM = 0	1	6/[4 ⁴⁾]	34
RC[D] cond [, cond [, cond]]	if (cond(s)) then PC = TOS, ++SP	1	5 ^{2)/3^{3)/[3⁴⁾]}}	32
RET[D]	PC = TOS, ++SP	1	5/[3 ⁴⁾]	32
RETE[D]	PC = TOS, ++SP, INTM = 0	1	5/[3 ⁴⁾]	32
RETF[D]	PC = RTN, ++SP, INTM = 0	1	3/[1 ⁴⁾]	33
¹⁾ Количество слов (W) и циклов (C) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных. ²⁾ Условие истинно. ³⁾ Условие ложно. ⁴⁾ Задержанная команда				

Таблица 20-16 – Команды повторения

Синтаксис	Выражение (операция)	W ¹⁾	C ¹⁾	Класс
RPT Smem	Repeat single, RC = Smem	1	3	5A, 5B
RPT #K	Repeat single, RC = #K	1	1	1
RPT #lk	Repeat single, RC = #lk	2	2	2
RPTB[D] pmad	Repeat block, RSA = PC + 2[4 ²⁾], REA = pmad, BRAF = 1	2	4/[2 ²⁾]	29A
RPTZ dst, #lk	Repeat single, RC = #lk, dst = 0	2	2	2
¹⁾ Количество слов (W) и циклов (C) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных. Добавляется одно слово и один цикл при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem. ²⁾ Задержанная команда				

Таблица 20-17 – Команды управления стеком

Синтаксис	Выражение (операция)	W *	C *	Класс
FRAME K	SP = SP + K	1	1	1
POPD Smem	Smem = TOS, ++SP	1	1	17A, 17B
POPM MMR	MMR = TOS, ++SP	1	1	17A
PSHD Smem	--SP, Smem = TOS	1	1	16A, 16B
PSHM MMR	--SP, MMR = TOS	1	1	16A
* Количество слов (W) и циклов (C) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных. Добавляется одно слово и один цикл при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem				

Таблица 20-18 – Разные команды управления программой

Синтаксис	Выражение (операция)	W *	C *	Класс
IDLE K	idle(K)	1	4	36
MAR Smem	If CMPT = 0, then modify ARx If CMPT = 1 and ARx _ AR0, then modify ARx, ARP = x If CMPT = 1 and ARx = AR0, then modify AR(ARP)	1	1	1,2
NOP	no operation	1	1	1
RESET	software reset	1	3	35
RSBX N, SBIT	STN (SBIT) = 0	1	1	1
SSBX N, SBIT	STN (SBIT) = 1	1	1	1
XC n , cond [, cond [, cond]]	If (cond(s)) then execute the next n instructions; n = 1 or 2	1	1	1
* Количество слов (W) и циклов (C) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных. Добавляется 1 слово и 1 цикл t при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem				

20.1.4 Команды загрузки и сохранения

Этот раздел включает команды загрузки и сохранения. Таблицы ниже содержат команды следующих функциональных групп:

- 1 Команды загрузки (Таблица 20-19);
- 2 Команды сохранения (Таблица 20-20);
- 3 Команды условного сохранения (Таблица 20-21);
- 4 Команды параллельной загрузки и сохранения (Таблица 20-22);
- 5 Команды параллельной загрузки и умножения (Таблица 20-23);
- 6 Команды параллельного сохранения и сложения/вычитания (Таблица 20-24);
- 7 Команды параллельного сохранения и умножения (Таблица 20-25);
- 8 Разные команды типа загрузки и сохранения (Таблица 20-26).

Таблица 20-19 – Команды загрузки

Синтаксис	Выражение (операция)	W *	C *	Класс
DLD Lmem, dst	dst = Lmem	1	1	9A, 9B
LD Smem, dst	dst = Smem	1	1	3A, 3B
LD Smem, TS, dst	dst = Smem << TS	1	1	3A, 3B
LD Smem, 16, dst	dst = Smem << 16	1	1	3A, 3B
LD Smem [, SHIFT], dst	dst = Smem << SHIFT	2	2	4A, 4B
LD Xmem, SHFT, dst	dst = Xmem << SHFT	1	1	3A
LD #K, dst	dst = #K	1	1	1
LD #lk [, SHFT], dst	dst = #lk << SHFT	2	2	2
LD #lk, 16, dst	dst = #lk << 16	2	2	2
LD src, ASM [, dst]	dst = src << ASM	1	1	1
LD src [, SHIFT], dst	dst = src << SHIFT	1	1	1
LD Smem, T	T = Smem	1	1	3A, 3B
LD Smem, DP	DP = Smem(8–0)	1	3	5A, 5B
LD Smem, DP	DP = #k9	1	1	1
LD #k5, ASM	ASM = #k5	1	1	1
LD #k3, ARP	ARP = #k3	1	1	1
LD Smem, ASM	ASM = Smem(4–0)	1	1	3A, 3B
LDM MMR, dst	dst = MMR	1	1	3A
LDR Smem, dst	dst = rnd(Smem)	1	1	3A, 3B
LDU Smem, dst	dst = uns(Smem)	1	1	3A, 3B
LTD Smem	T = Smem, (Smem + 1) = Smem	1	1	24A, 24B
* Количество слов (W) и циклов (C) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных. Добавляется одно слово и один цикл при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem или Lmem				

Таблица 20-20 – Команды сохранения

Синтаксис	Выражение (операция)	W *	С *	Класс
DST src, Lmem	Lmem = src	1	2	13A, 13B
ST T, Smem	Smem = T	1	1	10A, 10B
ST TRN, Smem	Smem = TRN	1	1	10A, 10B
ST #lk, Smem	Smem = #lk	2	2	12A, 12B
STH src, Smem	Smem = src << -16	1	1	10A, 10B
STH src, ASM, Smem	Smem = src << (ASM - 16)	1	1	10A, 10B
STH src, SHFT, Xmem	Xmem = src << (SHFT - 16)	1	1	10A
STH src [, SHIFT], Smem	Smem = src << (SHIFT - 16)	2	2	11A, 11B
STL src, Smem	Smem = src	1	1	10A, 10B
STL src, ASM, Smem	Smem = src << ASM	1	1	10A, 10B
STL src, SHFT, Xmem	Xmem = src << SHFT	1	1	10A, 10B
STL src [, SHIFT], Smem	Smem = src << SHIFT	2	2	11A, 11B
STLM src, MMR	MMR = src	1	1	10A
STM #lk, MMR	MMR = #lk	2	2	12A
* Количество слов (W) и циклов (C) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных. Добавляется одно слово и один цикл при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem или Lmem				

Таблица 20-21 – Команды условного сохранения

Синтаксис	Выражение (операция)	W *	С *	Класс
CMPS src, Smem	If src(31-16) > src(15-0) then Smem = src(31-16) If src(31-16) _ src(15-0) then Smem = src(15-0)	1	1	10A, 10B
SACCD src, Xmem, cond	If (cond) Xmem = src << (ASM - 16)	1	1	15
SRCCD Xmem, cond	If (cond) Xmem = BRC	1	1	15
STRCD Xmem, cond	If (cond) Xmem = T	1	1	15
* Количество слов (W) и циклов (Ц) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных. Добавляется 1 слово и 1 цикл при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem				

Таблица 20-22 – Команды параллельной загрузки и сохранения

Синтаксис	Выражение (операция)	W *	С *	Класс
ST src, Ymem LD Xmem, dst	Ymem = src << (ASM _ 16) dst = Xmem << 16	1	1	14
ST src, Ymem LD Xmem, T	Ymem = src << (ASM - 16) T = Xmem	1	1	14
* Количество слов (W) и циклов (C) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных				

Таблица 20-23 – Команды параллельной загрузки и умножения

Синтаксис	Выражение (операция)	W *	C *	Класс
LD Xmem, dst MAC Ymem, dst_	dst = Xmem << 16 dst_ = dst_ + T * Ymem	1	1	7
LD Xmem, dst MACR Ymem, dst_	dst = Xmem << 16 dst_ = rnd(dst_ + T * Ymem)	1	1	7
LD Xmem, dst MAS Ymem, dst_	dst = Xmem << 16 dst_ = dst_ – T * Ymem	1	1	7
LD Xmem, dst MASR Ymem, dst_	dst = Xmem << 16 dst_ = rnd(dst_ – T * Ymem)	1		7
* Количество слов (W) и циклов (C) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных				

Таблица 20-24 – Команды параллельного сохранения и сложения/вычитания

Синтаксис	Выражение (операция)	W *	C *	Класс
ST src, Ymem ADD Xmem, dst	Ymem = src << (ASM _ 16) dst = dst_ + Xmem << 16	1	1	14
ST src, Ymem SUB Xmem, dst	Ymem = src << (ASM – 16) dst = (Xmem << 16) – dst_	1	1	14
* Количество слов (W) и циклов (C) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных				

Таблица 20-25 – Команды параллельного сохранения и умножения

Синтаксис	Выражение (операция)	W *	C *	Класс
ST src, Ymem MAC Xmem, dst	Ymem = src << (ASM – 16) dst = dst + T * Xmem	1	1	14
ST src, Ymem MACR Xmem, dst	Ymem = src << (ASM – 16) dst = rnd(dst + T * Xmem)	1	1	14
ST src, Ymem MAS Xmem, dst	Ymem = src << (ASM – 16) dst = dst – T * Xmem	1	1	14
ST src, Ymem MASR Xmem, dst	Ymem = src << (ASM – 16) dst = rnd(dst – T * Xmem)	1	1	14
ST src, Ymem MPY Xmem, dst	Ymem = src << (ASM – 16) dst = T * Xmem	1	1	14
* Количество слов (W) и циклов (C) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных				

Таблица 20-26 – Разные программы типа загрузки и сохранения

Синтаксис	Выражение (операция)	W *	C *	Класс
MVDD Xmem, Ymem	Ymem = Xmem	1	1	14
MVDK Smem, dmad	dmad = Smem	2	2	19A, 19B
MVDM dmad, MMR	MMR = dmad	2	2	19A
MVDP Smem, pmad	pmad = Smem	2	4	20A, 20B
MVKD dmad, Smem	Smem = dmad	2	2	19A, 19B
MVMD MMR, dmad	dmad = MMR	2	2	19A
MVMM MMRx, MMRy	MMRy = MMRx	1	1	1
MVPD pmad, Smem	Smem = pmad	2	3	21A, 21B
PORTR PA, Smem	Smem = PA	2	2	27A, 27B
PORTW Smem, PA	PA = Smem	2	2	28A, 28B
READA Smem	Smem = A	1	5	25A, 25B
WRITA Smem	A = Smem	1	5	26A, 26B
* Количество слов (W) и циклов (C) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных. Добавляется одно слово и один цикл при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem				

20.1.5 Повторения одной команды

Процессор имеет команды повторения, которые заставляют следующую команду повториться нужное число раз. Количество повторений команды равно значению операнда команды повторения + 1. Это значение сохраняется в 16-разрядном регистре счетчика повторений (RC). Невозможно запрограммировать значение в регистре RC, оно загружается исключительно командами повторения. Максимальное число повторений команды – 65 536. Абсолютный адрес памяти программ или данных автоматически увеличивается при каждом повторении команды.

Как только команда повторения декодирована, все прерывания, включая NMI, но не RS, заблокированы до завершения цикла повторения. Однако процессор реагирует на сигнал HOLD при выполнении цикла повторения в соответствии со значением бита NM в регистре состояния 1 (ST1).

Функция повторения может использоваться с некоторыми командами типа умножение/накопление и блочные пересылки, для увеличения скорости выполнения этих команд. Эти многоцикловые команды (Таблица 20-27) становятся одноцикловыми после первой итерации команды повторения.

Таблица 20-27 – Многоцикловые команды, выполняемые за один цикл при повторении

Команда	Описание	# Циклы *
FIRS	Симметричный фильтр FIR	3
MACD	Умножаются с аккумулярованием и задержкой	3
MACP	Умножаются с аккумулярованием	3
MVDK	Пересылка данные-данные	2
MVDM	Пересылка данные-регистры MMR	2
MVDP	Пересылка данные-программа	4

Команда	Описание	# Циклы *
MVKD	Персылка данные-данные	2
MVMD	Пресылка регистры MMR – данные	2
MVPD	Пересылка программа-данные	3
READA	Чтение из памяти программ в память данных	5
WRITA	Запись из памяти данных в память программ	5
* Число циклов, когда команда не повторяется		

Отдельные команды одного операнда в памяти данных не могут быть повторены при использовании длинного смещения или абсолютного адреса (например, *ARn(lk), *+ARn(lk), *+ARn(lk)% и *(lk)). Таблица 20-28 содержит команды, которые не могут быть повторены командами RPT или RPTZ.

Таблица 20-28 – Неповторяемые команды

Команда	Описание
ADDM	Прибавить Длинную Константу К Памяти Данных
ANDM	AND Память Данных С Длинной Константой
B[D]	Безусловный Переход
BACC[D]	Переход По Адресу Из Аккумулятора
BANZ[D]	Переход По Вспомогательному Регистру Не Равному 0
BC[D]	Условный Переход
CALA[D]	Вызов Подпрограммы По Адресу Из Аккумулятора
CALL[D]	Безусловный Вызов
CC[D]	Условный Вызов
CMPR	Сравнение Вспомогательных Регистров
DST	Сохранение Длинного Слова (32-Бита)
FB[D]	Дальний Безусловный Переход (В Данной Модели Не Реализовано)
FBACC[D]	Дальний Переход По Адресу Из Аккумулятора(В Данной Модели Не Реализовано)
FCALA[D]	Дальний Вызов Подпрограммы По Адресу Из Аккумулятора (В Данной Модели Не Реализовано)
FCALL[D]	Дальний Безусловный Вызов (В Данной Модели Не Реализовано)
FRET[D]	Дальний Возврат(В Данной Модели Не Реализовано)
FRETE[D]	Разрешение Прерывания И Дальний Возврат(В Данной Модели Не Реализовано)
IDLE	Команда Ожидания
INTR	Прерывание
LD ARP	Загрузка Указателя Вспомогательного Регистра (ARP)
LD DP	Загрузка Указателя Страницы Данных (DP)
MVMM	Пересылка Регистров, Отображаемых На Память (MMR)
ORM	OR Память Данных С Длинной Константой
RC[D]	Условный Возврат
RESET	Программный Сброс
RET[D]	Безусловный Возврат
RETE[D]	Возврат Из Прерывания
RETF[D]	Быстрый Возврат Из Прерывания

Команда	Описание
RND	Округление Аккумулятора
RPT	Повторение Следующей Команды
RPTB[D]	Повторение Блока
RPTZ	Повторение Следующей Команды И Очистка Аккумулятора
RSBX	Сброс Бита Статусного Регистра
SSBX	Установка Бита Статусного Регистра
TRAP	Программное Прерывание
XC	Условное Выполнение Команды
XORM	XOR Память Данных С Длинной Константой

20.2 Классы и циклы команд

Команды классифицируются по нескольким категориям, или классам, в соответствии с требуемым количеством циклов выполнения команд. Этот раздел описывает классы команд. Поскольку одна команда может иметь несколько синтаксисов и типов исполнения, то она может появляться в нескольких классах.

Надо отметить, что данная глава имеет несколько условный характер деления на классы и возникла как перифраз главы описывающей прототип. Основанием для этого замечания является то, что деление всех команд на классы в прототипе предположительно связано с иерархической организацией схем управления, как средство преодоления сложности этого управления. В нашем случае управление ковейером оригинальное и деление на классы не связано с реализацией схем управления. Тем не менее, номера классов и принадлежность команд к тому или иному классу сохранена, хотя количество циклов может быть другим, нежели в прототипе. Тем не менее, так как алгоритмы выполнения инструкций те же, можно говорить о классах и циклах выполнения команд, конечно скорректировав как данные, так и допущения при рассмотрении в соответствии с данной реализацией процессора.

Таблицы в этой главе иллюстрируют количество циклов, требуемое для выполнения команды в данной конфигурации памяти при выполнении как одной команды, так и выполнении в режиме повторения. В таблицах также указан доступ операнда памяти данных, используемый с длинной константой. В первом столбце таблицы указано расположение источника программы. Эти заголовки определяются следующим образом:

- ROM** команды выполняется из внутреннего ПЗУ программ (ROM);
- RAM** команда выполняется из внутреннего ОЗУ данных (RAM);
- External** команда выполняется из внешней памяти программ.

Если класс команды требует наличие операнда(ов) памяти, то расположение операнда(ов) указывается в строках таблицы. Эти расположения определяются следующим образом:

- RAM** операнд(ы) имеет доступ к одной (двум) внутренним ячейкам в ОЗУ;
- DROM** операнд во внутренней памяти данных ПЗУ;
- PROM** операнд во внутренней памяти программ;
- External** операнд во внешней памяти;
- MMR** операнд является регистром, отображаемым в памяти.

Количество циклов, необходимое для каждой команды определяется в соответствии рабочими циклами процессора (период CLKOUT). Дополнительные состояния ожидания доступа к памяти программ/данных и к вводу/выводу определяются следующим образом:

d состояния ожидания памяти данных – количество дополнительных циклов, которое устройство ожидает для получения разрешения доступа к внешней ячейке памяти данных;

io состояние ожидания ввода/вывода – количество дополнительных циклов, которое устройство ожидает для получения разрешения доступа к внешнему вводу-выводу;

n повторения – количество выполнений повторяющейся команды;

nd состояние ожидания памяти данных, повторяющееся n раз;

np состояние ожидания памяти программ, повторяющееся n раз;

npd состояние ожидания памяти программ и данных, повторяющееся n раз;

p состояния ожидания памяти программ – количество дополнительных циклов, которое устройство ожидает для получения разрешения доступа к внешней ячейке памяти программ;

pd состояния ожидания памяти программ – количество дополнительных циклов, которое устройство ожидает для получения разрешения доступа к операнду памяти программ.

Эти переменные могут также использовать нижние индексы src, dst и code, чтобы обозначить источник, пункт назначения и код, соответственно.

Любое чтение из внешней ячейки памяти занимает, по меньшей мере, один полный цикл выполнения команды, а любая запись во внешнюю ячейку памяти занимает, по меньшей мере, два полных цикла выполнения команды.

Эти доступы к внешним ячейкам занимают больше времени, если дополнительные циклы ожидания добавляются в результате использования программируемого генератора состояний ожидания или внешнего входа READY.

Однако, любая запись из ЦП во внешнюю ячейку памяти занимает только один цикл до тех пор, пока нет ни одного другого доступа к внешней памяти в то же самое время.

Это возможно, потому что конвейер выполнения команд занимает только один цикл запроса доступа записи во внешнюю память, а блок интерфейса шины завершает ответ на доступ к записи немедленно. Запрос на запись фиксируется и в дальнейшем завершается.

В таблицах, приведенных ниже, не учтен дополнительный цикл, возможный на фоне других доступов во внешнюю память.

Количество циклов выполнения команд получается при следующих допущениях:

Не рассматривается конфликт между извлечением команды из ОЗУ и доступом по чтению или записи данных в тот же блок расслоенного ОЗУ.

Не рассматриваются любые конфликты по данным, вызванные конвейерным выполнением команды (например, типа RAW-read after write).

Класс 1

Одно слово, один цикл. Операнды отсутствуют, или отсутствуют операнды памяти при наличии коротких непосредственных или регистровых операндов.

Мнемоника:

ABS	MACA[R]	NORM	SFTA
ADD	MAR	OR	SFTC
AND	MASA[R]	RND	SFTL
CMPL	MAX	ROL	SQUR
CMPR	MIN	ROLTC	SSBX
EXP	MPYA	ROR	SUB
FRAME	MVMM	RPT	XC
LD	NEG	RSBX	XOR
LD T/DP/ASM/ARP	NOP	SAT	

Циклы

Циклы единичного выполнения		
Программа		
ROM	RAM	External
1	1	1+p

Циклы повторяющихся выполнений		
Программа		
ROM	RAM	External
n	n	n+p

Класс 2

Два слова, два цикла. Длинный непосредственный операнд и отсутствие операндов памяти.

Мнемоника:

ADD	MAC	OR	SUB
AND	MAR	RPT	XOR
LD	MPY	RPTZ	

Циклы

Циклы единичного выполнения		
Программа		
ROM	RAM	External
2	2	2+2p

Циклы повторяющихся выполнений		
Программа		
ROM	RAM	External
n+1	n+1	n+1+2p

Класс 3А

Одно слово, один цикл. Операнд чтения в памяти данных (Smem или Xmem) или операнд чтения MMR.

Мнемоника

ADD	LDM	MPYA	SUBB
ADDC	LDR	MPYU	SUBC
ADDS	LDU	OR	SUBS
AND	MAC[R]	POLY	XOR
BIT	MACA[R]	SQUR	
BITT	MAS[R]	SQURA	
LD	MASA	SQURS	
LD T/DP/ASM/ARP	MPY[R]	SUB	

Циклы

Циклы единичного выполнения			
Операнд	Программа		
Smem	ROM	RAM	External
RAM	1	1,2*	1+p
DROM	2	1	1+p
External	1+d	1+d	2+d+p
MMR	1	1	1+p
* Операнд и код в одном блоке памяти			

Циклы повторяющихся выполнений			
Операнд	Программа		
Smem	ROM	RAM	External
RAM	n	n, n+1*	n+p
DROM	n+1	n	n+p
External	n+nd	n+nd	n+1+nd+p
MMR	n	n	n+p
* Операнд и код в одном блоке памяти			

Класс 3В

Два слова, два цикла. Операнд чтения в памяти данных (Smem), использующий косвенную адресацию с длинным смещением.

Мнемоника

ADD	LDU	OR	SUBS
ADDC	MAC[R]	POLY	XOR
ADDS	MACA[R]	SQUR	
AND	MAS[R]	SQURA	
BITT	MASA	SQURS	
LD	MPY[R]	SUB	
LD T/DP/ASM/ARP	MPYA	SUBB	
LDR	MPYU	SUBC	

Циклы

Циклы единичного выполнения, использующие косвенную адресацию с длинным смещением			
Операнд	Программа		
Smem	ROM	RAM	External
RAM	2	2, 3*	2+2p
DROM	3	2	2+2p
External	2+d	2+d	3+d+2p
MMR	2	2	2+2p
* Операнд и код в одном блоке памяти			

Класс 4А

Два слова, два цикла. Операнд чтения в памяти данных (Smem).

Мнемоника

ADD LD SUB

Циклы

Циклы единичного выполнения, использующие косвенную адресацию с длинным смещением			
Операнд	Программа		
Smem	ROM	RAM	External
RAM	2	2, 3*	2+2p
DROM	3	2	2+2p
External	2+d	2+d	3+d+2p
MMR	2	2	2+2p
* Операнд и код в одном блоке памяти			

Циклы повторяющихся выполнений			
Операнд	Программа		
Smem	ROM	RAM	External
RAM	n+1	n+1, n+2*	n+1+2p
DROM	n+2	n+1	n+1+2p
External	n+1+nd	n+1+nd	n+2+nd+2p
MMR	n+1	n+1	n+1+2p
* Операнд и код в одном блоке памяти			

Класс 4В

Три слова, Три цикла. Операнд чтения в памяти данных (Smem) с использованием косвенной адресации с длинным смещением.

Мнемоника

ADD LD SUB

Циклы

Циклы единичного выполнения, использующие косвенную адресацию с длинным смещением			
Операнд	Программа		
Smem	ROM	RAM	External
RAM	3	3, 4*	3+3p
DROM	4	3	3+3p
External	3+d	3+d	4+d+3p
MMR	3	3	3+3p
* Операнд и код в одном блоке памяти			

Класс 5А

Одно слово, один цикл. Операнд чтения в памяти данных (Smem) (с назначением в качестве приемника регистра указателя страницы памяти данных – DP).

Мнемоника

LD

Циклы

Циклы единичного выполнения			
Операнд	Программа		
Smem	ROM	RAM	External
RAM	1	1	1+p
DROM	2	1	1+p
External	1+d	1+d	2+d+p
MMR	1	1	1+p

Класс 5В

Два слова, два цикла. Операнд чтения в памяти данных (Smem) с использованием косвенной адресации с длинным смещением (с назначением в качестве приемника регистра указателя страницы памяти данных – DP).

Мнемоника

LD

Циклы

Циклы единичного выполнения, использующие косвенную адресацию с длинным смещением			
Операнд	Программа		
Smem	ROM	RAM	External
RAM	2	2,3*	2+2p
DROM	3	2	2+2p
External	2+d	2+d	3+d+2p
MMR	2	2	2+2p
* Операнд и код в одном блоке памяти			

Класс 6А

Два слова, два цикла. Операнд чтения в памяти данных (Smem) и операнд с длинным непосредственным значением.

Мнемоника

BITF

CMPM

MAC MPY

Циклы

Циклы единичного выполнения			
Операнд	Программа		
Smem	ROM	RAM	External
RAM	2	2, 3*	2+2p
DROM	3	2	2+2p
External	2+d	2+d	3+d+2p
MMR	2	2	2+2p
* Операнд и код в одном блоке памяти			

Циклы повторяющихся выполнений			
Операнд	Программа		
Smem	ROM	RAM	External
RAM	n+1	n+1, n+2*	n+1+2p
DROM	n+2	n+1	n+1+2p
External	n+1+nd	n+1+nd	n+2+nd+2p
MMR	n+1	n+1	n+1+2p
* Операнд и код в одном блоке памяти			

Класс 6В

Три слова, Три цикла. Операнд чтения в памяти данных (Smem), использующий косвенную адресацию с длинным смещением и операнд с длинным непосредственным значением.

Мнемоника

BITF

CMPM

MAC

MPY

Циклы

Циклы единичного выполнения, использующие косвенную адресацию с длинным смещением			
Операнд	Программа		
Smem	ROM	RAM	External
RAM	3	3, 4*	3+3p
DROM	4	3	3+3p
External	3+d	3+d	4+d+3p
MMR	3	3	3+3p
* Операнд и код в одном блоке памяти			

Класс 7

Одно слово, один цикл. Операнды двойного чтения в памяти данных (Xmem и Ymem).

Мнемоника

ABDST	LD MAS[R]	MACSU	SQDST
ADD	LMS	MAS[R]	SUB
LD MAC[R]	MAC[R]	MPY	

Циклы

Циклы единичного выполнения				
Операнд		Программа		
Xmem	Ymem	ROM	RAM	External
RAM	RAM	1, 2 ¹⁾	1, 2 ²⁾ , 3 ³⁾	1+p, 2 ⁵⁾
	DROM	2	1, 2 ²⁾	1+p
	External	1+d	1+d, 2 ⁴⁾	2+d+p
DROM	RAM	2	1, 2 ²⁾	1+p
	DROM	3	2	1+p, 2 ⁵⁾
	External	1+d, 2 ⁴⁾	1+d	2+d+p
External	RAM	1+d	1+d, 2 ²⁾	2+d+p
	DROM	1+d, 2 ⁴⁾	1+d	2+d+p
	External	2+2d	2+2d	3+2d+p
MMR	RAM	1	1, 2 ²⁾	1+p
	DROM	2	1	1+p
	External	1+d	1+d	2+d+p

¹⁾ Операнды в одном блоке памяти;
²⁾ Операнд и код в одном блоке памяти;
³⁾ Два операнда и код в одном блоке памяти;
⁴⁾ Операнд и код в одном блоке памяти при d=0;
⁵⁾ Два операнда в одном блоке памяти при p=0

Циклы повторяющихся выполнений				
Операнд		Программа		
Xmem	Ymem	ROM	RAM	External
RAM	RAM	n, 2n ³⁾	n, 2n#, 2n+1 ²⁾	n+p, 2n (p=0) ³⁾ , 2n-1+p(p≥1) ³⁾
	DROM	n+1	n, n+1 ¹⁾	n+p
	External	n+nd	n+nd, n+nd+1 ¹⁾	n+nd+1+p
DROM	RAM	n+1 ¹⁾	n+1 ¹⁾	n+p
	DROM	2n+1	2n	2n (p=0), 2n-1+p(p≥1)
	External	n+nd+1	n+nd	n+nd+1+p
External	RAM	n+nd	n+nd, n+nd+1 ¹⁾	n+nd+1+p
	DROM	n+nd(d≠0), n+1(d=0)	n+nd	n+nd+1+p
	External	2n+2nd	2n+2nd	2n+2nd+1+p
MMR	RAM	n	n, n+1 ¹⁾	n+p
	DROM	n+1	n	n+p
	External	n+nd	n+nd	n+nd+1+p

¹⁾ Операнд и код в одном блоке памяти;
²⁾ Два операнда и код в одном блоке памяти;
³⁾ Два операнда в одном блоке памяти

Класс 8

Два слова, три цикла. Операнды двойного чтения в памяти данных (Xmem и Ymem) и операнд в памяти программ единичного доступа (pmad).

Мнемоника

FIRS

Циклы

Циклы единичного выполнения					
Операнд			Программа		
pmad	Xmem	Ymem	ROM	RAM	External
RAM	RAM	RAM	2 ¹⁾ , 3 ³⁾	3, 4 ¹⁾	2+p, 3+p ¹⁾
		DROM	2	2, 3 ¹⁾	2+p, 3+p ¹⁾
		External	1+d, 2+d ¹⁾	2+d, 3+d ¹⁾	2+d+p
	DROM	RAM	2	2, 3 ¹⁾	2+p, 3+p ¹⁾
		DROM	3	2	2+p
		External	2+d	2+d	2+d+p
	External	RAM	1+d, 2+d ¹⁾	2+d, 3+d ¹⁾	2+d+p, 3+d+p ¹⁾
		DROM	2+d	2+d	2+d+p
		External	2+2d	2+2d	3+2d+p
DROM	RAM	RAM	2	2, 3 ¹⁾	1+p, 2+p(p=0, ²⁾)
		DROM	3	2	2(p=0), 1+p(p≠0)
		External	2(d=0), 1+d(d≠0)	2(d=0& ¹⁾), 1+d	2+d+p
	DROM	RAM	3	2	2(p=0), 1+p(p≠0)
		DROM	4	3	3+2p
		External	3(d=0), 2+d(d≠0)	2(d=0), 1+d(d≠0)	2+d+p
	External	RAM	2(d=0), 1+d(d≠0)	2(d=0& ¹⁾), 1+d	2+d+p
		DROM	3(d=0), 2+d(d≠0)	2(d=0), 1+d(d≠0)	2+d+p
		External	2+2d	2+2d	3+2d+p
External	RAM	RAM	1+d, 2+d ²⁾	2+d, 3+d ^{1,2)}	2+d+p
		DROM	2(d=0), 1+d(d≠0)	2(d=0& ¹⁾), 1+d	2+d+p
		External	2+2d	2+2d	3+2d+p
	DROM	RAM	2(d=0), 1+d(d≠0)	2(d=0& ¹⁾), 1+d	2+d+p
		DROM	3(d=0), 2+d(d≠0)	2(d=0), 1+d(d≠0)	2+d+p
		External	2+2d	2+2d	3+2d+p
	External	RAM	2+2d	2+2d	3+2d+p
		DROM	2+2d	2+2d	3+2d+p
		External	3+3d	3+3d	4+3d+p

¹⁾ Xmem(Ymem) и pmad в одном блоке памяти;

²⁾ Xmem и Ymem в одном блоке памяти;

³⁾ Xmem, Ymem и pmad в одном блоке памяти

Циклы повторяющихся выполнений ($n > 1$)					
Операнд			Программа		
p _{mad}	X _{mem}	Y _{mem}	ROM	RAM	External
RAM	RAM	RAM	$n, 2n^{1)}, 3n^{3)}$	$n, 2n+1^{1,2)}, 3n+1^{4)}$	$n+p, 2n+p^{1,2)}, 3n+p^{1)}$
		DROM	$n+1, 2n+1^{1)}$	$n, 2n^{1)}, 3n^{1,3)}$	$n+p, 2n+p^{1)}$
		External	$n+nd, 2n(d=0)^{1)}$	$n+nd, 2n(d=0)^{1)}$	$1+n+nd+p, 2n+p(d=0)^{1)}$
	DROM	RAM DROM	$1+n, 1+2n^{1)}$ $2n+1$	$1+n, 1+2n^{1)}$ $2n$	$n+p, 2n+p^{1)}$ $2n+p$
		External	$n+1(d=0), n+nd(d \neq 0)$	$n+nd$	$n+nd+p$
	External	RAM	$n+nd, 2n(d=0)^{1)}$	$n+nd, 2n(d=0)^{1)}$	$1+n+nd+p, 2n+p(d=0)^{1)}$
		DROM	$n+1(d=0), n+nd(d \neq 0)$	$n+nd$	$n+nd+p$
		External	$2n+2nd$	$2n+2nd$	$2n+2nd+p$
DROM	RAM	RAM	$1+n, 1+2n^{1)}$	$1+n, 1+2n^{1)}$	$n+p, 2n+p^{1)}$
		DROM	$2n+1$	$2n$	$2n+p$
		External	$n+1(d=0), n+nd(d \neq 0)$	$n+nd$	$n+nd+p$
	DROM	RAM	$2n+1$	$2n$	$2n+p$
		DROM	$3n+1$	$3n$	$3n+p$
		External	$2n(d=0), n+nd(d \neq 0)$	$2n(d=0), n+nd(d \neq 0)$	$n+nd+p$
	External	RAM	$n+1(d=0), n+nd(d \neq 0)$	$n+nd$	$n+nd+p$
		DROM	$2n(d=0), n+nd(d \neq 0)$	$2n(d=0), n+nd(d \neq 0)$	$n+nd+p$
		External	$2n+2nd$	$2n+2nd$	$2n+2nd+p$
External	RAM	RAM	$n+nd, 2n(d=0)^{1)}$	$n+nd, 2n(d=0)^{1)}$	$1+n+nd+p, 2n+p(d=0)^{1)}$
		DROM	$n+1(d=0), n+nd(d \neq 0)$	$n+nd$	$n+nd+p$
		External	$2n+2nd$	$2n+2nd$	$2n+2nd+p$
	DROM	RAM	$n+1(d=0), n+nd(d \neq 0)$	$n+nd$	$n+nd+p$
		DROM	$2n(d=0), n+nd(d \neq 0)$	$2n(d=0), n+nd(d \neq 0)$	$n+nd+p$
		External	$2n+2nd$	$2n+2nd$	$2n+2nd+p$
	External	RAM	$2n+2nd$	$2n+2nd$	$2n+2nd+p$
		DROM	$2n+2nd$	$2n+2nd$	$2n+2nd+p$
		External	$3n+3nd$	$3n+3nd$	$3n+3nd+p$
¹⁾ X _{mem} и p _{mad} в одном блоке памяти; ²⁾ X _{mem} и Y _{mem} в одном блоке памяти; ³⁾ Y _{mem} и p _{mad} в одном блоке памяти; ⁴⁾ X _{mem} , Y _{mem} и p _{mad} в одном блоке памяти					

Класс 9А

Одно слово, один цикл. Операнд чтения в памяти данных, использующий адресацию длинного слова (Lmem).

Мнемоника

DADD	DLD	DSADT	DSUBT
DADST	DRSUB	DSUB	

Циклы

Циклы единичного выполнения			
Операнд	Программа		
Lmem	ROM	RAM	External
RAM	1	1, 2*	1+p
DROM	2	1	1+p
External	1+d	1+d	2+d+p
* Операнд и команда в одном блоке ОЗУ			

Циклы повторяющихся выполнений			
Операнд	Программа		
Lmem	ROM	RAM	External
RAM	n	n,n+1*	n+p
DROM	n+1	n	n+p
External	n+nd	n+nd	n+nd+p
* Операнд и код в одном блоке ОЗУ			

Класс 9В

Два слова, два цикла. Операнд чтения в памяти данных, использующий косвенную адресацию длинного слова (Lmem) с длинным смещением.

Мнемоника

DADD	DLD	DSADT	DSUBT
DADST	DRSUB	DSUB	

Циклы

Циклы выполнения, использующие косвенную адресацию с длинным смещением			
Операнд	Программа		
Lmem	ROM	RAM	External
RAM	2	1,2*,3*	2+2p
DROM	4	2	2+2p
External	2+2d	2+2d	4+2d+2p
* Операнд и код в одном блоке памяти			

Класс 10А

Одно слово, один цикл. Операнд записи в памяти данных (Smem или Xmem) или операнд записи в MMR.

Мнемоника:

CMPS STH STLM
ST STL

Циклы

Циклы единичного выполнения			
Операнд	Программа		
Smem	ROM	RAM	External
RAM	1	1,2*	1+p
MMR	1	1	1+p
External	1+d	1+d	2+d+p
* Операнд и код в одном блоке памяти			

Циклы повторяющихся выполнений			
Операнд	Программа		
Smem	ROM	RAM	External
RAM	n	n,n+1*	n+p
MMR	n	n	n+p
External	n+nd	n	n+nd+p
* Операнд и код в одном блоке памяти			

Класс 10В

Два слова, два цикла. Операнд записи в память данных (Smem или Xmem) использующий косвенную адресацию с длинным смещением.

Мнемоника

CMPS ST STH STL

Циклы

Циклы единичного выполнения, использующие косвенную адресацию с длинным смещением			
Операнд	Программа		
Smem	ROM	RAM	External
RAM	2	2	2+2p
MMR	2	2	2+2p
External	2+d	2+d	3+d+2p
* Операнд и код в одном блоке памяти			

Класс 11А

Два слова, два цикла. Операнд записи в память данных (Smem).

Мнемоника

STH STL

Циклы

Циклы единичного выполнения			
Операнд	Программа		
Smem	ROM	RAM	External
RAM	2	2,3*	2+2p
MMR	2	2	2+2p
External	1+d	1+d	3+d+2p
* Операнд и код в одном блоке памяти			

Циклы повторяющихся выполнений			
Операнд	Программа		
Smem	ROM	RAM	External
RAM	n+1	n+1,n+2*	n+1+2p
MMR	n+1	n+1	n+1+2p
External	n+nd,n+1(d=0)	n+nd,n+1(d=0)	n+nd+2+2p
* Операнд и код в одном блоке памяти			

Класс 11В

Три слова, три цикла Операнд записи в память данных (Smem) использующий косвенную адресацию с длинным смещением.

Мнемоника

STH STL

Циклы

Циклы единичного выполнения, использующие косвенную адресацию с длинным смещением			
Операнд	Программа		
Smem	ROM	RAM	External
RAM	3	3,4*	3+3p
MMR	3	3	3+3p
External	3,1+d(d≥2)	3,1+d(d≥2)	4+d+3p
* Операнд и код в одном блоке памяти			

Класс 12А

Два слова, два цикла. Операнд записи в память данных (Smem) или операнд записи в MMR.

Мнемоника

ST STM

Циклы:

Циклы единичного выполнения			
Операнд	Программа		
Smem	ROM	RAM	External
RAM	2	2, 3*	2+2p
MMR	2	2	2+2p
External	2,1+d(d≥2)	2,1+d(d≥2)	3+d+2p
* Операнд и код в одном блоке памяти			

Циклы повторяющихся выполнений			
Операнд	Программа		
Smem	ROM	RAM	External
RAM	2n	2n,2n+1*	2n+2p
MMR	2n	2n	2n+2p
External	2n+(n-1)d	2n+(n-1)d	3+n+nd+2p
* Операнд и код в одном блоке памяти			

Класс 12В

Три слова, три цикла. Операнд записи в память данных (Smem), использующий косвенную адресацию с длинным смещением.

Мнемоника

ST

Циклы

Циклы единичного выполнения, использующие косвенную адресацию с длинным смещением			
Операнд	Программа		
Smem	ROM	RAM	External
RAM	3	3, 4*	3+3p
MMR	3	3	3+3p
External	3,1+d(d≥2)	3,1+d(d≥2)	4+d+3p
* Операнд и код в одном блоке памяти			

Класс 13А

Одно слово, два цикла. Операнд записи в памяти данных, использующий адресацию длинного слова (Lmem).

Мнемоника

DST

Циклы

Циклы единичного выполнения			
Операнд	Программа		
Smem	ROM	RAM	External
RAM	1	1,2*	$2(p=0), 1+p(p>0)$
MMR	1	1	$2(p=0), 1+p(p>0)$
External	$2+2d$	$2+2d$	$3+2d+p$
* Операнд и код в одном блоке памяти			

Циклы повторяющихся выполнений			
Операнд	Программа		
Smem	ROM	RAM	External
RAM	n	$n, n+1^*$	$n+p+1$
MMR	n	2n	$n+p+1$
External	$2n+2nd$	$2n+2nd$	$2n+2nd+1+p$
* Операнд и код в одном блоке памяти			

Класс 13В

Два слова, три цикла. Операнд записи в памяти данных, использующий косвенную адресацию длинного слова (Lmem) с длинным смещением.

Мнемоника

DST

Циклы

Циклы единичного выполнения, использующие косвенную адресацию с длинным смещением			
Операнд	Программа		
Lmem	ROM	RAM	External
RAM	2	2,3*	$2+2p$
External	$2+2d$	$2+2d$	$4+2d+2p$
MMR	2	2	$2+2p$
* Операнд и код в одном блоке памяти			

Класс 14

Одно слово, один цикл. Операнды чтения и записи в расслоенной памяти данных (Xmem и Ymem).

Мнемоника

MVDD	ST LD	ST MAS[R]	ST SUB
ST ADD	ST MAC[R]	ST MPY	

Циклы

Циклы единичного выполнения				
Операнд		Программа		
Xmem	Ymem	ROM	RAM	External
RAM	RAM	1, 2 ¹⁾	1,2 ¹⁾ ,3 ^{1,2)}	1+p,2(p=0, ²⁾)
	External	1+d	1+d	2+d+p
DROM	RAM	2	1,2 ¹⁾	1+p
	External	1+d, 2(d=0, ¹⁾)	1	2+d+p
External	RAM	1+d	1+d,2(d=0, ¹⁾)	2+d+p
	External	2+2d	2+2d	3+2d+p
MMR	RAM	1	1, 2 ¹⁾	1+p
	External	1+d	1+d	2+d+p
¹⁾ Операнд и код в одном блоке памяти. ²⁾ Два операнда в одном блоке памяти				

Циклы повторяющихся выполнений				
Операнд		Программа		
Xmem	Ymem	ROM	RAM	External
RAM	RAM	n,2n ³⁾	n,n+1 ¹⁾ ,2n ³⁾ , 2n+1 ^{1,3)}	n+p,2n+p ³⁾
	External	n+nd	n+nd,n+nd+1 ¹⁾	n+nd+1+p
DROM	RAM	n,n+1 ¹⁾	n, n+1 ¹⁾	n+p
	External	n+nd+1	n+nd	n+nd+1+p
External	RAM	n+nd	n+nd,n+nd+1 ¹⁾	n+nd+1+p
	External	2n+2nd	2n+2nd	2n+2nd+p
MMR	RAM	n	n, n+1 ¹⁾	n+p
	External	n+nd	n+nd	n+nd+1+p
¹⁾ Операнд и код в одном блоке памяти. ²⁾ Два операнда и код в одном блоке памяти. ³⁾ Два операнда в одном блоке памяти				

Класс 15

Одно слово, один цикл. Операнд записи в память данных (Xmem).

Мнемоника

SACCD

SRCCD

STRCD

Циклы

Циклы единичного выполнения			
Операнд	Программа		
Xmem	ROM	RAM	External
RAM	1	1,2*	1+p
External	1+d	1+d	2+d+p
MMR	1	1	1+p
* Операнд и код в одном блоке памяти			

Циклы повторяющихся выполнений			
Операнд	Программа		
Xmem	ROM	RAM	External
RAM	n	n,n+1*	n+p
External	n+nd	n+nd	2n+nd+1+p
MMR	n	n	n+p
* Операнд и код в одном блоке памяти			

Класс 16A

Одно слово, один цикл. Операнд чтения в памяти данных (Smem) или операнд чтения в MMR, и операнд записи в память стека

Мнемоника

PSHD

PSHM

Циклы

Циклы единичного выполнения				
Операнд		Программа		
Xmem	Stack	ROM	RAM	External
RAM	RAM	1	1, 2 ¹⁾ ,3 ²⁾	1+p,2(p=0) ³⁾
	External	1	1, 2 ¹⁾	2+d+p
DROM	RAM	2	1,2 ¹⁾	1+p
	External	2(d=0),1+d(d≠0)	1	2+d+p
External	RAM	1+d	1+d,2+d ¹⁾	2+d+p
	External	2+2d	2+2d	3+2d+p
MMR	RAM	1	1, 2 ¹⁾	1+p
	External	1	1	2+d+p
¹⁾ Операнд и код в одном блоке памяти; ²⁾ Два операнда и код в одном блоке памяти; ³⁾ Два операнда в одном блоке памяти				

Циклы повторяющихся выполнений				
Операнд		Программа		
Xmem	Stack	ROM	RAM	External
RAM	RAM	$n, 2n^{3)}$	$n, n+1^{1,2)}, n+2^{2)}$	$n+p, n+p+1^{3)}$
	External	$n+nd$	$n+nd, n+nd+1^{1)}$	$n+nd+1+p$
DROM	RAM	$n+1$	$n, n+1^{1)}$	$n+p$
	External	$n+nd, n+nd+1^{1)}$	$n+nd$	$n+nd+1+p$
External	RAM	$n+nd$	$n+nd, n+nd+1^{1)}$	$n+nd+1+p$
	External	$2n+2nd$	$2n+2nd$	$2n+2nd+1+p$
MMR	RAM	n	$n, 2n^{1)}$	$n+p$
	External	$n+nd$	$n+nd$	$n+nd+1+p$
¹⁾ Операнд и код в одном блоке памяти; ²⁾ Два операнда и код в одном блоке памяти; ³⁾ Два операнда в одном блоке памяти				

Класс 16В

Два слова, два цикла. Операнд чтения в памяти данных (Smem) использующий косвенную адресацию с длинным смещением и операнд записи в память стека.

Мнемоника

PSHD

Циклы

Циклы единичного выполнения, использующие косвенную адресацию с длинным смещением				
Операнд		Программа		
Smem	Stack	ROM	RAM	External
RAM	RAM	2	2,3*	2+2p
	External	2+d	2+d	3+d+2p
DROM	RAM	3	2,3*	2+2p
	External	$3, 1+d(d \geq 2)$	$2, 1+d(d \geq 1)$	3+d+2p
External	RAM	2+d	2+d	3+d+2p
	External	2+2d	2+2d	4+2d+2p
MMR	RAM	2	2,3*	2+2p
	External	2+d	2+d	3+d+2p
* Операнд и код в одном блоке памяти				

Класс 17А

Одно слово, один цикл. Операнд записи в памяти данных (Smem) или операнд записи в MMR и операнд чтения в памяти стека.

Мнемоника

POPD POPM

Циклы

Циклы единичного выполнения				
Операнд		Программа		
Smem	Stack	ROM	RAM	External
RAM	RAM	1, 2 ¹⁾	1, 2 ¹⁾ , 3 ²⁾	1+p, 2(p=0) ³⁾
	DROM	2	1, 2 ¹⁾	1+p
	External	1+d	1+d, 2+d ¹⁾	2+d+p
	MMR	1	1, 2 ¹⁾	1+p
External	RAM	1+d	1+d	2+d+p
	DROM	1+d(d≠0), 2(d=0)	1+d	2+d+p
	External	2+2d	2+2d	3+2d+p
	MMR	1+d	1+d	2+d+p
¹⁾ Операнд и код в одном блоке памяти; ²⁾ Два операнда и код в одном блоке памяти; ³⁾ Два операнда в одном блоке памяти				

Циклы повторяющихся выполнений				
Операнд		Программа		
Smem	Stack	ROM	RAM	External
RAM	RAM	n, 2n ³⁾	n, n+1 ¹⁾ , 2n ³⁾	n+p, 2n+p ³⁾
	DROM	n+1	n n+1 ¹⁾	n+p
	External	n+nd	n+nd, n+dn+1 ¹⁾	n+nd+1+p
	MMR	n	n, n+1 ¹⁾	n+p
External	RAM	n+nd	n+nd, n+nd+1 ¹⁾	n+nd+1+p
	DROM	n+nd+1	n+nd	n+nd+1+p
	External	2n+2nd	2n+2nd	2n+2nd+1+p
	MMR	n+nd	n+nd	n+nd+1+p
¹⁾ Операнд и код в одном блоке памяти; ²⁾ Два операнда и код в одном блоке памяти; ³⁾ Два операнда в одном блоке памяти				

Класс 17В

Два слова, два цикла. Операнд записи в память данных (Smem) с использованием косвенной адресации с длинным смещением и операнд чтения в памяти стека.

Мнемоника

POPD

Циклы

Циклы единичного выполнения				
Операнд		Программа		
Smem	Stack	ROM	RAM	External
RAM	RAM	2, 3 ¹⁾	2, 3 ²⁾	2+2p
	DROM	3	2, 3 ¹⁾	2+2p
	External	1+d	1+d, 2+d	3+d+2p
	MMR	2	2, 3 ¹⁾	2+2p
External	RAM	2	2, 3 ¹⁾	3+d+2p
	DROM	3	2	3+d+2p
	External	2+2d	2+2d	4+2d+2p
	MMR	2	2	3+d+2p
¹⁾ Операнд и код в одном блоке памяти; ²⁾ Два операнда и код в одном блоке памяти				

Класс 18А

Два слова, два цикла. Операнд чтения и записи в памяти данных (Smem).

Мнемоника

ADDM ANDM ORM XORM

Циклы

Циклы единичного выполнения			
Операнд	Программа		
Smem	ROM	RAM	External
RAM	2	2, 3*	2+2p
External	2+2d	2+2d	4+2d+2p
MMR	2	2	1+p
* Операнд и код в одном блоке памяти			

Класс 18В

Три слова, три цикла. Операнд чтения и записи в памяти данных (Smem) с использованием косвенной адресации с длинным смещением.

Мнемоника

ADDM ANDM ORM XORM

Циклы

Циклы единичного выполнения			
Операнд	Программа		
Smem	ROM	RAM	External
RAM	3	3,4*	3+3p
External	3+d	3+d	5+2d+3p
MMR	3	3	3+3p
* Операнд и код в одном блоке памяти			

Класс 19А

Два слова, два цикла. Операнд чтения в памяти данных (Smem) или операнд чтения в MMR, операнд записи в памяти данных (dmd); или операнд чтения в памяти данных (dmd), и операнд записи в памяти данных (Smem) или операнд записи в MMR.

Мнемоника

MVDK MVDM MVKD MVMD

Циклы

Циклы единичного выполнения				
Операнд		Программа		
Smem	dmd	ROM	RAM	External
RAM	RAM	2, 3 ³⁾	2, 3 ³⁾ , 4 ²⁾	2+2p
	External	2+d	2+d	3+d+2p
	MMR	2	2,3 ¹⁾	2+2p
DROM	RAM	3	2,3 ¹⁾	2+2p
	External	3+d	2+d	3+d+2p
	MMR	3	2	2+2p
External	RAM	2+d	2+d	3+d+2p
	External	2+2d	2+d	4+2d+2p
	MMR	2+d	2+d	3+d+2p
MMR	RAM	2	2, 3 ¹⁾	2+2p
	External	2+d	2+d	3+d+2p
	MMR	2	2	2+2p
¹⁾ Операнд и код в одном блоке памяти; ²⁾ Два операнда и код в одном блоке памяти; ³⁾ Два операнда в одном блоке памяти				

Циклы повторяющихся выполнений				
Операнд		Программа		
Smem	dmad	ROM	RAM	External
RAM	RAM	$n, 2n^{3)}$	$n, n+1^{1)}, 2n^{3)}, 2n+1^{2)}$	$n+1+2p, 2n+1+2p^{3)}$
	External	$n+nd$	$n+nd, n+nd+1^{1)}$	$n+nd+2+2p$
	MMR	n	$n, n+1^{1)}$	$n+1+2p$
DROM	RAM	$n+2$	$n+1$	$n+1+2p$
	External	$n+nd+1$	$n+nd$	$n+nd+2+2p$
	MMR	$n+2$	$n+1$	$n+1+2p$
External	RAM	$n+nd$	$n+nd, n+nd+1^{1)}$	$n+nd+2+2p$
	External	$2n+2nd$	$2n+2nd$	$2n+2nd+2+2p$
	MMR	$n+nd$	$n+nd$	$n+nd+2+2p$
MMR	RAM	n	$n, n+1^{1)}$	$n+1+2p$
	External	$2n+2nd$	$2n+2nd$	$2n+2nd+2+2p$
	MMR	n	n	$n+1+2p$
¹⁾ Операнд и код в одном блоке памяти; ²⁾ Два операнда и код в одном блоке памяти; ³⁾ Два операнда в одном блоке памяти				

Класс 19В

Два слова, два цикла. Операнд чтения в памяти данных (Smem) с использованием косвенной адресации с длинным смещением и операнд записи в память данных (dmad) или операнд чтения в память данных (dmad) и операнд записи в память данных (Smem) с использованием косвенной адресации с длинным смещением.

Мнемоника

MVDK MVKD

Циклы

Циклы единичного выполнения				
Операнд		Программа		
Smem	dmad	ROM	RAM	External
RAM	RAM	$2, 3^{1)}$	$2, 3^{3)}, 4^{2)}$	$2+2p$
	External	$2+d$	$2+d$	$3+d+2p$
	MMR	2	$2, 3^{1)}$	$2+2p$
DROM	RAM	3	3	$2+2p$
	External	$3+d$	$2+d$	$3+d+2p$
	MMR	3	2	$2+2p$
External	RAM	$2+d$	$2+d$	$3+d+2p$
	External	$2+2d$	$2+d$	$4+2d+2p$
	MMR	$2+d$	$2+d$	$3+d+2p$
MMR	RAM	2	$2, 3^{1)}$	$2+2p$
	External	$2+d$	$2+d$	$3+d+2p$
	MMR	2	2	$2+2p$
¹⁾ Операнд и код в одном блоке памяти; ²⁾ Два операнда и код в одном блоке памяти; ³⁾ Два операнда в одном блоке памяти				

Класс 20А

Два слова, четыре цикла. Операнд чтения в памяти данных (Smem) и операнд записи в памяти программ (pmad).

Мнемоника

MVDP

Циклы

Циклы единичного выполнения				
Операнд		Программа		
Smem	pmad	ROM	RAM	External
RAM	RAM	2, 3 ³⁾	2,3 ³⁾ ,4 ²⁾	2+2p,3+2p ³⁾
	External	2+d	2+d,3+d ³⁾	3+d+2p
DROM	RAM	3	2,3 ¹⁾	2+2p
	External	3+d	2+d	3+d+2p
External	RAM	2+d	2+d,3+d ¹⁾	3+d+2p
	External	2+2d	2+2d	3+d+2p
MMR	RAM	2	2	2+2p
	External	2+d	2+d	3+d+2p
¹⁾ Операнд и код в одном блоке памяти; ²⁾ Два операнда и код в одном блоке памяти; ³⁾ Два операнда в одном блоке памяти				

Циклы повторяющихся выполнений				
Операнд		Программа		
Smem	pmad	ROM	RAM	External
RAM	RAM	2+n,3+n ³⁾	2+n,3+n ³⁾ ,4+n ²⁾	n+1+2p,2n+1+2p ³⁾
	External	n+nd+1	n+nd+1,n+nd+2 ¹⁾	n+nd+2+2p
DROM	RAM	n+3	n+3	n+3+2p
	External	n+nd+1	n+nd	n+nd+2+2p
External	RAM	n+nd+1	n+nd	n+nd+2+2p
	External	2n+2nd+1+2p	2n+2nd+1+2p	2n+2nd+2+2p
MMR	RAM	2+n	2+n	n+3+2p
	External	n+nd+1	n+nd+1	n+nd+2+2p
¹⁾ Операнд и код в одном блоке памяти; ²⁾ Два операнда и код в одном блоке памяти; ³⁾ Два операнда в одном блоке памяти				

Класс 20В

Три слова, пять циклов. Операнд чтения в памяти данных (Smem) использующий косвенную адресацию с длинным смещением и операнд записи в памяти программ (pmad).

Мнемоника

MVDP

Циклы

Циклы одиночного выполнения, использующие косвенную адресацию с длинным смещением				
Операнд		Программа		
Smem	pmad	ROM	RAM	External
RAM	RAM	3, 4 ³⁾	3,4 ³⁾ ,5 ²⁾	3+3p,4+3p ³⁾
	External	3+d	3+d,4+d ³⁾	4+d+3p
DROM	RAM	4	3,4 ¹⁾	3+3p
	External	4+d	3+d	4+d+3p
External	RAM	3+d	3+d,4+d ¹⁾	4+d+3p
	External	3+2d	3+2d	5+2d+3p
MMR	RAM	3	3,4 ²⁾	3+3p
	External	3+d	3+d	3+d+3p
¹⁾ Операнд и код в одном блоке памяти; ²⁾ Два операнда и код в одном блоке памяти; ³⁾ Два операнда в одном блоке памяти				

Класс 21А

Два слова, три цикла. Операнд чтения в памяти программ (pmad) и операнд записи в памяти данных (Smem).

Мнемоника

MVPD

Циклы

Циклы единичного выполнения				
Операнд		Программа		
pmad	Smem	ROM	RAM	External
RAM	RAM	2,3 ³⁾	2,3 ³⁾ ,4 ²⁾	2+2p,3+2p ³⁾
	External	2+d	2+d,3+d ²⁾	3+d+2p
	MMR	2	2,3 ¹⁾	2+2p
PROM	RAM	3	2,3 ¹⁾	2+2p
	External	3+d	2+d	3+d+2p
	MMR	3	2	2+2p
External	RAM	2+d	2+d,3+d ¹⁾	3+d+2p
	External	2+2d	2+2d	4+2d+2p
	MMR	2+d	2+d	3+d+2p
¹⁾ Операнд и код в одном блоке памяти; ²⁾ Два операнда и код в одном блоке памяти; ³⁾ Два операнда в одном блоке памяти				

Циклы повторяющихся выполнений				
Операнд		Программа		
pma _d	Smem	ROM	RAM	External
RAM	RAM	n+2	n+2,2n ³⁾	n+2+2p,2n+2p ³⁾
	External	n+nd	n+nd	n+nd+2+2p
	MMR	n+2	n+2	n+2+2p
PROM	RAM	n+2	n+2	3+2p
	External	n+nd+2	n+nd+1	n+nd+2+2p
	MMR	n+2	n+1	n+2+2p
External	RAM	n+nd	n+nd	n+nd+2+2p
	External	2n+2nd+1	2n+2nd+1	2n+2nd+2+2p
	MMR	n+nd	n+nd	n+nd+2+2p
¹⁾ Операнд и код в одном блоке памяти; ²⁾ Два операнда и код в одном блоке памяти; ³⁾ Два операнда в одном блоке памяти				

Класс 21В

Три слова, четыре цикла. Операнд чтения в памяти программ (pma_d) и операнд записи в памяти данных (Smem), использующий максимально удаленную косвенную адресацию.

Мнемоника

MVPD

Циклы

Циклы единичного выполнения, использующие косвенную адресацию с длинным смещением				
Операнд		Программа		
pma _d	Smem	ROM	RAM	External
RAM	RAM	3,4 ³⁾	3,4 ³⁾ ,5 ²⁾	3+3p,4+3p ³⁾
	External	3+d	3+d,4+d ²⁾	4+d+3p
	MMR	3	3,4 ¹⁾	3+3p
PROM	RAM	4	3,4 ¹⁾	3+3p
	External	4+d	3+d	4+d+3p
	MMR	4	3	3+3p
External	RAM	3+d	3+d,4+d ¹⁾	4+d+3p
	External	3+3d	3+3d	5+2d+3p
	MMR	3+d	3+d	3+d+3p
¹⁾ Операнд и код в одном блоке памяти; ²⁾ Два операнда и код в одном блоке памяти; ³⁾ Два операнда в одном блоке памяти				

Класс 22А

Два слова, три цикла. Операнд чтения в памяти данных (Smem) и операнд чтения в памяти программ (pmad).

Мнемоника

МАСР

Циклы

Циклы единичного выполнения				
Операнд		Программа		
pmad	Smem	ROM	RAM	External
RAM	RAM	2,3 ¹⁾	2,3 ¹⁾ ,4 ²⁾	2+2p,3+2p ³⁾
	External	2+d	2+d,3+d ¹⁾	3+d+2p
	MMR	2	2,3 ¹⁾	2+2p
PROM	RAM	3	3, 4 ¹⁾	2+2p
	External	3+d	3+d	3+d+2p
	MMR	3	3	2+2p
External	RAM	2+d	2+d, 3+d ¹⁾	3+d+2p
	External	3+2d	3+2d	4+2d+2p
	MMR	2+d	2+d	3+d+2p

¹⁾ Операнд и код в одном блоке памяти;
²⁾ Два операнда и код в одном блоке памяти;
³⁾ Два операнда в одном блоке памяти

Циклы повторяющихся выполнений				
Операнд		Программа		
pmad	Smem	ROM	RAM	External
RAM	RAM	n+2,2n+2 ³⁾	n+2, n+3 ¹⁾ ,2n+2 ³⁾	n+2+2p,2n+2+2p ³⁾
	External	n+nd+2	n+nd+2	n+nd+2+2p
	MMR	n+2	n+2,n+3 ¹⁾	n+2+2p
PROM	RAM	n+3	n+3, n+4 ¹⁾ ,	n+2+2p
	External	n+nd+3	n+nd+3	n+nd+2+2p
	MMR	n+3	n+3	n+2+2p
External	RAM	n+nd+2	n+nd+2,n+nd+3 ¹⁾	n+nd+2+2p
	External	2n+2nd+2	2n+2nd+2	2n+2nd+2+2p
	MMR	n+nd+2	n+nd+2	n+nd+2+2p

¹⁾ Операнд и код в одном блоке памяти;
³⁾ Два операнда в одном блоке памяти.
³⁾ Два операнда в одном блоке памяти

Класс 22В

Три слова, четыре цикла. Операнд чтения в памяти данных (Smem), использующий максимально удаленную косвенную адресацию и операнд чтения в памяти программ (pmad).

Мнемоника

MACP

Циклы

Циклы единичного выполнения, использующие косвенную адресацию с длинным смещением				
Операнд		Программа		
pma	Smem	ROM	RAM	External
RAM	RAM	3,4 ¹⁾	3,4 ¹⁾ ,5 ²⁾	3+3p,4+3p ³⁾
	External	3+d	3+d,4+d ¹⁾	4+d+3p
	MMR	3	3,4 ¹⁾	3+3p
PROM	RAM	4	4, 5 ¹⁾	4+3p
	External	4+d	4+d	5+d+3p
	MMR	4	4	4+3p
External	RAM	3+d	3+d, 4+d ¹⁾	4+d+3p
	External	4+2d	4+2d	5+2d+3p
	MMR	3+d	3+d	4+d+3p
¹⁾ Операнд и код в одном блоке памяти. ²⁾ Два операнда и код в одном блоке памяти; ³⁾ Два операнда в одном блоке памяти				

Класс 23А

Два слова, три цикла. Операнд чтения в памяти данных (Smem), операнд записи в памяти данных (Smem), и операнд чтения в памяти программ (pma).

Мнемоника

MACD

Циклы

Циклы единичного выполнения				
Операнд		Программа		
pma	Smem	ROM	RAM	External
RAM	RAM	2,3 ³⁾	2,3 ³⁾ ,4 ²⁾	2+2p,3+2p ³⁾
	External	2+d	2+d,3+d ²⁾	3+d+2p
	MMR	2	2,3 ¹⁾	2+2p
PROM	RAM	3	2,3 ¹⁾	2+2p
	External	3+d	2+d	3+d+2p
	MMR	3	2	2+2p
External	RAM	2+d	2+d,3+d ¹⁾	3+d+2p
	External	2+2d	2+2d	4+2d+2p
	MMR	2+d	2+d	3+d+2p
¹⁾ Операнд и код в одном блоке памяти; ²⁾ Два операнда и код в одном блоке памяти; ³⁾ Два операнда в одном блоке памяти				

Циклы повторяющихся выполнений				
Операнд		Программа		
pmad	Smem	ROM	RAM	External
RAM	RAM	n+2	n+2,2n ²⁾	n+2+2p,2n+2p ²⁾
	External	n+nd	n+nd	n+nd+2+2p
	MMR	n+2	n+2	n+2+2p
PROM	RAM	n+2	n+2	3+2p
	External	n+nd+2	n+nd+1	n+nd+2+2p
	MMR	n+2	n+1	n+2+2p
External	RAM	n+nd	n+nd	n+nd+2+2p
	External	2n+2nd+1	2n+2nd+1	2n+2nd+2+2p
	MMR	n+nd	n+nd	n+nd+2+2p
¹⁾ Операнд и код в одном блоке памяти; ²⁾ Два операнда в одном блоке памяти				

Класс 23В

Три слова, четыре цикла. Операнд чтения в памяти данных (Smem), использующий косвенную адресацию с длинным смещением, операнд записи в памяти данных (Smem), использующий косвенную адресацию с длинным смещением, и операнд чтения из памяти программ (pmad).

Мнемоника

MACD

Циклы

Циклы единичного выполнения, использующие косвенную адресацию с длинным смещением				
Операнд		Программа		
pmad	Smem	ROM	RAM	External
RAM	RAM	3,4 ³⁾	3,4 ³⁾ ,5 ²⁾	3+3p,4+3p ³⁾
	External	3+d	3+d,4+d ²⁾	4+d+3p
	MMR	3	3,4 ¹⁾	3+3p
PROM	RAM	4	3,4 ¹⁾	3+3p
	External	4+d	3+d	4+d+3p
	MMR	4	3	3+3p
External	RAM	3+d	3+d,4+d ¹⁾	4+d+3p
	External	3+3d	3+3d	5+2d+3p
	MMR	3+d	3+d	3+d+3p
¹⁾ Операнд и код в одном блоке памяти; ²⁾ Два операнда и код в одном блоке памяти; ³⁾ Два операнда в одном блоке памяти				

Класс 24А

Одно слово, один цикл. Операнд чтения в памяти данных (Smem) и операнд записи в памяти данных (Smem).

Мнемоника

DELAY LTD

Циклы

Циклы единичного выполнения			
Операнд	Программа		
Smem	ROM	RAM	External
RAM	1	1,2*	1+p
External	2+2d	2+2d	3+p+2d
* Операнд и код в одном блоке памяти			

Циклы повторяющихся выполнений			
Операнд	Программа		
Smem	ROM	RAM	External
RAM	n	n,n+1*	n+p
External	2n+2nd	2n+2nd	2n+2nd+2+p
* Операнд и код в одном блоке памяти			

Класс 24В

Два слова, два цикла. Операнд чтения в памяти данных (Smem), использующий косвенную адресацию с длинным смещением и операнд записи в памяти данных (Smem), использующий косвенную адресацию с длинным смещением.

Мнемоника

DELAY LTD

Циклы

Циклы единичного выполнения, использующие косвенную адресацию с длинным смещением			
Операнд	Программа		
Smem	ROM	RAM	External
RAM	2	2,3*	2+2p
External	3+2d	3+2d	4+2p+2d
* Операнд и код в одном блоке памяти			

Класс 25А

Одно слово, пять циклов. Адрес чтения в памяти программ (pmad адресуемый аккумулятором А) и операнд записи в памяти данных (Smem).

Мнемоника

READA

Циклы

Циклы единичного выполнения				
Операнд		Программа		
pmad	Smem	ROM	RAM	External
RAM	RAM	1	1,2 ¹⁾	1+p, 2+p ²⁾
	External	1+d	1+d	2+d+p
	MMR	1	1	1+p
PROM	RAM	2	1, 2 ¹⁾	1+p
	External	2	1+d	2+d+p
	MMR	2	1	1+p
External	RAM	1+d	1+d,2+d ¹⁾	2+d+p
	External	2+2d	2+2d	3+2d+p
	MMR	1+d	1+d	2+d+p
¹⁾ Операнд и код в одном блоке памяти; ²⁾ Два операнда в одном блоке памяти				

Циклы повторяющихся выполнений				
Операнд		Программа		
pmad	Smem	ROM	RAM	External
RAM	RAM	n	n,n+1 ¹⁾	n+p, 2n+p ³⁾
	External	n+nd	n+nd	n+nd+1+p
	MMR	n	n	n+1+p
PROM	RAM	n+1	n+1, n+2 ¹⁾	n+p
	External	n+nd+2	n+nd	n+nd+p
	MMR	n+1	n	n+p
External	RAM	n+nd	n+nd,n+nd+1 ¹⁾	2n+nd+p
	External	2n+2nd	2n+2nd	2n+2nd+1+p
	MMR	n+nd	n+nd	n+nd+1+p
¹⁾ Операнд и код в одном блоке памяти; ²⁾ Два операнда в одном блоке памяти				

Класс 25В

Два слова, шесть циклов. Адрес чтения в памяти программ (p_{mad}) операнд записи в памяти данных (S_{mem}) с использованием косвенной адресации с длинным смещением.

Мнемоника

READA

Циклы

Циклы единичного выполнения с косвенной адресацией и длинным смещением				
Операнд		Программа		
p _{mad}	S _{mem}	ROM	RAM	External
RAM	RAM	2	2,3 ¹⁾	2+2p, 3+2p ²⁾
	External	2+d	2+d	3+d+2p
	MMR	2	2	2+2p
PROM	RAM	3	2, 3 ¹⁾	2+2p
	External	3	2+d	3+d+2p
	MMR	3	2	2+2p
External	RAM	2+d	2+d,3+d ¹⁾	3+d+2p
	External	3+2d	3+2d	4+2d+2p
	MMR	2+d	2+d	3+d+2p
¹⁾ Операнд и код в одном блоке памяти; ²⁾ Два операнда в одном блоке памяти				

Класс 26А

Одно слово, пять циклов. Операнд чтения в памяти данных (S_{mem}) адрес записи в памяти программ (p_{mad}).

Мнемоника

WRITA

Циклы

Циклы единичного выполнения				
Операнд		Программа		
S _{mem}	p _{mad}	ROM	RAM	External
RAM	RAM	1	1,2 ¹⁾ ,3 ²⁾	1+p, 2+p ³⁾
	External	1+d	1+d	2+d+p
DROM	RAM	2	1, 2 ¹⁾	1+p
	External	2+d	1+d	2+d+p
External	RAM	1+d	1+d,2+d ¹⁾	2+d+p
	External	2+2d	2+2d	3+2d+p
MMR	RAM	1	1,2 ¹⁾	1+p, 2+p ³⁾
	External	1+d	1+d	2+d+p
¹⁾ Операнд и код в одном блоке памяти; ²⁾ Два операнда и код в одном блоке памяти; ³⁾ Два операнда в одном блоке памяти				

Циклы повторяющихся выполнений				
Операнд		Программа		
Smem	pmad	ROM	RAM	External
RAM	RAM	n	$n, 2n^{1)}, 2n+1^{2)}$	$n+p, 2n+p^{3)}$
	External	$n+nd$	$n+nd$	$n+nd+1+p$
DROM	RAM	$n+1$	$n, n+1^{1)}$	$n+p$
	External	$n+1$	$n+nd$	$n+nd+1+p$
External	RAM	$n+nd$	$n+nd, n+1+d^{1)}$	$n+nd+1+p$
	External	$2n+2nd$	$2n+2nd$	$2n+2nd+1+p$
MMR	RAM	n	$n, n+1^{1)}$	$n+p, n+1+p^{3)}$
	External	$n+nd$	$n+nd$	$n+nd+1+p$
¹⁾ Операнд и код в одном блоке памяти; ²⁾ Два операнда и код в одном блоке памяти; ³⁾ Два операнда в одном блоке памяти				

Класс 26В

Два слова, шесть циклов. Операнд чтения в памяти данных (Smem) использующий косвенную адресацию с длинным смещением и адрес записи в памяти программ (pmad).

Мнемоника

WRITA

Циклы

Циклы единичного выполнения с максимально удаленным модификатором				
Операнд		Программа		
Smem	pmad	ROM	RAM	External
RAM	RAM	2	$2, 3^{1)}, 4^{2)}$	$2+2p, 3+2p^{3)}$
	External	$2+d$	$2+d$	$3+d+2p$
DROM	RAM	3	$2, 3^{1)}$	$2+2p$
	External	$3+d$	$2+d$	$3+d+2p$
External	RAM	$2+d$	$2+d, 3+d^{1)}$	$3+d+2p$
	External	$3+2d$	$3+2d$	$4+2d+2p$
MMR	RAM	2	$2, 3^{1)}$	$2+2p, 3+2p^{3)}$
	External	$2+d$	$2+d$	$3+d+2p$
¹⁾ Операнд и код в одном блоке памяти; ²⁾ Два операнда и код в одном блоке памяти; ³⁾ Два операнда в одном блоке памяти				

Класс 27А

Два слова, два цикла. Операнд чтения порта ввода-вывода и операнд записи в память данных (Smem).

Мнемоника

PORTR

Циклы

Циклы единичного выполнения				
Операнд		Программа		
Port	Smem	ROM	RAM	External
External	RAM	2+io	3+io, 4+io*	3+2p+io
	External	2+d+io	3+d+io	4+2p+d+io
* Операнд и код в одном блоке памяти				

Циклы повторяющихся выполнений				
Операнд		Программа		
Port	Smem	ROM	RAM	External
External	RAM	n+nio+1	n+nio+1, n+nio+2*	n+nio+2+2p
	External	2n+nd+nio+1	2n+nd+nio+1	2n+nio+nd+2+2p
* Операнд и код в одном блоке памяти				

Класс 27В

Три слова, три цикла. Операнд чтения порта ввода-вывода и операнд записи в память данных (Smem) с использованием косвенной адресации с длинным смещением.

Мнемоника

PORTR

Циклы

Циклы единичного выполнения с максимально удаленным модификатором				
Операнд		Программа		
Port	Smem	ROM	RAM	External
External	RAM	3+io	4+io, 5+io*	4+3p+io
	External	3+d+io	4+d+io	5+3p+d+io
* Операнд и код в одном блоке памяти				

Класс 28А

Два слова, два цикла. Операнд чтения в памяти данных (Smem) и операнд записи в порт ввода-вывода.

Мнемоника

PORTW

Циклы

Циклы единичного выполнения				
Операнд		Программа		
Port	Smem	ROM	RAM	External
External	RAM	2+io	3+io,4+io*	3+io+2p
	DROM	3+io	2+io	3+io+2p
	External	2+d+io	3+d+io	4+io+2p+d
* Операнд и код в одном блоке памяти				

Циклы повторяющихся выполнений				
Операнд		Программа		
Port	Smem	ROM	RAM	External
External	RAM	n+nio+1	n+nio+1,n+nio+2*	n+nio+2+2p
	DROM	n+nio+2	n+nio+1	n+nio+2+2p
	External	2n+nd+nio+1	2n+nd+nio+1	2n+nio+nd+2+2p
* Операнд и код в одном блоке памяти				

Класс 28В

Три слова, три цикла. Операнд чтения в памяти данных (Smem) с использованием косвенной адресации с длинным смещением и операнд записи в порт ввода-вывода.

Мнемоника

PORTW

Циклы

Циклы единичного выполнения с максимально удаленным модификатором				
Операнд		Программа		
Port	Smem	ROM	RAM	External
External	RAM	3+io	4+io,5+io*	4+3p+io
	DROM	4+io	4+io	4+3p+io
	External	3+d+io	4+d+io	5+3p+d+io
* Операнд и код в одном блоке памяти				

Класс 29А

Два слова, четыре цикла, два цикла (с задержкой), два цикла (ложное условие).
Операнд в памяти программ (pmad).

Мнемоника

B[D] BANZ[D] FB[D] (в данной модели не реализовано) RPTB[D]

Циклы

Циклы единичного выполнения		
Программа		
ROM	RAM	External
4	4	4+4p

Циклы единичного выполнения с задержкой		
Программа		
ROM	RAM	External
2	2	2+2p

Класс 29В

Два слова, четыре цикла, два цикла (с задержкой). Операнд в памяти программ (pmad).

Мнемоника

CALL[D] FCALL[D] (в данной модели не реализовано)

Циклы

Циклы единичного выполнения			
Операнд	Программа		
Stack	ROM	RAM	External
RAM	4	4,5*	4+4p
External	4+d	4+d	5+4p+d
* Операнд и код в одном блоке памяти			

Циклы единичного выполнения с задержкой			
Операнд	Программа		
Stack	ROM	RAM	External
RAM	4	4,5*	4+4p
External	4+d	4+d	5+4p+d
* Операнд и код в одном блоке памяти			

Класс 30А

Одно слово, шесть циклов, четыре цикла (с задержкой). Операнд регистра.

Мнемоника

ВАСС[D] FBАСС[D] (в данной модели не реализована)

Циклы

Циклы единичного выполнения		
Программа		
ROM	RAM	External
4	4	4+3p

Циклы единичного выполнения с задержкой		
Программа		
ROM	RAM	External
4	4	4+3p

Класс 30В

Одно слово, шесть циклов, четыре цикла (с задержкой). Операнд регистра.

Мнемоника

CALA[D] FCALA[D] (в данной модели не реализовано)

Циклы

Циклы единичного выполнения			
Операнд	Программа		
Stack	ROM	RAM	External
RAM	6	6	6+3p
External	6	6	7+3p+d

Циклы единичного выполнения с задержкой			
Операнд	Программа		
Stack	ROM	RAM	External
RAM	4	4	4+p
External	4	4	5+p+d

Класс 31А

Два слова, пять цикла, три цикла (с задержкой). Операнд в памяти программ (pmad) и операнды короткого непосредственного значения

Мнемоника

BC[D]

Циклы

Циклы единичного выполнения			
Условие	Программа		
	ROM	RAM	External
Истина	5	5	5+4p
Ложь	3	3	3+2p

Циклы единичного выполнения с задержкой			
Условие	Программа		
	ROM	RAM	External
Истина	3	3	3+2p
Ложь	3	3	3+2p

Класс 31В

Два слова, пять циклов, три цикла (с задержкой), три цикла (ложное условие).
Операнд в памяти программ (pmad) и операнды с коротким непосредственным значением.

Мнемоника

СС[D]

Циклы

Циклы единичного выполнения (условие истинное)			
Операнд	Программа		
	ROM	RAM	External
Stack			
RAM	5	5,6*	5+4p
External	5	5	8+4p+d
* Операнд и код в одном блоке памяти			

Циклы единичного выполнения (условие ложное)			
Операнд	Программа		
	ROM	RAM	External
Stack			
RAM	3	3,4*	3+2p
External	3	3	6+2p+d
* Операнд и код в одном блоке памяти			

Циклы единичного выполнения (условие ложное)			
Операнд		Программа	
		ROM	RAM
Stack			
RAM		3	3,4*
External		3	3
* Операнд и код в одном блоке памяти			

Класс 32

Одно слово, пять циклов, три цикла (с задержкой), три цикла (ложное условие).
Операнд отсутствует, или операнд с коротким непосредственным значением.

Мнемоника

RC[D] RET[D] RETE[D]

Циклы

Циклы единичного выполнения			
Операнд	Программа		
Stack	ROM	RAM	External
RAM	5	5,6*	5+3p
External	5+d	5+d	6+d+3p
* Операнд и код в одном блоке памяти			

Циклы единичного выполнения с задержкой			
Операнд	Программа		
Stack	ROM	RAM	External
RAM	3	3,4*	3+p
External	3+d	3+d	4+d+p
* Операнд и код в одном блоке памяти			

Класс 33

Одно слово, три цикла, один цикл (с задержкой). Операнд отсутствует.

Мнемоника

RETF[D]

Циклы

Циклы единичного выполнения		
Программа		
ROM	RAM	External
3	3	3+p

Циклы единичного выполнения с задержкой		
Программа		
ROM	RAM	External
1	1	1+p

Класс 34

Одно слово, шесть циклов, четыре цикла (с задержкой). Операнд отсутствует.

Мнемоника

FRET[D] FRETE[D]

Циклы

В данной модели операции не реализованы.

Класс 35

Одно слово, три цикла. Операнд отсутствует или операнд с коротким непосредственным значением.

Мнемоника

INTR RESET TRAP

Циклы

Циклы единичного выполнения		
Программа		
ROM	RAM	External
3	3	3+p

Класс 36

Одно слово, четыре цикла (минимум). Операнд с коротким непосредственным значением.

Мнемоника

IDLE

Циклы

Количество циклов, необходимое для выполнения этой команды, зависит от периода бездействия

20.3 Команды на языке ассемблера

Этот раздел содержит подробную информацию о наборе команд для ядра DSP микросхемы. Набор команд включает команды обработки сигналов, требующие большого объема вычислений и команды общего назначения, такие как мультиобработка и высокоскоростное управление.

20.3.1 Обозначения и сокращения

В разделе перечислены сокращения и обозначения, используемые в обзоре набора команд и в описании отдельных команд.

Таблица 20-29 – Обозначения и сокращения в наборе команд

Обозначение	Содержание
A	Аккумулятор A
ALU	Арифметико-логическое устройство (АЛУ)
AR	Вспомогательный регистр общего пользования
ARx	Определяет указанный вспомогательный регистра ($0 \leq x \leq 7$)
ARP	Указатель вспомогательного регистра (трехбитное поле в ST0), указывает текущий AR.
ASM	Пятиразрядное поле режима сдвига аккумулятора в ST1 ($-16 \leq ASM \leq 15$)
B	Аккумулятор B
BRAF	Флаг активности повторения блока команд в ST1
BRC	Счетчик повторения блока программного кода
BITC	Четырехразрядное значение, определяющее бит ячейки памяти данных, тестируемый командой контроля битов ($0 \leq BITC \leq 15$)
C16	16-разрядный бит режима двухсловной/двойной точности арифметики в ST1
C	Бит переноса в ST0
CC	Двухразрядный код условия ($0 \leq CC \leq 3$)
CMPT	Бит режима совместимости в ST1
CPL	Бит режима трансляции в ST1
cond	Операнд, представляющий условие, используемое условными командами
[D]	Режим задержки
DAB	Адресная шина D
DAR	Адресный регистр DAB
dmad	Непосредственный 16-разрядный адрес памяти данных ($0 \leq dmad \leq 65\,535$)
Dmem	Операнд в памяти данных
DP	Десятиразрядное поле указателя страницы памяти данных в ST0 ($0 \leq DP \leq 511$)
dst	Аккумулятор приемник (A или B)
dst_	Противоположный аккумулятор: если dst = A, то dst_ = B; если dst = B, то dst_ = A
EAB	Адресная шина E
EAR	Адресный регистр EAB

Обозначение	Содержание
extpmad	Непосредственный 23-разрядный адрес памяти программ
FRCT	Бит режима дробных чисел ST1
hi(A)	Старшая часть аккумулятора A (биты 31–16)
HM	Бит режима захвата (HOLD режима) в ST1
IFR	Регистр флага прерывания
INTM	Бит режима прерывания в ST1
K	Короткое непосредственное значение меньше, чем 9 бит
k3	Трехразрядное непосредственное значение ($0 \leq k3 \leq 7$)
k5	Пятиразрядное непосредственное значение ($-16 \leq k5 \leq 15$)
k9	Девятиразрядное непосредственное значение ($0 \leq k9 \leq 511$)
lk	16-битное длинное непосредственное значение
Lmem	32-разрядный операнд памяти данных, использующий адресацию длинного слова
mnr, MMR	Регистр, отображаемый на память
MMRx,	Регистр, отображаемый на память, AR0–AR7 или SP
MMRy	
n	Число слов, полученных в результате выполнения команды XC; $n = 1$ или 2
N	Определяет регистр состояния, изменяемый в командах RSBX, SSBX, и XC: N = 0 статусный регистр ST0; N = 1 статусный регистр ST1
OVA	Флаг переполнения аккумулятора A в ST0
OVb	Флаг переполнения аккумулятора B в ST0
OVdst	Флаг переполнения аккумулятора приемника (A или B)
OVdst_	Флаг переполнения аккумулятора, противоположного приемнику (A или B)
OVsrc	Флаг переполнения аккумулятора источника (A или B)
OVM	Бит режима переполнения в ST1
PA	непосредственный 16-разрядный адрес порт ($0 \leq PA \leq 65\,535$)
PAR	Регистр адреса программы
PC	Счетчик программ
pmad	Непосредственный 16-разрядный адрес памяти программ ($0 \leq pmad \leq 65\,535$)
Pmem	Операнд в памяти программ
PMST	Регистр состояния режима процессора
prog	Операнд в памяти программ
[R]	Режим округления
RC	Счетчик повторения
REA	Регистр конечного адреса повторяемого блока
rnd	округление
RSA	Регистр начального адреса повторяемого блока
RTN	Регистр быстрого возврата, используемый в команде RETF[D]
SBIT	Четырехбитное значение, которое определяет номер разряда регистра состояния, изменяемого в командах RSBX, SSBX, и XC ($0 \leq SBIT \leq 15$)
SHFT	Четырехразрядное значение сдвига ($0 \leq SHFT \leq 15$)
SHIFT	Пятиразрядное значение сдвига ($0 \leq SHIFT \leq 15$)
Sind	Операнд в памяти данных, использующий косвенную адресацию
Smem	16-разрядный операнд в памяти данных

Обозначение	Содержание
SP	Указатель стека
src	Аккумулятор источник (A or B)
ST0, ST1	Статусный регистр 0, статусный регистр 1
SXM	Бит режима расширения знака в ST1
T	Временный регистр
TC	Флаг проверки/управления в ST0
TOS	Вершина стека
TRN	Регистр перехода
TS	Значение сдвига, указанное битами 5-0 регистра T ($-16 \leq TS \leq 31$)
uns	Беззнаковое значение
XF	Бит внешнего флага в ST1
XPC	Регистр расширения программного счетчика
Xmem	16-битный операнд в расслоенной памяти, используемый в двухоперандных командах и некоторых однооперандных командах
Ymem	16-битный операнд в расслоенной памяти, используемый в двухоперандных командах
-- SP	Значение указателя стека уменьшается на 1
++ SP	Значение указателя стека увеличивается на 1
++ PC	Значение программного счетчика увеличивается на 1

Таблица 20-30 – Обозначения и сокращения в коде операции

Символ	Значение
A	Бит адреса памяти данных
ARX	Трехразрядное значение, которое определяет вспомогательный регистр
BITC	Четырехразрядный код номера разряда
CC	Двухразрядный код состояния
CCCC CCCC	Восьмиразрядный код состояния
COND	Четырехразрядный код состояния
D	Бит аккумулятора приемника (dst): D = 0 аккумулятор A; D = 1 аккумулятор B
I	Бит режима адресации: I = 0 режим прямой адресации; I = 1 режим косвенной адресации
K	Короткое непосредственное значение меньше чем 9 бит
MMRX	Четырехбитное значение, определяющее один из девяти отображаемых на память регистров ($0 \leq MMRX \leq 8$)
MMRY	Четырехбитное значение, определяющее один из девяти отображаемых на память регистров ($0 \leq MMRY \leq 8$)
N	Одиночный бит
NN	Двухбитное значение, определяющее тип прерывания
R	Бит выбора округления (rnd): R = 0 команда выполняется без округления; R = 1 округление результата

Символ	Значение
S	Бит аккумулятора источника (src): S = 0 аккумулятор A; S = 1 аккумулятор B
SBIT	Четырехразрядный номер разряда статусного регистра
SHFT	Четырехразрядное значение сдвига ($0 \leq \text{SHFT} \leq 15$)
SHIFT	Пятиразрядное значение сдвига ($-16 \leq \text{SHIFT} \leq 15$)
X	Бит памяти данных
Y	Бит памяти данных
Z	Бит задержки команды: Z = 0 команда выполняется без задержки; Z = 1 команда выполняется с задержкой

Таблица 20-31 – Система обозначений набора команд

Символ	Значение
Выделение жирным шрифтом	Выделенные жирным шрифтом буквы в синтаксисе команды должны быть записаны точно в таком же виде, например: в синтаксисе ADD Xmem, Ymem, dst, можно использовать любые значения Xmem и Ymem, но слово ADD должно быть напечатано так, как показано
курсив	Символы, выделенные в синтаксисе курсивом, являются переменными. Например: в синтаксисе ADD Xmem, Ymem, dst, можно использовать любые значения для Xmem и Ymem
[x]	Операнд, взятый в квадратные скобки является необязательным. Например: в синтаксисе ADD Smem [, SHIFT], src [, dst], необходимо использовать значения для Smem и src; в то время как SHIFT и dst являются необязательными
#	Префикс константы, используемый в непосредственной адресации. Для коротких или длинных непосредственных операндов # используется в командах где присутствует неоднозначность по отношению к другим режимам адресации, использующим непосредственный операнд. Например: RPT #15 короткую непосредственную адресацию, что вызывает 16-тикратное повторение следующей команды. RPT 15 использует прямую адресацию. Количество повторов следующей команды определяется значением, хранимым в памяти. Для команд использующих непосредственные операнды, для которых нет неоднозначности, # принимается ассемблером. Например, RPTZ A, #15 и RPTZ A, 15 эквивалентны
(abc)	Содержание регистра или расположение abc. Например : (src) обозначает содержание аккумулятора источника
$x \rightarrow y$	Значение x присваивается регистру или ячейке y. Например: (Smem) $\rightarrow$ dst обозначает, что содержанием ячейки памяти данных загружается аккумулятор приемник
r(n-m)	Биты с n по m в регистре или ячейке r . Например: src(15-0) означает биты аккумулятора источника с 15 по 0
$\ll nn$	Сдвиг влево битов nn (отрицательный или положительный)
	Параллельная команда

Символ	Значение
\\	Циклически сдвинуть влево
//	Циклически сдвинуть вправо
– x	Логическая инверсия x (в двоичной системе)
x	Абсолютное значение x
AAh	Показывает, что AA представляет шестнадцатеричное число

Таблица 20-32 – Операторы, используемые в наборе команд

Символ	Операторы	Вычисление
+ – ~	унарный плюс, минус, обратный код	справа налево
* / %	умножение, деление, число по модулю	слева направо
+ –	сложение, вычитание	слева направо
<< >>	сдвиг влево, сдвиг вправо	слева направо
<<<	логический сдвиг влево	слева направо
< ≤	меньше чем, меньше чем или равно	слева направо
> ≥	больше чем, больше чем или равно	слева направо
≠ !=	не равно	слева направо
&	побитовое «И» (AND)	слева направо
^	побитовое исключение ИЛИ (XOR)	слева направо
	побитовое «ИЛИ» (OR)	слева направо

20.3.1.1 Структура описания команд

Данный типичный пример описания команды приводится для того, чтобы ознакомить с форматом описания команды и объяснить, что описывается каждым заголовком. Каждое описание команды представляет следующую информацию:

- 1 Синтаксис ассемблера;
- 2 Операнды;
- 3 Код операции;
- 4 Выполнение;
- 5 Статусные биты (признаки);
- 6 Описание;
- 7 Слова;
- 8 Циклы;
- 9 Классы;
- 10 Примеры.

Каждое описание команды начинается с записи на языке ассемблера. Метки могут размещаться как перед командой в той же строке, так и на предыдущей строке в первой колонке. Дополнительное поле для комментариев может содержать запись на ассемблере. Необходимо оставлять пространство между полями:

- 1 Метка;
- 2 Команда и операнды;
- 3 Комментарий.

Синтаксис	1: EXAMPLE Smem, src 2: EXAMPLE Smem, TS, src 3: EXAMPLE Smem, 16, src [, dst] 4: EXAMPLE Smem [, SHIFT], src [, dst] Каждое описание команды начинается с выражения на языке ассемблера. См. раздел о значении символов синтаксиса.																																
Операнды	Smem: Единичный операнд памяти данных Xmem, Ymem: Двойные операнды памяти данных src, dst: A (аккумулятор A) B (аккумулятор B) $-16 \leq \text{SHIFT} \leq 15$ Операнды могут быть константами или выражениями, возникающими во время ассемблирования. Это относится к памяти, порту ввода/вывода, адресам регистров, указателям и другим константам. В этом разделе приводится список допустимых значений для разных типов операндов.																																
Код операции	<table><tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td></tr></table> Код операции разделяет различные поля битов для формирования каждой из команд. См. раздел об определении символы кода операции.	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																		
x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x																		
Выполнение	1: (Smem) + (src) → src 2: (Smem) << (TS) + (src) → src 3: (Smem) << 16 + (src) → dst 4: (Smem) [<< SHIFT] + (src) → dst Раздел выполнения описывает процессы, которые происходят во время выполнения команды. Примеры выполнений пронумерованы в соответствии с нумерацией синтаксиса. См. раздел определения символов выполнения.																																
Статусные биты	Выполнение команды может зависеть от полей в статусном регистре. Также состояние и самих полей статусного регистра может измениться. Оба эффекта описываются в данном разделе.																																
Описание	Этот раздел описывает выполнение команды и ее влияние на процессор и содержимое памяти. Обсуждаются все ограничения операндов, вызванные процессором или ассемблером. Информационная поддержка символически представлена в данном разделе.																																
Слова	Это поле уточняет количество слов памяти, требуемое для хранения команд и слов расширения. Для команд, работающих в режиме одиночной адресации, количество слов представлено для всех модификаций, кроме тех, которые с длинным смещением и требуют одно дополнительное слово.																																
Циклы	Это поле уточняет количество циклов, требуемых для выполнения данной команды в процессоре один раз с доступом данных в DARAM доступом программ из ROM. Дополнительное уточнение количества циклов необходимое для других конфигураций памяти и режимов повторения, приводится в главе 3, Классы и циклы команд.																																
Классы	Это поле уточняет класс команды для каждого синтаксиса команды. См. главу о классах и циклах команд, для определения каждого класса.																																
Пример	Образец кода включен в каждую команду. Действие кода на память и/или регистры суммируется, если это возможно.																																

20.3.2 ABDST Xmem, Ymem

Операнды	Xmem, Ymem: операнды в памяти данных двойного доступа.															
Код	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
операции	1	1	1	1	0	0	0	1	X	X	X	X	Y	Y	Y	Y
Выполнение	(B) + (A(32–16)) → B ((Xmem) - (Ymem)) << 16 → A															
Статусные биты	Изменяется под влиянием OVM, FRCT, и SXM. Влияет на C, OVA, и OVB.															
Описание	Эта команда рассчитывает абсолютное значение расстояния между векторами Xmem и Ymem. Абсолютное значение аккумулятора A(32–16) добавляется к значению аккумулятора B. Содержимое Ymem вычитается из Xmem, результат сдвигается влево на 16 бит и сохраняется в аккумуляторе A. Если установлен режим дробных чисел (FRCT = 1), то абсолютное значение умножается на 2.															
Слова	1 слово.															
Циклы	1 цикл.															
Классы	Класс 7.															
Пример	ABDST *AR3+, *AR4+															

Перед исполнением				После исполнения			
A	FF	ABCD	0000	A	FF	FFAB	0000
B	00	0000	0000	B	00	0000	5433
AR3			0100	AR3			0101
AR4			0200	AR4			0201
FRCT			0	FRCT			0
Память данных							
0100h			0055	0100h			0055
0200h			00AA	0200h			00AA

20.3.3 ABS src [, dst]

Операнды src, dstA (аккумулятор A)
B (аккумулятор B)

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	S	D	1	0	0	0	0	1	0	1

Выполнение |(src)| → dst (или src, если dst не задан).

Статусные биты OVM влияет на выполнение команды следующим образом:
если OVM = 1, то абсолютное значение 80 0000 0000h равно 00 7FFF FFFFh.
если OVM = 0, то абсолютное значение 80 0000 0000h равно 80 0000 0000h.
Осказывает влияние на C и OVdst (или OVsrc, если dst = src).

Описание Данная команда вычисляет абсолютное значение src и загружает это значение в dst. Если dst не задан, то абсолютное значение загружается в src. Если результат команды равен 0, то устанавливается бит C.

Слова 1 слово.

Циклы 1 цикл.

Классы Класс 1.

Пример 1 ABS A, B

Перед исполнением					После исполнения				
A	FF	FFFF	FFCB	-53	A	FF	FFFF	FFCB	-53
B	FF	FFFF	FC18	-1000	B	00	0000	0035	+53

Пример 2 ABS A

Перед исполнением					После исполнения				
A	03	1234	5678		A	00	7FFF	FFFF	
OVM				1	OVM				1

Пример 3 ABS A

Перед исполнением					После исполнения				
A	03	1234	5678		A	00	7FFF	FFFF	
OVM				0	OVM				0

20.3.4 ADD

- 1 ADD Smem, src
- 2 ADD Smem, TS, src
- 3 ADD Smem, 16, src [, dst]
- 4 ADD Smem [, SHIFT], src [, dst]
- 5 ADD Xmem, SHFT, src
- 6 ADD Xmem, Ymem, dst
- 7 ADD #lk [, SHFT], src [, dst]
- 8 ADD #lk, 16, src [, dst]
- 9 ADD src [, SHIFT], [, dst]
- 10 ADD src, ASM [, dst]

Операнды

Smem: операнд памяти данных одиночного доступа.

Xmem, Ymem: операнды памяти данных двойного доступа.

src, dstA: A (аккумулятор A).

B (аккумулятор B).

$-32\ 768 \leq lk \leq 32\ 767$

$-16 \leq SHIFT \leq 15$

$0 \leq SHFT \leq 15$

Код операции

1:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	S	I	A	A	A	A	A	A	A

2:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	S	I	A	A	A	A	A	A	A

3:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	1	1	S	D	I	A	A	A	A	A	A	A

4:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	1	1	1	1	I	A	A	A	A	A	A	A
0	0	0	0	1	1	S	D	0	0	0	S	H	I	F	T

5:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	0	0	0	S	X	X	X	X	S	H	F	T

6:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	0	0	0	D	X	X	X	X	Y	Y	Y	Y

7:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	S	D	0	0	0	0	S	H	F	T
16 битная константа															

8:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	S	D	0	0	1	1	0	0	0	0
16 битная константа															

9:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	S	D	0	0	0	S	H	I	F	T

10:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	S	D	1	0	0	0	0	0	0	0

Выполнение

- 1: (Smem) + (src) → src
- 2: (Smem) << (TS) + (src) → src
- 3: (Smem) << 16 + (src) → dst
- 4: (Smem) [<< SHIFT] + (src) → dst
- 5: (Xmem) << SHFT + (src) → src
- 6: ((Xmem) + (Ymem)) << 16 → dst
- 7: lk << SHFT + (src) → dst
- 8: lk << 16 + (src) → dst
- 9: (src or [dst]) + (src) << SHIFT → dst
- 10: (src or [dst]) + (src) << ASM → dst

Статусные биты

Изменяется под влиянием SXM и OVM.
Влияет на C и OVdst (или OVsrc, если dst = src).
Для синтаксиса команды 3, если результат сложения вызывает перенос, то бит переноса C, принимает значение 1; в противном случае, C остается неизменным.

Описание

Эта команда складывает 16-битное значение к содержимому выбранного аккумулятора или к 16-битному операнду Xmem в режиме адресации операнда в памяти данных двойного доступа. Слагаемое 16-битное значение может быть одним из следующих:

- 1 Содержимое операнда в памяти данных одиночного доступа (Smem);
- 2 Содержимое операнда в памяти данных двойного доступа (Ymem);
- 3 Операнд с непосредственным 16-битным значением (#lk);
- 4 Значение со сдвигом src.

Если dst указан, то данная команда сохраняет результат в dst. если dst не определен, то команда сохраняет результат в src. Большинство вторых операндов может быть со сдвигом. Для сдвига влево:

- 1 Младшие разряды принимают значение 0.
- 2 Старшие разряды:

- знак расширяется, если SXM = 1;
- принимает значение 0, если SXM = 0.

В случае сдвига вправо старшие разряды:

- знак расширяется, если SXM = 1;
- принимает значение 0, SXM = 0.

Примечание – Синтаксисы, указанные ниже, транслируются с языка ассемблера как различные синтаксисы в определенных случаях.

1 Синтаксис 4: если dst = src и SHIFT = 0, тогда код операции команды транслируется как синтаксис 1.

2 Синтаксис 4: если dst = src, SHIFT ≤ 15 и режим косвенной адресации Smem включен в Xmem, тогда код операции команды транслируется как синтаксис 5.

3 Синтаксис 5: если SHIFT = 0, тогда код операции команды транслируется как синтаксис 1.

Слова	Синтаксисы 1, 2, 3, 5, 6, 9, и 10: 1 слово. Синтаксисы 4, 7, и 8: 2 слова. Добавляется 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.
Циклы	Синтаксисы 1, 2, 3, 5, 6, 9, и 10: 1 цикл. Синтаксис 4, 7, и 8: 2 цикла. Добавляется 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.
Классы	Синтаксисы 1, 2, 3, и 5: класс 3А. Синтаксисы 1, 2, и 3: класс 3. Синтаксис 4: класс 4А. Синтаксис 4: класс 4В. Синтаксис 6: класс 7. Синтаксисы 7 и 8: класс 2. Синтаксисы 9 и 10: класс 1.
Пример 1	ADD *AR3+, 14, A

Перед исполнением				После исполнения			
A	00	0000	1200	A	00	0540	1200
C			1	C			01
AR3			0100	AR3			0101
SXM			1	SXM			1
Память данных							
0100h			1500	0100h			1500

Пример 2 ADD A, -8, B

Перед исполнением				После исполнения			
A	00	0000	1200	A	00	0000	1200
B	00	0000	1800	B	00	0000	1812
C			1	C			0

Пример 3 ADD #4568, 8, A, B

Перед исполнением				После исполнения			
A	00	0000	1200	A	00	0000	1200
B	00	0000	1800	B	00	0045	7A00
C			1	C			0

Пример 4 ADD *AR2+, *AR2-, A ; после доступа к операнду, AR2 увеличивается на 1.
Пример 4 показывает один и тот же вспомогательный регистр (AR2) с различными режимами адресации, установленный для обоих операндов. Режим, установленный полем Xmod (*AR2+), используется для адресации.

20.3.5 ADDC Smem, src

Операнды Smem операнд в памяти данных одиночного доступа.
src: A (аккумулятор A).
B (аккумулятор B).

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	1	S	I	A	A	A	A	A	A	A

Выполнение (Smem) + (src) + (C) → src

Статусные биты Результат зависит от OVM, C.

Влияет на C и OVsrc.

Описание Эта команда складывает 16-битный операнд в памяти данных одиночного доступа Smem и значение бита переноса (C) с src. Команда сохраняет результат в src. Знаковый разряд не расширяется независимо от значения бита SXM.

Слова 1 слово.

Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Циклы 1 цикл

Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Классы Класс 3A.

Класс 3B.

Пример 1 ADDC *+AR2(5), A

Перед исполнением				После исполнения			
A	00000013			A	00000018		
C	00000001			C	00000000		
AR2	00000100			AR2	00000105		
Память данных							
0105h	00000004			0105h	00000004		

20.3.6 ADDM #lk, Smem

Операнды Smem: операнд памяти данных одиночного доступа.
 $-32\,768 \leq lk \leq 32\,767$

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	1	1	0	1	0	1	1	I	A	A	A	A	A	A	A
16 битная константа																

Выполнение #lk + (Smem) → Smem

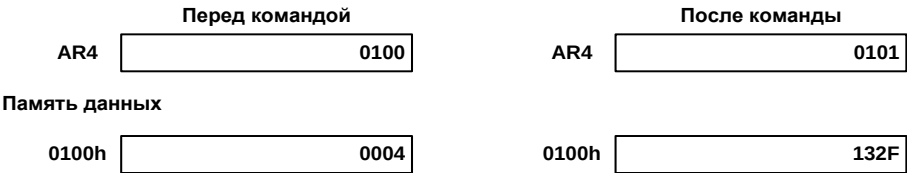
Статусные биты Результат зависит от OVM и SXM.

Влияет на C и OVA.

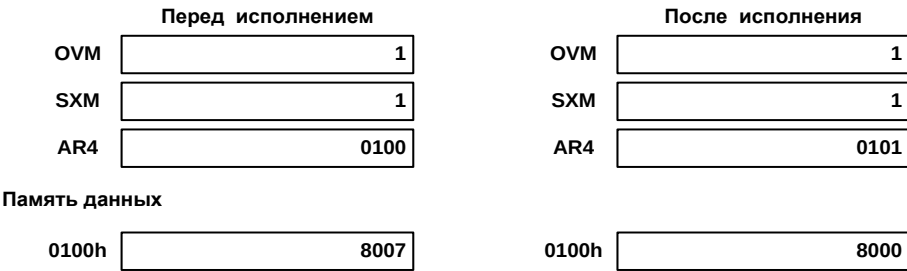
Описание Эта команда добавляет 16-битный операнд в памяти данных одиночного доступа Smem к 16-битному непосредственному значению памяти lk и сохраняет результат в Smem.

Примечание – Эта команда не повторяемая.

Слова	2 слова. Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.
Циклы	2 цикла. Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.
Классы	Класс 18A. Класс 18B.
Пример 1	ADDM 0123Bh, *AR4+



Пример 2	ADDM 0FFF8h, *AR4+



20.3.7 ADDS Smem, src

Операнды	Smem: операнд в памяти данных одиночного доступа. src: A (аккумулятор A). B (аккумулятор B).																																
Код операции	<table><tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>S</td><td>I</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td></tr></table>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	0	0	0	0	0	0	1	S	I	A	A	A	A	A	A	A
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																		
0	0	0	0	0	0	1	S	I	A	A	A	A	A	A	A																		
Выполнение	uns(Smem) + (src) → src																																
Статусные биты	На результат влияет OVM. Сама инструкция влияет на C и OVsrc.																																
Описание	Эта команда складывает 16-битный операнд в памяти данных одиночного доступа Smem к src и сохраняет результат в src. Расширение знака подавляется независимо от значения бита SXM.																																
Слова	1 слово. Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.																																
Циклы	1 цикл. Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.																																
Классы	Класс 3A. Класс 3B.																																
Пример	ADDS *AR2-, B																																

Статусные биты	Не используются
Описание	Данная команда реализует логическое умножение(AND) с src одного из следующих операндов: 1 16-битного операнда Smem; 2 16-битного непосредственного операнда lk; 3 Аккумулятора источник или аккумулятора приемник (src или dst). Когда определен сдвиг, то команда сдвигает операнд влево до выполнения AND. При сдвиге влево значения младших разрядов обнуляются, а старший разряд знака не расширяется. При сдвиге вправо к старшие разряды знака не расширяются.
Слова	Синтаксис 1 и 4: 1 слово. Синтаксис 2 и 3: 2 слова. Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.
Циклы	Синтаксис 1 и 4: 1 цикл. Синтаксис 2 и 3: 2 цикла. Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.
Классы	Синтаксис 1: класс 3А. Синтаксис 1: класс 3В. Синтаксис 2 и 3: класс 2. Синтаксис 4: класс 1.
Пример 1	AND *AR3+, A

Перед исполнением			После исполнения		
A	00	00FF 1200	A	00	0000 1000
AR3		0100	C		0101
Память данных					
0100h		1500	0100h		1500

Пример 2 AND A, 3, B

Перед исполнением			После исполнения		
A	00	0000 1200	A	00	0000 1200
B	00	0000 1800	B	00	0000 1000

20.3.9 ANDM #lk, Smem

Операнды Smem: операнд в памяти данных одиночного доступа.
 $0 \leq lk \leq 65\,535$

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	1	1	0	1	0	0	0	I	A	A	A	A	A	A	A
	16 битная константа															

Выполнение lk AND (Smem) → Smem

Статусные биты Не используются.

Описание Данная команда производит логическое умножение (AND) 16-битного операнда памяти данных одиночного доступа Smem и 16-разрядной длинной константы lk. Результат сохраняется в ячейке памяти данных, определяемой Smem.

Примечание – Данная команда не повторяемая.

Слова 2 слова.
Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Циклы 2 цикла.
Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Классы Класс 18A.
Класс 18B.

Пример 1 ANDM #00FFh, *AR4+

Перед исполнением		После исполнения	
AR4	0100	AR4	0101
Память данных			
0100h	0444	0100h	0044

Пример 2 ANDM #0101h, 4; DP = 0

Перед исполнением				После исполнения			
Память данных				Память данных			
0004h	00	0000	0100	0004h	00	0000	0100

20.3.10 B[D] pmad

Операнды $0 \leq \text{pmad} \leq 65\,535$

Код операции

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	Z	0	0	1	1	1	0	0	1	1
16 битная константа															

Выполнение pmad → PC

Статусные биты Не используются.

Описание Данная команда передает управление по адресу памяти программ (pmad), который может быть задан как символически так и числом. Если переход задержанный (определяется по наличию суффикса D), то две однословные команды или одна двухсловная команда, следующие за командой перехода, вызываются из памяти программ и выполняются.

Примечание – Командна не повторяемая.

Слова 2 слова.

Циклы 4 цикла.
2 цикла (задержанный).

Классы Класс 29A.

Пример 1 В 2000h

Перед командой		После команды	
PC	1F45	PC	2000

Пример 2 BD 1000h
ANDM 4444h, *AR1+

Перед командой		После команды	
PC	1F45	PC	1000

После того, как команда AND была выполнена над операндом со значением 4444h, программа продолжает выполнение в ячейке 1000h.

20.3.11 BACC[D] src

Операнды src: A (аккумулятор A)
B (аккумулятор B)

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	Z	S	1	1	1	0	0	0	1	0

Выполнение (src(15–0)) → PC
Статусные биты Не используются.

Описание Данная команда передает управление по 16-разрядному адресу в младшей части src (биты 15–0). Если переход задержанный (определяется по наличию суффикса D), то две команды с одним словом или одна команда с двумя словами, следующие за командой перехода, вызываются из памяти программ и выполняются.

Примечание – Данная команда не повторяющаяся.

Слова 1 слово.
Циклы 6 циклов.
4 цикла (задержанный).

Классы Класс 30A.

Пример 1 BACC A

Перед командой		После команды	
A	00 0000 3000	A	00 0000 3000
PC	1F45	PC	3000

Пример 2 BACCD B
ANDM 4444h, *AR1+

Перед командой		После команды	
B	00 0000 2000	B	00 0000 2000
PC	1F45	PC	2000

После того, как команда AND была выполнена над операндом со значением 4444h, программа продолжает выполнение в ячейке 2000h.

20.3.12 BANZ[D] pmad, Sind

Операнды Sind: операнд косвенной адресации одиночного доступа.
 $0 \leq \text{pmad} \leq 65\,535$

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	1	Z	0	I	A	A	A	A	A	A	A
16 битная константа															

Выполнение If ((ARx) ≠ 0)
then
 pmad → PC
Else
 (PC) + 2 → PC

Статусные биты Не используются.

Описание Если значение текущего вспомогательного регистра ARx не равно 0, команда осуществляет переход по указанному адресу памяти программ (pmad). В противном случае, PC увеличивается на 2. Если переход задержанный (определяется по наличию суффикса D), то две команды с одним словом или одна команда с двумя словами, следующие за командой перехода, вызываются из памяти программ и выполняются.

Примечание – Данная команда не повторяемая.

Слова 2 слова.

Циклы 4 цикла (истинное условие).
2 цикла (ложное условие).
2 цикла (задержанная).

Классы Класс 29A.

Пример 1 BANZ 2000h, *AR3–

Перед командой		После команды	
PC	1000	PC	2000
AR3	0005	AR3	0004

Пример 2 BANZ 2000h, *AR3–

Перед командой		После команды	
PC	1000	PC	1002
AR3	0000	AR3	FFFF

Пример 3 BANZ 2000h, *AR3(–1)

Перед командой		После команды	
PC	1000	PC	1003
AR3	0001	AR3	0001

Пример 4 BANZD 2000h, *AR3–
ANDM 4444h, *AR5+

Перед командой		После команды	
PC	1000	PC	2000
AR3	0004	AR3	0003

После того, как ячейку памяти логически умножается (AND) на значение 4444h, программа продолжает выполнение с ячейки 2000h.

20.3.13 BC[D] pmad, cond [, cond [, cond]]

Операнды

$0 \leq \text{pmad} \leq 65\,535$

В таблице ниже приводятся условия для данной команды (операнд условия).

Операнд условия	Описание	Код условия	Операнд условия	Описание	Код условия
BIO	$\overline{\text{BIO}}_{\text{low}}$	0000 0011	NBIO	$\overline{\text{BIO}}_{\text{high}}$	0000 0010
C	C=1	0000 1100	NC	C=0	0000 1000
TC	TC=1	0011 0000	NTC	TC=0	0010 0000
AEQ	(A) = 0	0100 0101	BEQ	(B) = 0	0100 1101
ANEQ	(A) $\neq$ 0	0100 0100	BNEQ	(B) $\neq$ 0	0100 1100
AGT	(A) > 0	0100 0110	BGT	(B) > 0	0100 1110
AGEQ	(A) $\geq$ 0	0100 0010	BGEQ	(B) $\geq$ 0	0100 1010
ALT	(A) < 0	0100 0011	BLT	(B) < 0	0100 1011
ALEQ	(A) $\leq$ 0	0100 0111	BLEQ	(B) $\leq$ 0	0100 1111
AOV	A переполнение	0111 0000	BOV	B переполнение	0111 1000
ANOV	A отсутствие переполнения	0110 0000	BNOV	B отсутствие переполнения	0110 1000
UNC	безусловный	0000 0000			

Код операции

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	1	0	Z	0		C	C	C	C	C	C	C
16 битная константа															

Выполнение

If (cond(s))

Then

$\text{pmad} \rightarrow \text{PC}$

Else

$(\text{PC}) + 2 \rightarrow \text{PC}$

Статусные биты

Зависит от OVA или OVB, если выбран OV или NOV.

Описание

Данная команда осуществляет переход по адресу памяти программ (pmad) в случае выполнения определенного(ых) условия(й). Две команды однословные или одна команда двухсловная, следующие за командой перехода извлекаются из памяти программ. Если условие(я) выполняется(ются), то два слова, следующие за командой, удаляются из конвейера и начинается выполнение с адреса pmad. Если условие(я) не выполняется(ются), то значение PC увеличивается на 2 и выполняются два слова, следующие за командой.

Если переход задержанный (определяется по наличию суффикса D), то две команды по одному слову или одна команда с двумя словами извлекаются из памяти программ и выполняются. Два слова инструкции(й), следующие за задержанной командой, не влияют на анализируемое условие. Если условие выполняется, то выполнение будет продолжено по адресу pmad. Если условие не выполняется, то значение PC увеличивается на 2 и выполняются два слова инструкций, следующие за задержанной командой.

Данная команда проверяет множественные условия перед передачей управления к другой части программы. Данная команда осуществляет проверку как одиночных условий, так и сочетание условий. Вы можете сочетать условия только из одной перечисленной ниже группы.

Группа 1: Вы можете выбрать не более двух условий. Каждое из этих условий должно быть из различных категорий (категория А или В); у вас не должно быть двух условий из одной категории. Например, вы можете осуществлять проверку EQ и OV одновременно, но не можете одновременно тестировать GT и NEQ. Для обоих условий аккумулятор должен быть один и тот же; одной командой нельзя осуществлять проверку сразу двух аккумуляторов. Например, вы можете одновременно осуществлять проверку AGT и AOV и не можете проверять одновременно AGT и BOV.

Группа 2: Вы можете выбрать не более трех условий. Каждое из этих условий должно быть из различных категорий (категория А, В, или С); у вас не должно быть двух условий из одной категории. Например, вы можете осуществлять проверку TC, C, и BIO одновременно, но не можете одновременно тестировать NTC, C, и NC.

Условия для данной команды

Группа 1		Группа 2		
Категория А	Категория В	Категория А	Категория В	Категория С
EQ	OV	TC	C	BIO
NEQ	NOV	NTC	NC	NBIO
LT				
LEQ				
GT				
GEO				

Примечание – Данная инструкция не повторяемая.

Слова

2 слова.

Циклы

5 циклов (истинное условие).
3 цикла (ложное условие).
3 цикла (задержанная).

Классы

Класс 31А.

Пример 1

BC 2000h, AGT

Перед выполнением			
A	00	0000	0053
PC	1000		

После выполнения			
A	00	0000	0053
PC	2000		

Пример 2

BC 2000h, AGT

Перед выполнением			
A	FF	FFFF	FFFF
PC	1000		

После выполнения			
A	FF	FFFF	FFFF
PC	1002		

Пример 3

BCD 1000h, BOV
ANDM 4444h, *AR1+

Перед выполнением	
PC	3000
OVV	1

После выполнения	
PC	1000
OVV	1

После того, как в ячейку памяти добавляется (AND) значение 4444h, переход осуществляется при выполнении условия (OVV). В противном случае, выполнение продолжается командой, следующей за данной.

Пример 4 BC 1000h, TC, NC, BIO

Перед выполнением		После выполнения	
PC	3000	PC	3002
C	1	C	1

20.3.14 BIT Xmem, BITC

Операнды Xmem: операнд в памяти данных двойного доступа $0 \leq \text{BITC} \leq 15$

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	0	1	0	1	1	0	X	X	X	X	B	I	T	C

Выполнение (Xmem(15 – BITC)) → TC

Статусные биты Результат влияет на TC.

Описание Данная команда копирует указанный бит операнда в памяти данных двойного доступа Xmem в бит TC статусного регистра ST0. В таблице ниже приведены коды битов, соответствующие каждому биту в памяти данных.

Код бита соответствует BITC, а адрес бита соответствует (15 – BITC).

Коды битов для данной команды

Адрес бита	Код бита	Адрес бита	Код бита
(LSB) 0	1111	8	0111
1	1110	9	0110
2	1101	10	0101
3	1100	11	0100
4	1011	12	0011
5	1010	13	0010
6	1001	14	0001
7	1000	(MSB) 15	0000

Слова 1 слово.

Циклы 1 цикл.

Классы Класс 3A.

Пример BIT *AR5+, 15-12; test bit 12

Перед исполнением		После исполнения	
AR5	0100	AR5	0101
TC	0	TC	1
Память данных			
0100h	7688	0100h	7688

20.3.15 BITF Smem, #lk

Операнды Smem: операнд в памяти данных одиночного доступа.
 $0 \leq lk \leq 65\,535$

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	1	1	0	0	0	0	1	I	A	A	A	A	A	A	A
	16 битная константа															

Выполнение If ((Smem) AND lk) = 0
Then
 0 → TC
Else
 1 → TC

Статусные биты Влияет на TC.

Описание Данная команда осуществляет проверку определенного бита или битов ячейки памяти данных Smem.
Если указанный бит (биты) принимает значение 0, то бит TC статусного регистра ST0 сбрасывается; в противном случае, TC принимает значение 1.
константа lk является маской тестируемого бита или битов.

Слова 2 слова.
Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Циклы 2 цикла.
Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Классы Класс 6A.
Класс 6B.

Пример 1 BITF 5, 00FFh

Перед исполнением		После исполнения	
TC	x	TC	0
DP	004	DP	004
Память данных			
0205h	5400	0205h	5400

Пример 2 BITF 5, 0800h

Перед исполнением		После исполнения	
TC	x	TC	0
DP	004	DP	004
Память данных			
0205h	0F7F	0205h	0F7F

20.3.16 BITT Smem

Операнды Smem: операнд памяти данных одиночного доступа.

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	1	1	0	1	0	0	I	A	A	A	A	A	A	A

Выполнение (Smem (15 – T(3–0))) → TC

Статусные биты Операция устанавливает бит TC.

Описание Данная команда копирует указанный бит значения памяти данных Smem в бит TC статусного регистра ST0. Четыре младших разряда T содержат код бита, который указывает на то, какой именно бит копируется. Адрес бита соответствует (15 – T(3–0)). Код бита соответствует содержимому T(3–0).

Коды битов для данной команды

Адрес бита	Код бита	Адрес бита	Код бита
(LSB) 0	1111	8	0111
1	1110	9	0110
2	1101	10	0101
3	1100	11	0100
4	1011	12	0011
5	1010	13	0010
6	1001	14	0001
7	1000	(MSB) 15	0000

Слова 1 слово.

Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Циклы 1 цикл.

Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Классы Класс 3А.

Класс 3В.

Пример BITT *AR7+0

Перед командой		После команды	
T	C	T	C
TC	0	TC	1
AR0	0008	AR0	0008
AR7	0100	AR7	0108
Память данных			
0100h	0008	0100h	0008

20.3.17 CALA[D] src

Операнды src: A (аккумулятор A).
B (аккумулятор B).

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	Z	S	1	1	1	0	0	0	1	1

Выполнение Без задержки:
(SP) – 1 → SP;
(PC) + 1 → TOS;
(src(15–0)) → PC.
С задержкой:
(SP) – 1 → SP;
(PC) + 3 → TOS;
(src(15–0)) → PC.

Статусные биты Не используются.

Описание Данная команда передает управление по 16-разрядному адресу в младшей части src (биты 15–0). Если переход задержанный (определяется по наличию суффикса D), то две однословные команды одна двухсловная команда, следующие за командой перехода, вызываются из памяти программ и выполняются.

Примечание – Эта команда не может повторяться.

Слова 1 слово.
Циклы 6 циклов.
4 циклов (задержанная).
Классы Класс 30В.
Пример 1 CALA A

Перед командой		После команды	
PC	0025	PC	3333
SP	1111	SP	1110
Память данных			
1110h	4567	1110h	0027

Пример 2 CALAD B
ANDM 4444h, *AR1+

Перед командой		После команды	
B	00 0000 2000	B	00 0000 2000
PC	0025	PC	2000
SP	1111	SP	1110
Память данных			
1110h	4567	1110h	0028

После того, как содержимое ячейки памяти логически умножается (AND) со значением 4444h, программа продолжает выполнение с адреса 2000h.

20.3.18 CALL[D] pmad

Операнды	$0 \leq \text{pmad} \leq 65\,535$															
Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	0	Z	0	0	1	1	1	0	1	0	0
	16 битная константа															

Выполнение Без задержки:
(SP) - 1 → SP;
(PC) + 2 → TOS;
pmad → PC.
С задержкой:
(SP) - 1 → SP;
(PC) + 4 → TOS;
pmad → PC.

Статусные биты Не используются.

Описание Данная команда передает управление по указанному адресу памяти программ (pmad). Адрес возврата помещается в TOS стека до того, как загрузится pmad в PC. Если вызов задерживается (определяется по наличию суффикса D), то две команды однословные или одна команда двухсловная, следующие за командой перехода, вызываются из памяти программ и выполняются.

Примечание – Эта команда не может повторяться.

Слова 2 слова.
Циклы 4 цикла.
2 цикла (с задержкой).

Классы Класс 29В.

Пример 1 CALL 3333h

Перед командой		После команды	
PC	0025	PC	3333
SP	1111	SP	1110
Память данных			
1110h	4567	1110h	0027

Пример 2 CALLD 1000h
ANDM #4444h, *AR1+

Перед командой		После команды	
PC	0025	PC	1000
SP	1111	SP	1110
Память данных			
1110h	4567	1110h	0029

После того, как в ячейка памяти логически умножается (AND) со значением 4444h, программа продолжает выполнение с адреса 1000h.

20.3.19 CC[D] pmad, cond [, cond [, cond]]

Операнды $0 \leq \text{pmad} \leq 65\,535$

В таблице ниже перечислены условия (операнды условий) для данной команды.

Условие	Описание	Код условия	Условие	Описание	Код условия
BIO	$\overline{\text{BIO}}_{\text{low}}$	0000 0011	NBIO	$\overline{\text{BIO}}_{\text{high}}$	0000 0010
C	C=1	0000 1100	NC	C=0	0000 1000
TC	TC=1	0011 0000	NTC	TC=0	0010 0000
AEQ	(A) = 0	0100 0101	BEQ	(B) = 0	0100 1101
ANEQ	(A) $\neq$ 0	0100 0100	BNEQ	(B) $\neq$ 0	0100 1100
AGT	(A) > 0	0100 0110	BGT	(B) > 0	0100 1110
AGEQ	(A) $\geq$ 0	0100 0010	BGEQ	(B) $\geq$ 0	0100 1010
ALT	(A) < 0	0100 0011	BLT	(B) < 0	0100 1011
ALEQ	(A) $\leq$ 0	0100 0111	BLEQ	(B) $\leq$ 0	0100 1111
AOV	A переполнение	0111 0000	BOV	B переполнение	0111 1000
ANOV	A отсутствие переполнения	0110 0000	BNOV	B отсутствие переполнения	0110 1000
UNC	безусловный	0000 0000			

Код

операции

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	0	Z	1	C	C	C	C	C	C	C	C
16 битная константа															

Выполнение

Без задержки:

If (cond(s))

Then

(SP) - 1 $\rightarrow$ SP

(PC) + 2 $\rightarrow$ TOS

pmad $\rightarrow$ PC

Else

(PC) + 2 $\rightarrow$ PC

С задержкой:

If (cond(s))

Then

(SP) - 1 $\rightarrow$ SP

(PC) + 4 $\rightarrow$ TOS

pmad $\rightarrow$ PC

Else

(PC) + 2 $\rightarrow$ PC

Статусные биты

Выполнение зависит от OVA или OVB (если выбраны OV или NOV).

Описание

Данная команда передает управления в адресе памяти программ (pmad) если выполняется(ются) определенное(ые) условие(я). Две команды с одним словом или одна команда с двумя словами, следующие за командой перехода, вызываются из памяти программ.

Если условие(я) выполняется(ются), то два слова, следующие за командой, удаляются из конвейера и начинается выполнение с адреса pmad. Если условие(я) не выполняется(ются), то значение PC увеличивается на 2 и выполняются два слова, следующие за командой.

Если вызов задержанный (определяется по наличию суффикса D), то две однословные команды или одна двухсловная команда извлекаются из памяти

программ и выполняются. Два слова, следующие за задержанной командой, не влияют на тестируемое условие. Если условие выполняется, то выполнение будет продолжено с адреса *rtad*. Если условие не выполняется, то значение *PC* увеличивается на 2 и выполняются инструкции определенные двумя словами, следующие за задержанной командой.

Данная команда проверяет множественные условия перед передачей управления в другую часть программы. Данная команда осуществляет проверку как одиночных условий так и их совокупности. Вы можете сочетать условия только в одной из перечисленных ниже групп

Группа 1: Вы можете выбрать не более двух условий. Каждое из этих условий должно быть из различных категорий (категория А или В); у вас не должно быть двух условий из одной категории. Например, вы можете осуществлять проверку EQ и OV одновременно, но не можете одновременно тестировать GT и NEQ. Для обоих условий аккумулятор должен быть один и тот же; одной командой нельзя осуществлять проверку сразу двух аккумуляторов. Например, вы можете одновременно осуществлять проверку AGT и AOV и не можете проверять одновременно AGT и BOV.

Группа 2: Вы можете выбрать не более трех условий. Каждое из этих условий должно быть из различных категорий (категория А, В, или С); у вас не должно быть двух условий из одной категории. Например, вы можете осуществлять проверку TC, C, и BIO одновременно, но не можете одновременно тестировать NTC, C, и NC.

Условия для данной команды

Группа 1		Группа 2		
Категория А	Категория В	Категория А	Категория В	Категория С
EQ	OV	TC	C	BIO
NEQ	NOV	NTC	NC	NBIO
LT				
LEQ				
GT				
GEO				

Примечание – Данная команда не повторяемая.

Слова	2 слова.
Циклы	5 циклов (истинное условие). 3 цикла (ложное условие). 3 цикла (с задержкой).
Классы	Класс 31В.
Пример 1	CC 2222h, AGT

Перед командой				После команды			
A	00	0000	3000	A	00	0000	3000
PC	0025			PC	2222		
SP	1111			SP	1110		

Память данных

1110h	4567	1110h	0027

Пример 2 CCD 1000h, BOV
ANDM 4444h, *AR1+

Перед командой			После команды		
PC		0025	PC		1000
OVb		1	OVb		1
SP		1111	SP		1110
Память данных					
1110h		4567	1110h		0029

После того, как ячейка памяти логически умножается (AND) на значение 4444h, программа продолжает выполнение с ячейки 1000h.

20.3.20 CMPL src [, dst]

Операнды src, dst: A (аккумулятор A).
B (аккумулятор B).

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	S	D	1	0	0	1	0	0	1	1

Выполнение ($\overline{\text{src}}$) $\rightarrow$ dst

Статусные биты Не используются.

Описание

Данная команда вычисляет обратный код содержимого src (логическая инверсия). Результат сохраняется в dst, если он указан, или, в противном случае, в src.

Слова 1 слово.

Циклы 1 цикл.

Классы Класс 1.

Пример CMPL A, B

Перед исполнением				После исполнения			
A	FC	DFFA	AEAA	A	FC	DFFA	AEAA
B	00	0000	7899	B	03	2005	5155

20.3.21 CMPM Smem, #lk

Операнды Smem: операнд памяти данных одиночного доступа.
 $-32\,768 \leq lk \leq 32\,767$

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	1	1	0	0	0	0	0	I	A	A	A	A	A	A	A
16 битная константа																

Выполнение If (Smem) = lk

Then

1 $\rightarrow$ TC

Else

0 $\rightarrow$ TC

Статусные биты	Команда влияет на значение ТС.
Описание	Эта команда сравнивает 16-разрядный операнд памяти данных одиночного доступа Smem с 16-разрядной константой constant lk . если они равны, ТС принимает значение 1. В противном случае, значение ТС сбрасывается.
Слова	2 слова. Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.
Циклы	2 цикла Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.
Классы	Класс 6A. Класс 6B.
Пример	CMPPM *AR4+, 0404h

Перед исполнением		После исполнения	
ТС	1	ТС	1000
AR4	0100	AR4	1110
Память данных			
0100h	4444	0100h	4444

20.3.22 CMPPR CC, ARx

Операнды	$0 \leq CC \leq 3$ ARxAR0–AR7																																
Код операции	<table><tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>C</td><td>C</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>A</td><td>R</td><td>X</td></tr></table>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	1	1	1	1	0	1	C	C	1	0	1	0	1	A	R	X
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																		
1	1	1	1	0	1	C	C	1	0	1	0	1	A	R	X																		
Выполнение	If (cond) Then $1 \rightarrow TC$ Else $0 \rightarrow TC$																																
Статусные биты	Команда влияет на значение TC.																																
Описание	Данная команда сравнивает содержание определенного регистра (ARx) с содержимым AR0 и устанавливает бит TC в соответствии с результатом сравнения. Сравнение определяется значением кода условия CC (см. таблицу ниже). Если условие истинно, то TC принимает значение 1. если условие ложно, то значение TC сбрасывается. Все условия рассматриваются так, как если бы операнды были беззнаковыми.																																
	<table><tr><th>Условие</th><th>Код условия</th><th>Описание</th></tr><tr><td>EQ</td><td>00</td><td>Test if (ARx) = (AR0)</td></tr><tr><td>LT</td><td>01</td><td>Test if (ARx) < (AR0)</td></tr><tr><td>GT</td><td>10</td><td>Test if (ARx) > (AR0)</td></tr><tr><td>NEQ</td><td>11</td><td>Test if (ARx) ≠ (AR0)</td></tr></table>	Условие	Код условия	Описание	EQ	00	Test if (ARx) = (AR0)	LT	01	Test if (ARx) < (AR0)	GT	10	Test if (ARx) > (AR0)	NEQ	11	Test if (ARx) ≠ (AR0)																	
Условие	Код условия	Описание																															
EQ	00	Test if (ARx) = (AR0)																															
LT	01	Test if (ARx) < (AR0)																															
GT	10	Test if (ARx) > (AR0)																															
NEQ	11	Test if (ARx) ≠ (AR0)																															
Слова	1 слово.																																

Циклы 1 цикл.
Классы Класс 1.
Пример CMPR 2, AR4

Перед командой		После команды	
TC	1	TC	0
AR0	FFFF	AR0	FFFF
AR4	7FFF	AR4	7FFF

20.3.23 CMPS src, Smem

Операнды src: A (аккумулятор A).
B (аккумулятор B).
Smem: операнд памяти данных одиночного доступа.

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	0	0	1	1	1	S	I	A	A	A	A	A	A	A

Выполнение If ((src(31–16)) > (src(15–0)))
Then
 (src(31–16)) → Smem
 (TRN) << 1 → TRN
 0 → TRN(0)
 0 → TC
Else
 (src(15–0)) → Smem
 (TRN) << 1 → TRN
 1 → TRN(0)
 1 → TC

Статусные биты Инструкция влияет на значение TC.

Описание Эта команда сравнивает два 16-разрядных значения в дополнительном двоичном коде, расположенных в старшей и младшей части src и сохраняет максимальное значение в ячейке памяти данных Smem. Если старшая часть src (биты 31–16) больше, то 0 вдвигается в младший разряд (LSB) регистра перехода (TRN) и значение бита TC сбрасывается. Если младшая часть src (биты 15–0) больше, то 1 вдвигается в LSB регистра TRN и бит TC принимает значение 1.

Слова 1 слово.
Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Циклы 1 цикл.
Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Классы Класс 10А.
Класс 10В.

Пример CMPS A, *AR4+

Перед исполнением				После исполнения			
A	00	2345	7899	A	00	2345	7899
TC				0	1		
AR4	0100			AR4	0101		
TRN	4444			TRN	8889		
Память данных							
0100h	0000			0100h	7899		

20.3.24 DADD Lmem, src [, dst]

Операнды Lmem: операнд памяти данных, использующий адресацию длинного слова.
src, dst: A (аккумулятор A).
B (аккумулятор B).

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	1	0	1	0	0	S	D	I	A	A	A	A	A	A	A

Выполнение If C16 = 0
Then
(Lmem) + (src) → dst
Else
(Lmem(31–16)) + (src(31–16)) → dst(39–16)
(Lmem(15–0)) + (src(15–0)) → dst(15–0)

Статусные биты Исполнение зависит от SXM и OVM (только если C16 = 0).
Команда влияет на значение C и OVdst (или OVsrc, если dst не определен).

Описание Данная команда добавляет содержимое src к 32-разрядному операнду памяти данных, использующему адресацию длинного слова, Lmem. Если dst определен, то команда сохраняет результат в dst. Если dst не определен, то команда сохраняет результат в src. Значение C16 определяет режим команды: если C16 = 0, команда выполняется в режиме двойной точности. 40-разрядное значение src добавляется к Lmem. Биты насыщения и переполнения устанавливаются в зависимости от результата операции.
если C16 = 1, команда выполняется в двойном 16-разрядном режиме. Старшая часть src (биты 31–16) добавляются к 16 самым старшим битам (MSB) операнда Lmem, а младшая часть src (биты 15–0) добавляются к 16 самым младшим битам (LSB) операнда Lmem. Биты насыщения и переполнения не изменяются в этом режиме. В этом режиме результаты являются ненасыщенными, независимо от состояния бита OVM.

Слова 1 слово.
Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem.

Циклы 1 цикл.
Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem.

Классы Класс 9А.
Класс 9В.
Пример 1 DADD *AR3+, A, B

Перед исполнением				После исполнения			
A	00	5678	8933	A	00	5678	8933
B	00	0000	0000	B	00	6BAC	BD89
C16				C16			
AR3				AR3↑			
Память данных							
0100h				0100h			
0101h				0101h			

* Данная команда является командой с длинным операндом и как следствие после его выполнения AR3 увеличивается на 2.

Пример 2 DADD *AR3–, A, B

Перед исполнением				После исполнения			
A	00	5678	3933	A	00	5678	3933
B	00	0000	0000	B	00	6BAC	6D89
C16				C16	1		
AR3				AR3†	00FE		
Память данных							
0100h				0100h	1534		
0101h				0101h	3456		

* Данная команда является командой с длинным операндом, и потому после выполнения AR3 уменьшается на 2.

Пример 3 DADD *AR3–, A, B

Перед командой				После команды			
A	00	5678	3933	A	00	5678	3933
B	00	0000	0000	B	00	8ACE	4E67
C16	0			C16	0		
AR3	0101			AR3†	00FF		
Память данных							
0100h	1534			0100h	1534		
0101h	3456			0101h	3456		

* Данная команда является командой с длинным операндом, и потому после выполнения AR3 уменьшается на 2.

20.3.25 DADST Lmem, dst

Операнды Lmem: операнд памяти данных, использующий адресацию длинного слова.
dst: A (аккумулятор A).
B (аккумулятор B).

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	1	0	1	1	0	1	D	I	A	A	A	A	A	A	A

Выполнение If C16 = 1
Then
 $(Lmem(31-16)) + (T) \rightarrow dst(39-16)$
 $(Lmem(15-0)) - (T) \rightarrow dst(15-0)$
Else
 $(Lmem) + ((T) + (T) << 16) \rightarrow dst$

Статусные биты На результат инструкции влияет SXM и OVM (только если C16 = 0)
Операция влияет на C и OVdst.

Описание Данная команда добавляет содержимое T к 32-разрядному операнду памяти данных, использующему адресацию длинного слова Lmem. Значение C16 определяет режим исполнения команды:
Если C16 = 0, команда выполняется в режиме двойной точности. Lmem добавляется к 32-разрядному значению, состоящему из содержимого T сочлененному с содержимым 16 битов того же регистра T сдвинутого влево на 16 разрядов ($T << 16 + T$). Результат сохраняется в dst.
Если C16 = 1, команда выполняется в двойном 16-разрядном режиме. 16 самых старших битов операнда Lmem добавляются к содержимому T и сохраняются в старших 24 битах dst. В то же время содержимое T вычитается из 16 самых младших битов операнда Lmem. В этом режиме результаты являются ненасыщенными, независимо от состояния бита OVM.
Примечание – Эта команда имеет смысловую интерпретацию только когда C16 принимает значение 1 (двойной 16-битный режим).

Слова 1 слово.
Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem.

Циклы 1 цикл.
Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem.

Классы Класс 9A.
Класс 9B.

Пример 1 DADST *AR3–, A

Перед командой				После команды			
A	00	0000	0000	A	00	3879	579B
T				T	2345		
C16	0			C16	0		
AR3	0100			AR3†	0102		
Память данных							
0100h	1534			0100h	1534		
0101h	3456			0101h	3456		

* Данная команда использует длинный операнд и после выполнения AR3 уменьшается на 2.

Пример 2 DADST *AR3+, A

Перед командой				После команды			
A	00	0000	0000	A	00	3879	579B
T				T	2345		
C16	0			C16	0		
AR3	0100			AR3†	0102		
Память данных							
0100h	1534			0100h	1534		
0101h	3456			0101h	3456		

* Данная команда использует длинный операнд и после выполнения AR3 увеличивается на 2.

20.3.26 DELAY Smem

Операнды	Smem: операнд памяти данных одиночного доступа.															
Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	1	0	0	1	1	0	1	I	A	A	A	A	A	A	A
Выполнение	(Smem) → Smem + 1															
Статусные биты	Не используются.															
Описание	Данная команда копирует содержимое ячейки памяти данных Smem в ячейку памяти по следующему большему адресу. Когда данные скопированы, содержимое адресуемой ячейки остается таким же. Эта функция может быть полезной при осуществлении задержки Z при применении в цифровой обработке сигналов. Операция задержки также присутствует в командах загрузки T (инструкция LTD) и инструкции умножения ячейки памяти программ с ячейкой памяти данных с накоплением (MACD).															
Слова	1 слово. Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.															
Циклы	1 цикл Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.															
Классы	Класс 24A. Класс 24B.															
Пример	DELAY *AR3															

Перед исполнением		После исполнения	
AR3	0100	AR3	0100
Память данных			
0100h	6CAC	0100h	6CAC
0101h	0000	0101h	6CAC

20.3.27 DLD Lmem, dst

Операнды Lmem: операнд памяти данных, использующий адресацию длинного слова.
dst: A (аккумулятор A).
B (аккумулятор B).

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	1	0	1	0	1	1	D	I	A	A	A	A	A	A	A

Выполнение If C16 = 0
Then
(Lmem) → dst
Else
(Lmem(31–16)) → dst(39–16)
(Lmem(15–0)) → dst(15–0)

Статусные биты Изменяется под влиянием SXM.

Описание Данная команда загружает 32-разрядный операнд Lmem в dst. Значение C16 определяет режим исполнения команды:
Если C16 = 0, команда выполняется в режиме двойной точности. Lmem загружается в dst.
Если C16 = 1, команды выполняется в двойном 16-разрядном режиме. 16 самых старших битов Lmem загружаются в старшие 24 бита dst. В то же время, 16 самых младших битов Lmem загружаются в 16 младших битов dst.
Слова 1 слово.

Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem.

Циклы 1 цикл.
Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem.

Классы Класс 9А.
Класс 9В.

Пример DLD *AR3+, B

Перед командой			После команды		
B	00	0000 0100	B	00	6CAC 0201
AR3		0100	AR3†		0102
Память данных					
0100h		6CAC	0100h		6CAC
0101h		BD90	0101h		BD90

* Данная команда является командой с чтением длинного операнда, потому после выполнения AR3 увеличивается на 2.

20.3.28 DRSUB Lmem, src

Операнды Lmem:операнд памяти данных, использующий адресацию длинного слова.
src: A (аккумулятор A).
B (аккумулятор B).

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	1	0	1	1	0	0	S	I	A	A	A	A	A	A	A

Выполнение If C16 = 0
Then
(Lmem) - (src) → src
Else
(Lmem(31–16)) - (src(31–16)) → src(39–16)
(Lmem(15–0)) - (src(15–0)) → src(15–0)

Статусные биты Результат зависит от SXM и OVM (только если C16 = 0).
Операция влияет на C и OVsrc.

Описание Данная команда вычитает содержимое src из 32-битного операнда Lmem и сохраняет результат в src. Значение C16 определяет режим команды:
Если C16 = 0, команда выполняется в режиме двойной точности. Содержимое src (32 бита) вычитается из Lmem. Результат сохраняется в src.
Если C16 = 1, команда выполняется в двойном 16-битном режиме. Старшая часть src (биты 31–16) вычитается из 16 самых старших битов операнда Lmem и результат сохраняется в старшей части src (биты 39–16). В то же время, младшая часть src (биты 15–0) вычитаются из 16 самых младших битов операнда Lmem. Результат сохраняется в младшей части src (биты 15–0). В этом режиме результаты являются ненасыщенными, независимо от состояния бита OVM.

Слова 1 слово.
Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem.

Циклы 1 цикл.
Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem.

Классы Класс 9A.
Класс 9B.

Пример 1 DRSUB *AR3+, A

Перед командой			После команды		
A	00	5678 8933	A	FF	BEVB AB23
T		x	T		0
C16		0	C16		0
AR3		0100	AR3†		0102
Память данных					
0100h		1534	0100h		1534
0101h		3456	0101h		3456

* Данная команда является командой обработки длинного операнда и после выполнения AR3 увеличивается на 2.

Пример 2 DRSUB *AR3–, A

Перед командой				После команды			
A	00	5678	3933	A	FF	BEBC	FB23
T				T	0		
C16	1			C16	1		
AR3	0100			AR3†	00FE		
Память данных							
0100h	1534			0100h	1534		
0101h	3456			0101h	3456		

* Данная команда является командой обработки длинного операнда и после выполнения AR3 уменьшается на 2.

20.3.29 DSADT Lmem, dst

Операнды	Lmem: операнд памяти данных, использующий адресацию длинного слова. dst: A (аккумулятор A). B (аккумулятор B).																																
Код операции	<table><tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>D</td><td>I</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td></tr></table>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	0	1	0	1	1	1	1	D	I	A	A	A	A	A	A	A
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																		
0	1	0	1	1	1	1	D	I	A	A	A	A	A	A	A																		
Выполнение	If C16 = 1 Then (Lmem(31–16)) - (T) → dst(39–16) (Lmem(15–0)) + (T) → dst(15–0) Else (Lmem) – ((T) + (T << 16)) → dst																																
Статусные биты	Результат зависит от SXM и OVM (только если C16 = 0) Команда влияет на C и OVdst.																																
Описание	Данная команда вычитает/складывает содержимое T из 32-битного операнда Lmem и сохраняет результат в dst. Значение C16 определяет режим исполнения команды: Если C16 = 0, команда выполняется в режиме двойной точности. 32-разрядное значение, состоящее из содержимого T, сочлененному с содержимым того же регистра T сдвинутого влево на 16 разрядов (T << 16 + T) вычитается из Lmem. Результат сохраняется в dst. Если C16 = 1, команда выполняется в двойном 16-разрядном режиме. Содержимое T вычитается из 16 самых старших битов операнда Lmem и результат сохраняется в dst (биты 39–16). В то же самое время содержимое T добавляется к 16 самым младшим битам операнда Lmem и результат сохраняется в младшей части dst (биты 15–0). В этом режиме результаты являются ненасыщенными, независимо от состояния бита OVM. Примечание – Данная команда имеет смысл только, если установлен бит C16 (двойной 16-битный режим).																																
Слова	1 слово. Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem.																																

Циклы	1 цикл. Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem.
Классы	Класс 9А. Класс 9В.
Пример 1	DSADT *AR3+, A

Перед исполнением				После исполнения			
A	00	0000	0000	A	FF	F1EF	1111
T				T	2345		
C				C	0		
C16				C16	0		
AR3				AR3†	0102		
Память данных							
0100h				0100h	1534		
0101h				0101h	3456		

* Данная команда является командой обработки длинного операнда и после выполнения AR3 увеличивается на 2.

Пример 2	DSADT *AR3-, A

Перед командой				После команды			
A	00	0000	0000	A	FF	F1EF	1111
T	2345			T	2345		
C	0			C	1		
C16	1			C16	1		
AR3	0100			AR3†	00FE		
Память данных							
0100h	1534			0100h	1534		
0101h	3456			0101h	3456		

* Данная команда является командой обработки длинного операнда и после выполнения AR3 уменьшается на 2.

20.3.30 DST src, Lmem

Операнды	src: A (аккумулятор A). B (аккумулятор B). Lmem операнд памяти данных, использующий адресацию длинного слова.															
Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	1	0	0	1	1	1	S	I	A	A	A	A	A	A	A
Выполнение	(src(31–0)) → Lmem															
Статусные биты	Не используются.															
Описание	Данная команда сохраняет содержимое src в 32-разрядной ячейке памяти данных Lmem.															

Слова	1 слово. Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem.
Циклы	1 цикл. Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem.
Классы	Класс 13А. Класс 13В.
Пример 1	DST B, *AR3+

Перед командой				После команды			
В	00	6CAC	BD90	В	00	6CAC	BD90
AR3				AR3↑			
Память данных							
0100h				0100h			
0101h				0101h			

* Данная команда является командой обработки длинного операнда и после выполнения AR3 увеличивается на 2.

Пример 2 DST B, *AR3–

Перед командой				После команды			
В	00 6CAC BD90			В	00 6CAC BD90		
AR3	0101			AR3↑	00FF		
Память данных							
0100h	0000			0100h	BD90		
0101h	0000			0101h	6CAC		

* Данная команда является командой обработки длинного операнда и после выполнения AR3 увеличивается на 2.

20.3.31 DSUB Lmem, src

Операнды	Lmem: операнд памяти данных, использующий адресацию длинного слова. src: A (аккумулятор A). B (аккумулятор B).																																
Код операции	<table><tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>S</td><td>I</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td></tr></table>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	0	1	0	1	0	1	0	S	I	A	A	A	A	A	A	A
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																		
0	1	0	1	0	1	0	S	I	A	A	A	A	A	A	A																		
Выполнение	If C16 = 0 Then (src) - (Lmem) → src Else (src(31–16)) - (Lmem(31–16)) → src(39–16) (src(15–0)) - (Lmem(15–0)) → src(15–0)																																
Статусные биты	Результат зависит от SXM и OVM (только если C16 = 0). Команда влияет на C и OVsrc.																																

Описание	Данная команда вычитает 32-разрядный операнд Lmem из содержимого src и сохраняет результат в src. Значение C16 определяет режим команды: если C16 = 0, команда выполняется в режиме двойной точности. Lmem вычитается из содержимого src. если C16 = 1, команда выполняется в двойном 16-битном режиме. 16 самых старших битов операнда Lmem вычитаются из старшей части src (биты 31-16) и результат сохраняется в старшей части src (биты 39-16). В то же самое время, 16 самых младших разрядов операнда Lmem вычитаются из младшей части src (биты 15-0) и результат сохраняется в младшей части src (биты 15-0).
Слова	1 слово. Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem.
Циклы	1 цикл. Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem.
Классы	Класс 9A. Класс 9B.
Пример 1	DSUB *AR3+, A

Перед командой				После команды			
A	00	5678	8933	A	00	4144	54DD
C16	0			T	0		
AR3	0100			AR3↑	0102		
Память данных							
0100h	1534			0100h	1534		
0101h	3456			0101h	3456		

* Поскольку данная команда является командой с длинным операндом, после выполнения AR3 увеличивается на 2.

Пример 2 DSUB *AR3-, A

Перед командой				После команды			
A	00	5678	3933	A	00	4144	04DD
C				C			
C16				C16			
AR3				AR3†			
Память данных							
0100h				0100h			
0101h				0101h			

* Поскольку данная команда является командой с длинным операндом, после выполнения AR3 увеличивается на 2.

20.3.32 DSUBT Lmem, dst

Операнды Lmem: операнд памяти данных, использующий адресацию длинного слова.
dst: A (аккумулятор A).
B (аккумулятор B).

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	1	0	1	1	1	0	D	I	A	A	A	A	A	A	A

Выполнение If C16 = 1
Then
 $(Lmem(31-16)) - (T) \rightarrow dst(39-16)$
 $(Lmem(15-0)) - (T) \rightarrow dst(15-0)$
Else
 $(Lmem) - ((T) + (T \ll 16)) \rightarrow dst$

Статусные биты Результат зависит от SXM и OVM (только если C16 = 0).
Инструкция влияет на C и OVdst.

Описание Слова Данная команда вычитает содержимое T из 32-разрядного операнда Lmem.
1 слово.

Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem.

Циклы 1 цикл.
Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Lmem.

Классы Класс 9A.
Класс 9B.

Пример 1 DSUBT *AR3+, A

A	00 0000 0000	A	FF F1EF 1111
T	2345	T	2345
C16	0	C16	0
AR3	0100	AR3†	0102
Память данных			
0100h	1534	0100h	1534
0101h	3456	0101h	3456

* Поскольку данная команда является командой с длинным операндом, после выполнения AR3 увеличивается на 2.

Пример 2 DSUBT *AR3-, A

Перед исполнением		После исполнения	
A	00 0000 0000	A	FF F1EF 1111
T	2345	T	2345
C16	1	C16	1
AR3	0100	AR3†	00FE
Память данных			
0100h	1534	0100h	1534
0101h	3456	0101h	3456

* Поскольку данная команда является командой с длинным операндом, после выполнения AR3 уменьшается на 2.

20.3.33 EXP src

Операнды src: A (аккумулятор A).
B (аккумулятор B).

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	0	S	1	0	0	0	1	1	1	0

Выполнение If (src) = 0
Then
0 → T
Else
(количество ведущих битов в src) - 8 → T

Статусные биты Не используются.

Описание Команда рассчитывает значение экспоненты, которое является значением со знаком в двоичном дополнительном коде в диапазоне от -8 до 31, и сохраняет результат в T. Экспонента вычисляется путем подсчета количества ведущих бит в src и вычитанием 8 из этого значения. Количество ведущих бит эквивалентно количеству сдвигов влево, необходимых для того, чтобы исключить незначимые биты из 40-разрядного src за исключением знакового бита. src не изменяется после выполнения команды.
Результат вычитания 8 из количества ведущих битов вызывает появление отрицательной степени для значений аккумулятора, которые имеют значимые биты в чисте «сторожевых» разрядов (8 самых старших разрядов аккумулятора, используемых для обнаружения ошибки и исправления). См. инструкцию нормализации.

Слова 1 слово.

Циклы 1 цикл.

Классы Класс 1

Пример 1 EXP A

Перед исполнением					После исполнения				
A	FF	FFFF	FFCB	-53	A	FF	FFFF	FFCB	-53
T	FC18				T	0019			25

Пример 2 EXP B

Перед исполнением				После исполнения				
B	07	8543	2105	B	07	8543	2105	
T	FFFC			T	FFFC			-4

* Значение аккумулятора B среди «сторожевых» битов имеет значимые разряды, что приводит к отрицательному показателю.

20.3.34 FB[D] extpmad

Операнды $0 \leq \text{extpmad} \leq 7F\text{ FFFF}$

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	1	0	Z	0	1	7 БИТ ПОСТОЯННЫЙ = pmad(22-16)						
	16 битная константа = pmad(15-0)															

Выполнение (pmad(15–0)) → PC

(pmad(22–16)) → XPC

Статусные биты Не используются.

Описание Данная команда передает управление по адресу памяти программ pmad (биты 15–0) на страницу, определенную по pmad (биты 22–16). Адрес pmad может быть задан и как число и символически. Если переход с задержкой (определяется по наличию индекса D), то две команды по одному слову или одна команда из двух слов, следующие за командой перехода, вызываются из памяти программ и выполняются.

Примечания – Данная команда не повторяемая.

В данной модели процессора операция не реализована!

Слова 2 слова.

Циклы 4 цикла.

2 цикла (с задержкой).

Классы Класс 29A.

Пример 1 FB 012000h

Перед командой		После команды	
PC	1000	PC	2000
XPC	00	XPC	7F

2000h загружается в PC, 01h загружаются в XPC и программа продолжает выполнение с этой ячейки.

Пример 2

FBD 7F1000h

ANDM #4444h, *AR1+

Перед командой		После команды	
PC	2000	PC	1000
XPC	00	XPC	7F

После того, как в операнд логически умножается (AND) на значение 4444h, программа продолжает выполнение с ячейки 1000h на странице 7Fh.

20.3.35 FBACC[D] src

Операнды src: A (аккумулятор A)
B (аккумулятор B)

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	Z	S	1	1	1	0	0	1	1	0

Выполнение (src(15–0)) → PC

(src(22–16)) → XPC

Статусные биты	Не используются.
Описание	Данная команда загружает в XPC значение src (биты 22–16) и передает управление в 16-разрядный адрес младшей части src (биты 15–0). Если переход задержанный (определяется по наличию индекса D), то две команды по одному слову или одна команда из двух слов, следующие за командой перехода, вызываются из памяти программ и выполняются. Примечания – Данная команда не повторяемая. В данной модели процессора операция не реализована!
Слова	1 слово.
Циклы	6 циклов. 4 цикла (с задержкой).
Классы	Класс 30A.
Пример 1	FBACC A

Перед командой				После команды					
A	00	0001	3000	A	00	0001	3000		
PC				1000	PC				3000
XPC				00	XPC				01

1h загружается в XPC, 3000h загружается в PC, и программа продолжает выполнение с этой ячейки на странице 1h.

Пример 2	FBACCD B ANDM 4444h *AR1+

Перед командой				После команды			
B	00	007F	2000	B	00	007F	2000
XPC	01			XPC	7F		

После того, как в операнд логически умножается (AND) на значение 4444h, 7Fh загружается в XPC, и программа продолжает выполнение с ячейки 2000h на странице 7Fh.

20.3.36 FCALA[D] src

Операнды	src: A (аккумулятор A) B (аккумулятор B)

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	Z	S	1	1	1	0	0	1	1	1

Выполнение	Без задержки: (SP) – 1 → SP; (PC) + 1 → TOS; (SP) – 1 → SP; (XPC) → TOS; (src(15–0)) → PC; (src(22–16)) → XPC.

С задержкой:
(SP) – 1 → SP;
(PC) + 3 → TOS;
(SP) – 1 → SP;
(XPC) → TOS;
(src(15–0)) → PC;
(src(22–16)) → XPC.

Статусные биты

Не используются.

Описание

Данная команда загружает в XPC значение src (биты 22–16) и передает управление в 16-разрядный адрес младшей части src (биты 15–0). Если вызов задержанный (определяется по наличию индекса D), то две команды с одним словом или одна команда с двумя словами, следующие за командой вызова, вызываются из памяти программ и выполняются.

Примечание – Данная команда не повторяемая.

В данной модели процессора операция не реализована!

Слова

1 слово.

Циклы

6 циклов.
4 цикла (с задержкой).

Классы

Класс 30В.

Пример 1

FCALA A

Перед выполнением				После выполнения			
A	00	007F	3000	A	00	007F	3000
PC	0025			PC	3000		
XPC	00			XPC	7F		
SP	1111			SP	110F		
Память данных							
1110h	4567			1110h	0026		
110Fh	4567			110Fh	0000		

Пример 2

FCALAD B
ANDM #4444h, *AR1+

Перед командой				После команды			
A	00	0020	2000	A	00	0020	2000
PC	0025			PC	2000		
XPC	7F			XPC	20		
SP	1111			SP	110F		
Память данных							
1110h	4567			1110h	0028		

После того, как ячейка памяти логически умножается (AND) на значение 4444h, программа продолжает выполнение с ячейки 2000h на странице 20h.

20.3.37 FCALL[D] extpmad

Операнды	$0 \leq \text{extpmad} \leq 7F\text{ FFFF}$															
Код	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
операции	1	1	1	1	1	0	Z	1	1	7 БИТ ПОСТОЯННЫЙ = pmad(22-16)						
	16 битная константа = pmad(15-0)															

Выполнение Без задержки:

(SP) - 1 → SP;
(PC) + 2 → TOS;
(SP) - 1 → SP;
(XPC) → TOS;
(pmad(15-0)) → PC;
(pmad(22-16)) → XPC.

С задержкой:

(SP) - 1 → SP;
(PC) + 4 → TOS;
(SP) - 1 → SP;
(XPC) → TOS;
(pmad(15-0)) → PC;
(pmad(22-16)) → XPC.

Статусные биты Не используются.

Описание Данная команда передает управления по указанному адресу памяти программ pmad (биты 15-0) на странице, определяемой pmad (биты 22-16). Адрес возврата записывается в стек до загрузки pmad в PC. Если вызов задержанный (определяется по наличию индекса D), то две команды по одному слову или одна команда из двух слов, следующие за командой вызова, извлекаются из памяти программ и выполняются.

Примечание – Данная команда не повторяемая.

В данной модели процессора операция не реализована!

Слова 2 слова.

Циклы 4 цикла.

2 цикла (с задержкой).

Классы Класс 29В.

Пример 1 FCALL 013333h

Перед командой			После команды		
PC		0025	PC		3333
XPC		00	XPC		01
SP		1111	SP		110F
Память данных					
1110h		4567	1110h		0027
110Fh		4657	110Fh		0000

Пример 2 FCALLD 301000h
ANDM #4444h, *AR1+

Перед командой		После команды	
PC	3001	PC	1000
XPC	7F	XPC	30
SP	1111	SP	110F
Память данных			
1110h	4567	1110h	3005
110Fh	4657	110Fh	007F

После того, как ячейка памяти логически умножается (AND) на значение 4444h, программа продолжает выполнение с ячейки 1000h.

20.3.38	FIRS Xmem, Ymem, pmad																																																
Операнды	Xmem, Ymem: операнды памяти данных двойного доступа. 0 ≤ pmad ≤ 65 535																																																
Код операции	<table><tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>X</td><td>X</td><td>X</td><td>X</td><td>Y</td><td>Y</td><td>Y</td><td>Y</td></tr><tr><td colspan="16">16 битная константа</td></tr></table>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	1	1	1	0	0	0	0	0	X	X	X	X	Y	Y	Y	Y	16 битная константа															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																		
1	1	1	0	0	0	0	0	X	X	X	X	Y	Y	Y	Y																																		
16 битная константа																																																	
Выполнение	pmad → PAR While (RC) ≠ 0 (B) + (A(32–16)) x (Pmem адресован PAR) → B ((Xmem) + (Ymem)) << 16 → A (PAR) + 1 → PAR (RC) - 1 → RC																																																
Статусные биты	Изменяются под влиянием SXM, FRCT, и OVM. Влияет на C, OVA, и OVB.																																																
Описание	Данная команда выполняет функцию симметричного фильтра с конечной импульсной характеристикой (КИХ-фильтра). Команда умножает аккумулятор A (биты 32–16) на значение Pmem, адресованное pmad (через регистр программного адреса PAR) и складывает результат со значением в аккумуляторе B. Одновременно складывает значение операндов Xmem и Ymem, сдвигает результат влево на 16 бит, и загружает это значение в аккумулятор A. Во время следующей итерации, pmad увеличивается на 1. Как только конвейер повторений начинает работу, команда становится одноцикловой.																																																
Слова	2 слова.																																																
Циклы	3 цикла.																																																
Классы	Класс 8.																																																
Пример	FIRS *AR3+, *AR4+, COEFFS																																																

Перед выполнением				После выполнения			
A	00	0077	0000	A	00	00FF	0000
B	00	0000	0000	B	00	0008	762C
FRCT	0			FRCT	0		
AR3	0100			AR3	0101		
AR4	0200			AR4	0201		
Память данных							
0100h	0055			0100h	0055		
0200h	00AA			0200h	00AA		
Память программ							
COEFS	1234			COEFS	1534		

20.3.39	FRAME K																																
Операнды	$-128 \leq K \leq 127$																																
Код операции	<table><tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>K</td><td>K</td><td>K</td><td>K</td><td>K</td><td>K</td><td>K</td><td>K</td></tr></table>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	1	1	1	0	1	1	1	0	K	K	K	K	K	K	K	K
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																		
1	1	1	0	1	1	1	0	K	K	K	K	K	K	K	K																		
Выполнение	$(SP) + K \rightarrow SP$																																
Статусные биты	Не используются.																																
Описание	Данная команда складывает короткое непосредственное смещение K к SP. Задержка генерации адреса отсутствует в режиме компилятора (CPL = 1), когда используется указатель стека или в стековой обработке, осуществляемой командой, следующей за данной.																																
Слова	1 слово.																																
Циклы	1 цикл.																																
Классы	Класс 1.																																
Пример 1	FRAME 10h																																

Перед командой		После команды	
SP	1000	SP	1010

20.3.40 FRET[D]

Операнды Отсутствуют.

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	Z	0	1	1	1	0	0	1	0	0

Выполнение (TOS) → XPC
(SP) + 1 → SP
(TOS) → PC
(SP) + 1 → SP

Статусные биты Не используются.

Описание Данная команда заменяет значение XPC 7-разрядным значением из TOS, затем заменяет PC следующим 16-разрядным значением в стеке. SP увеличивается на 1 во время каждой из двух замен. Если возврат задержанный (определяется по наличию индекса D), то две команды по одному слову или одна команда из двух слов, следующие за этой командой, извлекаются из памяти программ и выполняются.

Примечание – Данная команда не повторяемая.

В данной модели процессора операция не реализована!

Слова 1 слово.

Циклы 6 циклов.

4 цикла (с задержкой).

Классы Класс 34.

Пример FRET

Перед командой			После команды		
PC		2112	PC		1000
XPC		01	XPC		05
SP		0300	SP		0302
Память данных					
0300h		0005	0300h		0005
0301h		1000	0301h		1000

20.3.41 FRETE[D]

Операнды Отсутствуют.

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	Z	0	1	1	1	0	0	1	0	1

Выполнение (TOS) → XPC
(SP) + 1 → SP
(TOS) → PC
(SP) + 1 → SP
0 → INTM

Статусные биты	Влияет на INTM
Описание	Данная команда заменяет значение XPC 7-разрядным значением из TOS, затем заменяет PC следующим 16-разрядным значением из стека, продолжая выполнение по новому значению PC. Эта команда автоматически устанавливает в 0 бит маски прерывания (INTM) в ST1. (сбрасывание этого бита делает возможным прерывание.) Если возврат задерживается (определяется по наличию индекса D), то две команды с одним словом или одна команда с двумя словами, следующие за этой командой, вызываются из памяти программ и выполняются.
	Примечание – Данная команда не повторяемая.
	В данной модели процессора операция не реализована!
Слова	1 слово.
Циклы	6 циклов. 4 цикла (с задержкой).
Классы	Класс 34.
Пример	FRETE

Перед командой		После команды	
PC	2112	PC	0110
XPC	05	XPC	6E
ST1	xCxx	ST1	x4xx
SP	0300	SP	0302
Память данных			
0300h	006E	0300h	006E
0301h	0110	0301h	0110

20.3.42 IDLE K

Операнды Код операции	$1 \leq K \leq 3$															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	N	N	1	1	1	0	0	0	0	1
	При K:								NN							
	1								00							
	2								10							
	3								01							

Выполнение	(PC) +1 → PC
Статусные биты	Выполнение зависит от INTM.
Описание	Данная команда задерживает выполнение программы до появления немаскированного прерывания или сброса. Значение PC увеличивается на 1. устройство остается в состоянии незанятости (режим пониженного потребления энергии) до прерывания.

Выход из состояния неактивности происходит после незамаскированного прерывания, даже если $INTM = 1$. Если $INTM = 1$, программа продолжает выполнение с команды, следующей за нерабочим состоянием. Если $INTM = 0$, программа переходит к соответствующей программе обработки прерывания. Прерывание включается регистром маски прерывания (IMR), независимо от значения $INTM$. Следующие опции, установленные значением K , определяют тип прерываний, которые могут вывести устройство из состояния неактивности:

$K = 1$ периферийные устройства, такие как таймер и последовательный порт, активны. Прерывания от периферийного устройства, а также сброс и внешние прерывания выводят процессор из состояния неактивности.

$K = 2$ периферийные устройства, такие как таймер и последовательный порт, не активны. Сброс и внешние прерывания выводят процессор из состояния неактивности. Поскольку в режиме неактивности прерывания не фиксируются в регистре-защелке, как во время нормальной работы устройства, они должны быть меньше на количество подтверждаемых циклов.

$K = 3$ периферийные устройства, такие как таймер и последовательный порт, не активны и останавливается работа PLL. Сброс и внешние прерывания выводят процессор из состояния неактивности. Поскольку в режиме неактивности прерывания не фиксируются в регистре-защелке, как во время нормальной работы устройства, они должны быть меньше на количество подтверждаемых циклов.

Примечание – Данная команда не повторяемая.

Слова 1 слово.

Циклы Число циклов, необходимое для выполнения данной команды зависит от периода бездействия. Поскольку все устройство прекращает работу при $K = 3$, то число циклов не может быть установлено. Минимальное число циклов равно 4.

Классы Класс 36.

Пример 1 IDLE 1
Процессор бездействует до возникновения немаскированного прерывания или сброса.

Пример 2 IDLE 2
Процессор бездействует до возникновения немаскированного внешнего прерывания или сброса.

Пример 3 IDLE 3
Процессор бездействует до возникновения немаскированного внешнего прерывания или сброса.

20.3.43 INTR K

Операнды $0 \leq K \leq 31$

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	1	1	1	1	0	K	K	K	K	K

Выполнение (SP) - 1 $\rightarrow$ SP
(PC) + 1 $\rightarrow$ TOS
Вектор прерывания определяемый $K \rightarrow$ PC
1 $\rightarrow$ INTM

Статусные биты	На команду влияет INTM и IFR.
Описание	Данная команда осуществляет передачу управления на программу обработки прерывания по вектору, определяемому значением К. Эта команда позволяет использовать прикладную программу по обработке прерывания. Перечень прерываний и соответствующих значений К представлен в Приложении В. Во время выполнения команды PC увеличивается на 1 и записывается в TOS. Таким образом, вектор прерывания, определенный значением К, загружается в PC и для него выполняется программа обработки прерывания. Соответствующий бит в регистре флага прерывания (IFR) сбрасывается, а все прерывания отключаются (INTM = 1). Регистр маски прерываний (IMR) не оказывает влияние на команду INTR. INTR выполняется независимо от значения INTM. Примечание – Данная команда не повторяемая.
Слова	1 слово.
Циклы	3 цикла.
Классы	Класс 35.
Пример	INTR 3

Перед выполнением		После выполнения	
PC	0025	PC	FF8C
INTM	0	INTM	1
IPTR	01FF	IPTR	01FF
SP	1000	SP	0FFF
Память данных			
0FFFh	9653	0FFFh	0026

20.3.44 LD

- 1 LD Smem, dst
- 2 LD Smem, TS, dst
- 3 LD Smem, 16, dst
- 4 LD Smem [, SHIFT], dst
- 5 LD Xmem, SHFT, dst
- 6 LD #K, dst
- 7 LD #lk [, SHFT], dst
- 8 LD #lk, 16, dst
- 9 LD src, ASM [, dst]
- 10 LD src [, SHIFT], dst

Дополнительные команды загрузки см. в разделе «Load T/DP/ASM/ARP».

Операнды

Smem: операнд памяти данных одиночного доступа.

Xmem: операнд памяти данных двойного доступа.

src, dst: A (аккумулятор A).

B (аккумулятор B).

$0 \leq K \leq 255$

$-32\,768 \leq lk \leq 32\,767$

$-16 \leq \text{SHIFT} \leq 15$

$0 \leq \text{SHFT} \leq 15$

Код

операции

1:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	0	0	D	I	A	A	A	A	A	A	A

2:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	1	0	D	I	A	A	A	A	A	A	A

3:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	1	0	D	I	A	A	A	A	A	A	A

4:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	1	1	I	A	A	A	A	A	A	A
0	0	0	0	1	1	0	D	0	1	0	S	H	I	F	T

5:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	0	1	0	D	X	X	X	X	S	H	F	T

6:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	0	0	D	K	K	K	K	K	K	K	K

7:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	D	0	0	1	0	S	H	F	T
16 битная константа															

8:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	D	0	1	1	0	0	0	1	0
16 битная константа															

9:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	S	D	1	0	0	0	0	0	1	0

10:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	S	D	0	1	0	S	H	I	F	T

Выполнение

- 1: (Smem) → dst
- 2: (Smem) << TS → dst
- 3: (Smem) << 16 → dst
- 4: (Smem) << SHIFT → dst
- 5: (Xmem) << SHFT → dst
- 6: K → dst
- 7: lk << SHFT → dst
- 8: lk << 16 → dst
- 9: (src) << ASM → dst
- 10: (src) << SHIFT → dst

Статусные биты

Выполнение зависит от SXM во всех загрузках аккумулятора.
 Выполнение зависит от OVM при сдвиге по SHIFT или ASM.
 Результат влияет на OVdst (или OVsrc, если dst = src) при загрузке со сдвигом SHIFT или ASM.

Описание

Данная команда загружает в аккумулятор (dst, или src, если dst не указан) значение памяти данных или непосредственное значение, поддерживая при этом различные величины сдвига.
 Кроме того, команда обеспечивает передачу данных между аккумуляторами со сдвигом.

Примечание – Следующие синтаксисы ассемблируются как различные синтаксис в определенных условиях.

- 1 Синтаксис 4: если SHIFT = 0, код операции команды ассемблируется по синтаксису 1.
- 2 Синтаксис 4: если $0 < \text{SHIFT} \leq 15$ и режим косвенной адресации Smem входит в Xmem, код операции команды ассемблируется как синтаксис 5.
- 3 Синтаксис 5: если SHFT = 0, код операции команды ассемблируется как синтаксис 1.
- 4 Синтаксис 7: если SHFT = 0 и $0 \leq \text{lk} \leq 255$, код операции команды ассемблируется как синтаксис 6.

Слова

Синтаксисы 1, 2, 3, 5, 6, 9, и 10: 1 слово.
 Синтаксисы 4, 7, и 8: 2 слова.
 Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Циклы

Синтаксисы 1, 2, 3, 5, 6, 9, и 10: 1 цикл.
 Синтаксисы 4, 7, и 8: 2 цикла.
 Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Классы Синтаксисы 1, 2, 3 и 5: Класс 3А.
 Синтаксисы 1, 2 и 3: Класс 3В.
 Синтаксис 4: Класс 4А.
 Синтаксис 4: Класс 4В.
 Синтаксисы 6, 9, и 10: Класс 1.
 Синтаксисы 7 и 8: Класс 2.

Пример 1 LD *AR1, A

Перед командой				После команды			
A	00	0000	0000	A	00	0000	FEDC
SXM				SXM	0		
AR1	0200			AR1	0200		
Память данных							
0200h	FEDC			0200h	FEDC		

Пример 2 LD *AR1, A

Перед командой				После команды			
A	00	0000	0000	A	FF	FFFF	FEDC
SXM	1			SXM	1		
AR1	0200			AR1	0200		

Память данных

0200h	FEDC	0200h	FEDC

Пример 3 LD *AR1, TS, B

Перед командой			
B	00	0000	0000
SXM	1		
AR1	0200		
T	8		

После команды			
B	FF	FFFE	DC00
SXM	1		
AR1	0200		
T	8		

Память данных

0200h	FEDC

0200h	FEDC

Пример 4 LD *AR3+, 16, A

Перед командой				После команды			
B	00	0000	0000	B	FF	FEDC	0000
SXM	1			SXM	1		
AR3	0300			AR3	0301		
Память данных							
0300h	FEDC			0300h	FEDC		

Пример 5 LD #248, B

	Перед командой		После команды
В	00 0000 0000	В	00 0000 00F8
SXM	1	SXM	1

Пример 6 LD A, 8, В

	Перед командой		После команды
А	00 7FFD 0040	А	00 7FF0 0040
В	00 0000 FFFF	В	7F FD00 4000
OVВ	0	OVВ	1
SXM	1	SXM	1
Память данных			
0200h	FEDC	0200h	FEDC

20.3.45	LDM MMR, dst																																
Операнды	MMR: отображаемый в памяти регистр. dst: A (аккумулятор). B (аккумулятор).																																
Код операции	<table><tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>D</td><td>I</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td></tr></table>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	0	1	0	0	1	0	0	D	I	A	A	A	A	A	A	A
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																		
0	1	0	0	1	0	0	D	I	A	A	A	A	A	A	A																		
Выполнение	(MMR) → dst(15–0) 00 0000h → dst(39–16)																																
Статусные биты	Не используются.																																
Описание	Данная команд загружает в dst значение отображаемого в памяти регистра MMR. Девять старших бит действительного адреса отбрасываются для обозначения страницы данных как 0, независимо от текущего значения DR и старших девяти бит ARx. Данная команда не зависит от значения SXM.																																
Слова	1 слово.																																
Циклы	1 цикл.																																
Классы	Класс 3A.																																
Пример 1	LDM AR4, A																																

	Перед командой		После команды
А	00 0000 1111	А	00 0000 FFFF
AR4	FFFF	AR4	FFFF

Пример 2 LDM 060h, В

	Перед командой		После команды
В	00 0000 0000	В	00 0000 1234
Память данных			
0060h	1234	0060h	1234

20.3.46 LD Xmem, dst || MAC[R] Ymem [, dst_]

Операнды	dst:	A (аккумулятор A).
		B (аккумулятор B).
	dst_:	If dst = A, then dst_ = B; if dst = B, then dst_ = A
	Xmem, Ymem:	операнды памяти данных двойного доступа

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	1	0	1	0	R	D	X	X	X	X	Y	Y	Y	Y

Выполнение	(Xmem) << 16 → dst (31–16)
	If (округление)
	Round (((Ymem) x (T)) + (dst_)) → dst_
	Else
	((Ymem) x (T)) + (dst_) → dst_

Статусные биты	Выполнение изменяется под влиянием SXM, FRCT и OVM
	Команда влияет на OVdst_ и OVdst

Описание	Данная команда загружает операнд памяти данных двойного доступа Xmem, сдвигая его влево на 16 бит в сторону старшей части dst (биты 31-16). Параллельно с эти, данная команда умножает операнд Ymem на содержимое T, складывает результат умножения с dst_ и сохраняет результат в dst_. При использовании индекса R, эта команда округляет результат умножения и выполняет операцию сложения путем добавления 2 ¹⁵ к результату и сбрасыванием значений младших битов (15–0) в 0, и сохраняет результат в dst_.

Слова	1 слово.

Циклы	1 цикл.

Классы	Класс 7.

Пример 1	LD *AR4+, A
	MAC *AR5+, B

Перед командой				После команды			
A	00	0000	1000	A	00	1234	0000
B	00	0000	1111	B	00	010C	9511
T				T	0400		
FRCT	0			FRCT	0		
AR4	0100			AR4	0101		
AR5	0200			AR5	0201		
Память данных							
0100h	1234			0100h	1234		
0200h	4321			0200h	4321		

Пример 2	LD *AR4+, A
	MACR *AR5+, B

Перед командой				После команды			
A	00	0000	1000	A	00	1234	0000
B	00	0000	1111	B	00	010D	0000
T				T	0400		
FRCT	0			FRCT	0		
AR4	0100			AR4	0101		
AR5	0200			AR5	0201		
Память данных							
0100h	1234			0100h	1234		
0200h	4321			0200h	4321		

20.3.47 LD Xmem, dst || MAS[R] Ymem [, dst_]

Операнды	Xmem, Ymem:	операнды памяти данных двойного доступа.																																														
	dst:	A (аккумулятор A). B (аккумулятор B). dst_: If dst = A, then dst_ = B; if dst = B, then dst_ = A																																														
Код операции	<table><tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>R</td><td>D</td><td>X</td><td>X</td><td>X</td><td>X</td><td>Y</td><td>Y</td><td>Y</td><td>Y</td></tr></table>																15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	1	0	1	0	1	1	R	D	X	X	X	X	Y	Y	Y	Y
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																	
1	0	1	0	1	1	R	D	X	X	X	X	Y	Y	Y	Y																																	
Выполнение	(Xmem) << 16 → dst (31–16)																																															
	If (Rounding) Round ((dst_) × ((T) _ (Ymem))) → dst_ Else (dst_) – ((T) × (Ymem)) → dst_																																															
Статусные биты	Выполнение зависит от SXM, FRCT и OVM																																															
Описание	Команда влияет на OVDst_ и OVDst																																															
	Данная команда загружает операнд памяти данных двойного доступа Xmem, сдвигая его влево на 16 бит в сторону старшей части dst (биты 31–16). Параллельно с эти, данная команда умножает операнд Ymem на содержимое T, вычитает результат умножения из dst_ и сохраняет результат в dst_. При использовании индекса R, эта команда округляет результат умножения с вычитанием путем сложения 2 ¹⁵ с результатом и сбрасыванием значений младших бит (15–0) в 0, и сохраняет результат в dst_.																																															
Слова	1 слово.																																															
Циклы	1 цикл.																																															
Классы	Класс 7.																																															
Пример 1	LD *AR4+, A																																															
	MAS *AR5+, B																																															

Перед командой				После команды				
A	00	0000	1000	A	00	1234	0000	
B	00	0000	1111	B	FF	FEF3	8D11	
T				0400	T			0400
FRCT				0	FRCT			0
AR4				0100	AR4			0101
AR5				0200	AR5			0201
Память данных								
0100h				1234	0100h			1234
0200h				4321	0200h			4321

Пример 2 LD *AR4+, A
||MASR *AR5+, B

Перед командой				После команды				
A	00	0000	1000	A	00	1234	0000	
B	00	0000	1111	B	FF	FEF4	0000	
T				0400	T			0400
FRCT				0	FRCT			0
AR4				0100	AR4			0101
AR5				0200	AR5			0201
Память данных								
0100h				1234	0100h			1234
0200h				4321	0200h			4321

20.3.48 LDR Smem, dst

Операнды Smem: операнд памяти данных одиночного доступа.
dst: A (аккумулятор A).
B (аккумулятор B).

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	1	0	1	1	D	I	A	A	A	A	A	A	A

Выполнение (Smem) << 16 + 1 << 15 → dst(31–16)

Статусные биты Выполнение зависит от SXM.

Команда влияет на OVdst

Описание Данная команда загружает значение операнда памяти, сдвинутого влево на 16 битов в старшую часть dst (биты 31–16). Smem округляется путем сложения с числом 2^{15} с этим значением и сбрасыванием 15 самых младших битов (14–0) аккумулятора. Бит 15 аккумулятора принимает значение 1.

Слова 1 слово.

Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Циклы 1 цикл.

Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Классы Класс 3A.

Класс 3B.

Пример LDR *AR1, A

Перед командой			
A	00	0000	0000
SXM	0		
AR1	0200		

После команды			
A	00	FEDC	8000
SXM	0		
AR1	0200		

Память данных

0200h	FEDC

0200h	FEDC

20.3.49 LDU Smem, dst

Операнды Smem: операнд памяти данных одиночного доступа.
dst: A (аккумулятор A).
B (аккумулятор B).

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	1	0	0	1	D	I	A	A	A	A	A	A	A

Выполнение (Smem) → dst(15–0)

00 0000h → dst(39–16)

Статусные биты Не используются.

Описание Данная команда загружает значение Smem в младшую часть dst (биты 15–0). Сторожевые биты и старшая часть dst (биты 39–16) сбрасываются в 0. Данные обрабатываются как 16-разрядное число без знака. Знак не расширяется независимо от статуса бита SXM.

Слова	1 слово. Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.
Циклы	1 цикл. Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.
Классы	Класс 3А. Класс 3В.
Пример	LDU *AR1, A

Перед командой				После команды			
A	00 0000 0000			A	00 0000 FEDC		
AR1	0200			AR1	0200		
Память данных							
0200h	FEDC			0200h	FEDC		

20.3.50 LMS Xmem, Ymem

Операнды	Xmem, Ymem: операнды памяти данных двойного доступа.															
Код	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
операции	1	1	1	0	0	0	0	1	X	X	X	X	Y	Y	Y	A
Выполнение	(A) + (Xmem) << 16 + 2 ¹⁵ → A (B) + (Xmem) × (Ymem) → B															
Статусные биты	Выполнение команды зависит от SXM, FRCT и OVM. Инструкция влияет на C, OVA и OVB.															
Описание	Данная команда выполняет алгоритм метода наименьших квадратов (LMS). Операнд Xmem сдвигается влево на 16 битов и добавляется к аккумулятору A. Результат округляется путем добавления 2 ¹⁵ к старшей части аккумулятора (биты 31–16). Результат сохраняется в аккумуляторе A. Параллельно, Xmem и Ymem перемножаются и результат добавляется к аккумулятору B. Xmem не записывает поверх T; в связи с этим, T всегда содержит значение ошибки, используемое для обновлениря коэффициента.															
Слова	1 слово.															
Циклы	1 цикл.															
Классы	Класс 7.															
Пример	LMS *AR3+, *AR4+															

Перед командой				После команды			
A	00	7777	8888	A	00	77CD	0888
B	00	0000	0100	B	00	0000	3972
FRCT	0			FRCT	0		
AR3	0100			AR3	0101		
AR4	0200			AR4	0201		
Память данных							
0100h	0055			0100h	0055		
0200h	00AA			0200h	00AA		

20.3.51 LTD Smem

Операнды Smem: операнд памяти данных одиночного значения.

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	1	0	0	1	1	0	0	I	A	A	A	A	A	A	A

Выполнение (Smem) → T
(Smem) → Smem + 1

Статусные биты Не используются.

Описание Данная команда копирует содержимое ячейки памяти данных Smem в T и по адресу ячейки памяти данных, следующей за данной. Когда данные скопированы, содержимое по адресу ячейки остается прежним. Эта функция полезна при осуществлении задержки Z в применении цифровой обработки сигнала. Эту функцию также содержит команда DELAY.

Слова 1 слово.

Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Циклы 1 цикл.

Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Классы Класс 24А.

Класс 24В.

Пример LTD *AR3

Перед командой		После команды	
T	0000	T	6CAC
AR3	0100	AR3	0100
Память данных			
0100h	6CAC	0100h	6CAC
0101h	xxxx	0101h	6CAC

20.3.52 MAC[R]

1 MAC[R] Smem, src

2 MAC[R] Xmem, Ymem, src [, dst]

3 MAC #lk, src [, dst]

4 MAC Smem, #lk, src [, dst]

Операнды Smem: операнд памяти данных одиночного доступа.
Xmem, Ymem: операнды памяти данных двойного доступа.
src, dst: A (аккумулятор A).
B (аккумулятор B).
 $-32\,768 \leq lk \leq 32\,767$

Код операции	1:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		0	0	1	0	1	0	R	S	I	A	A	A	A	A	A	A
	2:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		1	0	1	1	0	R	S	D	X	X	X	X	Y	Y	Y	Y
	3:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		1	1	1	1	0	0	S	D	0	1	1	0	0	1	1	Y
		16 битная константа															
	4:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		0	1	1	0	0	1	S	D	I	A	A	A	A	A	A	A
		16 битная константа															

Выполнение	1:	$(Smem) \times (T) + (src) \rightarrow src$
	2:	$(Xmem) \times (Ymem) + (src) \rightarrow dst$ $(Xmem) \rightarrow T$
	3:	$(T) \times lk + (src) \rightarrow dst$
	4:	$(Smem) \times lk + (src) \rightarrow dst$ $(Smem) \rightarrow T$
Статусные биты	Результат зависит от FRCT и OVM. Команда влияет на OVdst (или OVsrc, если не определен dst).	
Описание	Данная команда умножает с накоплением, с округлением или без него. Результат сохраняется в dst или src, в зависимости от того, что определено. Для синтаксисов 2 и 4, значение памяти данных после выполнения команды сохраняется в T. T обновляется в фазе чтения. При использовании индекса R, эта команда округляет результат умножения и выполняет операцию сложения путем добавления 2^{15} к результату и сбрасыванием значений младших битов (15–0) в 0.	
Слова	Синтаксисы 1 и 2: 1 слово. Синтаксисы 3 и 4: 2 слова. Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.	
Циклы	Синтаксисы 1 и 2: 1 цикл. Синтаксисы 3 и 4: 2 цикла. Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.	
Классы	Синтаксис 1: Класс 3А. Синтаксис 1: Класс 3В. Синтаксис 2: Класс 7. Синтаксис 3: Класс 2. Синтаксис 4: Класс 6А. Синтаксис 4: Класс 6В.	

Пример 5 MACR *AR5+, A

Перед командой				После команды			
A	00	0000	1000	A	00	0049	0000
T				T	0400		
FRCT	0			FRCT	0		
AR5	0100			AR5	0101		
Память данных							
0100h	1234			0100h	1234		

Пример 6 MACR *AR5+, *AR6+,A, B

Перед командой				После команды			
A	00	0000	1000	A	00	0000	1000
B	00	0000	0004	B	00	0C4C	0000
T				T	5678		
FRCT	1			FRCT	1		
AR5	0100			AR5	0101		
AR6	0200			AR6	0201		
Память данных							
0100h	5678			0100h	5678		
0200h	1234			0200h	1234		

20.3.53 MACA[R]

1 MACA[R] Smem [, B]

2 MACA[R] T, src [, dst]

Операнды Smem: операнд памяти данных одиночного доступа.
src, dst: A (аккумулятор A).
B (аккумулятор B).

Код операции

1:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	1	R	1	I	A	A	A	A	A	A	A

2:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	S	D	1	0	0	0	1	0	0	R

Выполнение 1: (Smem) × (A(32–16)) + (B) → B
(Smem) → T
2: (T) × (A(32–16)) + (src) → dst

Статусные биты Результат под влиянием FRCT и OVM.
Команда влияет на OVdst (или OVsrc, если dst не указан) и OVB в синтаксисе 1.

Описание Данная команда умножает старшую часть аккумулятора А (биты 32–16) на значение операнда Smem или на содержимое Т, добавляет результат к аккумулятору В (синтаксис 1) или к src. Результат сохраняется в аккумулятор В (синтаксис 1) или в dst или src если не указан dst. А(32–16) используется в множителе в качестве 17-разрядного операнда.

При использовании индекса R, эта команда округляет результат умножения аккумулятора А путем добавления 2^{15} к результату и сбрасыванием значений 16 младших битов dst (15–0) в 0.

Слова 1 слово.

Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Циклы 1 цикл.

Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Классы Синтаксисы 1 и 2: Класс 3А.

Синтаксисы 1 и 2: Класс 3В.

Синтаксисы 3 и 4: Класс 1.

Пример 1 MACA *AR5+

Перед командой				После команды			
A	00	1234	0000	A	00	1234	0000
B	00	0000	0000	B	00	0626	0060
T			0400	T			5678
FRCT			0	FRCT			0
AR5			0100	AR5			0101
Память данных							
0100h			5678	0100h			5678

Пример 2 MACA T, B, B

Перед командой				После команды			
A	00	1234	0000	A	00	1234	0000
B	00	0002	0000	B	00	009D	4BA0
T			0444	T			0444
FRCT			1	FRCT			1

Пример 3 MACAR *AR5+, B

Перед командой				После команды			
A	00	1234	0000	A	00	1234	0000
B	00	0000	0000	B	00	0626	0000
T			0400	T			5678
FRCT			0	FRCT			0
AR5			0100	AR5			0101
Память данных							
0100h			5678	0100h			5678

Пример 4 MACAR T, B, B

Перед командой				После команды			
A	00 1234 0000			A	00 1234 0000		
B	00 0002 0000			B	00 009D 0000		
T	0444			T	0444		
FRCT	1			FRCT	1		

20.3.54 MACD Smem, pmad, src

Операнды Smem: операнд памяти данных одиночного доступа.
src: A (аккумулятор A).
B (аккумулятор B).
 $0 \leq \text{pmad} \leq 65\,535$

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	1	1	1	1	0	1	S	I	A	A	A	A	A	A	A
	16 битная константа															

Выполнение pmad → PAR
If (RC) ≠ 0
Then
 (Smem) x (Pmem адресованный через PAR) + (src) → src
 (Smem) → T
 (Smem) → Smem + 1
 (PAR) + 1 → PAR
Else
 (Smem) x (Pmem адресованный через PAR) + (src) → src
 (Smem) → T
 (Smem) → Smem + 1

Статусные биты Выполнение зависит от FRCT и OVM.
Команда влияет на OVsrc.

Описание Данная команда умножает значение Smem на значение pmad, добавляет произведение к src, и сохраняет результат в src. Значение Smem копируется в T и в ячейку по адресу, следующему за адресом Smem. При повторении этой команды, адрес памяти программ (в регистре адреса программ PAR) увеличивается на 1. Как только конвейер повторений начинает работу, команда становится одноцикловой. Данная функция включает в себя команду DELAY.

Слова 2 слова.
Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Циклы 3 цикла.
Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Классы Класс 23А.
Класс 23В.

Пример MACSU *AR4+, *AR5+, A

Перед командой			
A	00	0000	1000
T	0008		
FRCT	0		
AR4	0100		
AR5	0200		

После команды			
A	00	09A0	AA84
T	8765		
FRCT	0		
AR4	0101		
AR5	0201		

Память данных

0100h	8765
0200h	1234

0100h	5678
0200h	00AA

20.3.57 MAR Smem

Операнды Smem: операнд памяти данных одиночного доступа.

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	1	1	0	1	1	0	1	I	A	A	A	A	A	A	A

Выполнение В режиме косвенной адресации, вспомогательный регистр изменяется следующим образом:

If compatibility is on (CMPT = 1), then:

If (ARx = AR0)
AR(ARP) is modified
ARP is unchanged
Else
ARx is modified
x → ARP

Else compatibility is off (CMPT = 0)

ARx is modified
ARP is unchanged

Статусные биты Выполнение зависит от CMPT.

Команда влияет на ARP (если CMPT = 1).

Описание Данная команда изменяет содержимое выбранного вспомогательного регистра (ARx) определенного по Smem. В режиме совместимости (CMPT = 1), данная команда изменяет содержание ARx и значение указателя вспомогательного регистра (ARP).

Если CMPT = 0, то вспомогательный регистр изменяется, а ARP - нет.

Слова 1 слово.

Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Циклы 1 цикл.

Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Классы Класс 1.
Класс 2.

Пример 1 MAR *AR3+

Перед командой		После команды	
CMRT	0	CMRT	1
ARP	0	ARP	3
AR3	0100	AR3	0101

Пример 2 MAR *AR0–

Перед командой		После команды	
CMRT	1	CMRT	1
ARP	4	ARP	4
AR4	0100	AR4	00FF

Пример 3 MAR *AR3

Перед командой		После команды	
CMPT	1	CMPT	1
ARP	0	ARP	3
AR0	0008	AR0	0008
AR3	0100	AR3	0100

Пример 4 MAR *+AR3

Перед командой		После команды	
CMRT	1	CMRT	1
ARP	0	ARP	3
AR3	0100	AR3	0101

Пример 5 MAR *AR3–

Перед командой		После команды	
CMRT	1	CMRT	1
ARP	0	ARP	3
AR3	0100	AR3	00FF

20.3.58 MAS[R]

1 MAS[R] Smem, src
2 MAS[R] Xmem, Ymem, src [, dst]

Операнды	Smem:	операнд памяти данных одиночного доступа.															
	Xmem, Ymem:	операнды памяти данных двойного доступа.															
	src, dst:	A (аккумулятор A).															
		B (аккумулятор B).															
Код операции	1:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		0	0	1	0	1	1	R	S	I	A	A	A	A	A	A	A
	2:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		1	0	1	1	1	R	S	D	X	X	X	X	Y	Y	Y	Y
Выполнение	1:	(src) - (Smem) × (T) → src															
	2:	(src) - (Xmem) × (Ymem) → dst (Xmem) → T															
Статусные биты	Результат зависит от FRCT и OVM.																
Описание	Команда влияет на OVdst (или OVsrc, если dst = src).																
	Данная команда умножает операнд на содержание T или перемножает два операнда, вычитает результат из src, если не установлен dst, и сохраняет результат в src или dst. Xmem загружается в T в фазе чтения.																
	При использовании индекса R, эта команда округляет результат умножения и вычитания путем добавления 2 ¹⁵ к результату и сбрасыванием значений битов 15–0 в 0.																
	Данные, адресованные Xmem, передаются по шине DB и данные, адресованные Ymem, передаются по CB.																
Слова	1 слово.																
	Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.																
Циклы	1 цикл.																
	Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.																
Классы	Синтаксис 1: Класс 3А.																
	Синтаксис 1: Класс 3В.																
	Синтаксис 2: Класс 7.																
Пример 1	MAS *AR5+, A																

Перед командой			После команды		
A	00	0000 1000	A	FF	FFB7 4000
T		0400	T		0400
FRCT		0	FRCT		0
AR5		0100	AR5		0101
Память данных					
0100h		1234	0100h		1234

Пример 2 MAS *AR5+, *AR6+, A, B

Перед командой				После команды			
A	00	0000	1000	A	00	0000	1000
B	00	0000	0004	B	FF	F9DA	0FA0
T	0008			T	5678		
FRCT	1			FRCT	1		
AR5	0100			AR5	0101		
AR6	0200			AR6	0201		
Память данных							
0100h	5678			0100h	5678		
0200h	1234			0200h	1234		

Пример 3 MASR *AR5+, A

Перед командой				После команды			
A	00	0000	1000	A	FF	FFB7	0000
T	0400			T	0400		
FRCT	0			FRCT	0		
AR5	0100			AR5	0101		
Память данных							
0100h	1234			0100h	1234		

Пример 4 MASR *AR5+, *AR6+, A, B

Перед командой				После команды			
A	00	0000	1000	A	00	0000	1000
B	00	0000	0004	B	FF	F9DA	0000
T	0008			T	5678		
FRCT	1			FRCT	1		
AR5	0100			AR5	0101		
AR6	0200			AR6	0201		
Память данных							
0100h	5678			0100h	5678		
0200h	1234			0200h	1234		

20.3.59 MASA

1 MASA Smem [, B]

2 MASA[R] T, src [, dst]

Операнды Smem: операнд памяти данных одиночного доступа.
src, dst: A (аккумулятор A).
B (аккумулятор B).

Код операции	1:															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	1	1	0	0	1	1	I	A	A	A	A	A	A	A
2:																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	S	D	1	0	0	0	1	0	1	R

Выполнение 1: (B) - (Smem) × (A(32–16)) → B
(Smem) → T

2: (src) - (T) × (A(32–16)) → dst

Статусные биты На результат влияет FRCT и OVM.

Описание Команда влияет на OVdst (или OVsrc, если dst не указан) и OVB в синтаксисе 1.
Данная команда умножает старшую часть аккумулятора A (биты 32–16) на операнд Smem или на содержимое T, вычитает результат из аккумулятора B (синтаксис 1) или из src. Результат сохраняется в аккумуляторе B (синтаксис 1) или в dst или src, если не указан dst. T принимает значение Smem в фазе чтения. При использовании индекса R в синтаксисе 2, данная команда округляет результат умножения на аккумулятор A и вычитания путем добавления 2¹⁵ к результату и сбрасыванием значений битов 15–0.

Слова 1 слово.

Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Циклы 1 цикл.

Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Классы Синтаксис 1 Класс 3A.

Синтаксис 1 Класс 3B.

Синтаксис 2 Класс 1.

Пример 1 MASA *AR5+

Перед командой				После команды			
A	00	1234	0000	A	00	1234	0000
B	00	0002	0000	B	FF	F9DB	FFA0
T			0400	T			5678
FRCT			0	FRCT			1
AR5			0100	AR5			0101
Память данных							
0100h			5678	0100h			5678

Пример 2 MASA T, B

Перед командой				После команды			
A	00 1234 0000			A	00 1234 0000		
B	00 0002 0000			B	FF FF66 B460		
T	0444			T	0444		
FRCT	1			FRCT	1		

Пример 3 MASA T, B

Перед командой				После команды			
A	00	1234	0000	A	00	1234	0000
B	00	0002	0000	B	FF	FF67	0000
T	0444			T	0444		
FRCT	1			FRCT	1		

20.3.60 MAX dst

Операнды dst: A (аккумулятор A).
B (аккумулятор B).

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	0	D	1	0	0	0	0	1	1	0

Выполнение If (A > B)
Then
 (A) → dst
 0 → C
Else
 (B) →dst
 1 → C

Статусные биты Команда устанавливает C.

Описание Данная команда сравнивает содержание аккумуляторов и сохраняет максимальное значение в dst. Если максимальное значение принадлежит аккумулятору A, то бит переноса, C, принимает значение 0; в противном случае принимает значение 1.

Слова 1 слово.

Циклы 1 цикл.

Классы Класс 1.

Пример 1 MAX A

Перед командой			После команды		
A	FFF6	-10	A	FFF6	-10
B	FFCB	-53	B	0035	-53
C	1		C	0	

Пример 2 MAX A

Перед командой				После команды			
A	00	0000	0055	A	00	0000	1234
B	00	0000	1234	B	00	0000	1234
C	0			C	1		

20.3.61 MIN dst

Операнды dst: A (аккумулятор A).
B (аккумулятор B).

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	0	D	1	0	0	0	0	1	1	1

Выполнение If (A < B)
Then
(A) → dst
0 → C
Else
(B) → dst
1 → C

Статусные биты Команда устанавливает C.

Описание Данная команда сравнивает содержимое аккумуляторов и сохраняет минимальное значение в dst. Если минимальное значение принадлежит аккумулятору A, то бит переноса, C, принимает значение 0; иначе-1.

Слова 1 слово.

Циклы 1 цикл.

Классы Класс 1.

Пример 1 MIN A

Перед командой			После команды		
A	FFCB	-53	A	FFCB	-53
B	FFF6	-10	B	FFF6	-10
C	1		C	0	

Пример 2 MIN A

Перед командой			После команды		
A	00 0000 1234		A	00 0000 1234	
B	00 0000 1234		B	00 0000 1234	
C	0		C	1	

20.3.62 MPY[R]

1 MPY[R] Smem, dst

2 MPY Xmem, Ymem, dst

3 MPY Smem, #lk, dst

4 MPY #lk, dst

Операнды Smem: операнд памяти данных одиночного доступа.
Xmem, Ymem: операнды памяти данных двойного доступа.
dst: A (аккумулятор A).
B (аккумулятор B).
 $-32\,768 \leq lk \leq 32\,767$

Код операции	1:	<table><tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>R</td><td>D</td><td>I</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td></tr></table>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	0	0	1	0	0	0	R	D	I	A	A	A	A	A	A																
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																	
	0	0	1	0	0	0	R	D	I	A	A	A	A	A	A																																		
	2:	<table><tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>D</td><td>X</td><td>X</td><td>X</td><td>X</td><td>Y</td><td>Y</td><td>Y</td><td>Y</td></tr></table>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	1	0	1	0	0	1	0	D	X	X	X	X	Y	Y	Y	Y															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																	
1	0	1	0	0	1	0	D	X	X	X	X	Y	Y	Y	Y																																		
3:	<table><tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>D</td><td>I</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td></tr><tr><td colspan="16">16 битная константа</td></tr></table>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	0	1	1	0	0	0	1	D	I	A	A	A	A	A	A	A	16 битная константа															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																		
0	1	1	0	0	0	1	D	I	A	A	A	A	A	A	A																																		
16 битная константа																																																	
4:	<table><tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>D</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td colspan="16">16 битная константа</td></tr></table>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	1	1	1	1	0	0	0	D	0	1	1	0	0	1	1	0	16 битная константа															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																		
1	1	1	1	0	0	0	D	0	1	1	0	0	1	1	0																																		
16 битная константа																																																	
Выполнение	1: (T) × (Smem) → dst 2: (Xmem) × (Ymem) → dst (Xmem) → T 3: (Smem) × lk → dst (Smem) → T 4: (T) × lk → dst																																																
Статусные биты	Выполнение команды зависит от FRCT и OVM. Команда влияет на OVdst.																																																
Описание	Данная команда умножает содержимое T или значение памяти данных на значение памяти данных или на непосредственное значение, и сохраняет результат в dst. T загружается значением Smem или Xmem в фазе чтения. При использовании индекса R, данная команда округляет результат операции умножения путем добавления 2 ¹⁵ к результату с последующим сбрасыванием битов 15–0.																																																
Слова	Синтаксисы 1 и 2: 1 слово. Синтаксисы 3 и 4: 2 слова. Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.																																																
Циклы	Синтаксисы 1 и 2: 1 цикл. Синтаксисы 3 и 4: 2 цикла. Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.																																																
Классы	Синтаксис 1: Класс 3А. Синтаксис 1: Класс 3В. Синтаксис 2: Класс 7. Синтаксис 3: Класс 6А. Синтаксис 3: Класс 6В. Синтаксис 4: Класс 2.																																																

Пример 1 МРУ 13, А

Перед командой				После команды			
A	00	0000	0036	A	00	0000	0054
T	0036			T	0006		
FRCT	1			FRCT	1		
DP	008			DP	008		
Память данных							
040Dh	0007			040Dh	0007		

Пример 2 МРУ *AR2-, *AR4+0%, В;

Перед командой			
B	FF	FFFF	FFE0
FRCT	0		
AR0	0001		
AR2	01FF		
AR4	0300		
Память данных			
01FFh	0010		
0300h	0002		

После команды			
B	00	0000	0020
FRCT	0		
AR0	0001		
AR2	01FE		
AR4	0301		
Память данных			
01FFh	0010		
0300h	0002		

Пример 3 МРУ #0FFFEh, А

Перед командой				После команды			
A	00	0000	1234	A	FF	FFFF	C000
T	2000			T	2000		
FRCT	0			FRCT	0		

Пример 4 МРУ R 0, В

Перед командой				После команды			
B	FF	FE00	0001	B	00	0626	0000
T	1234			T	1234		
FRCT	0			FRCT	0		
DP	004			DP	004		
Память программ							
0200h	5678			0200h	5678		

20.3.63 МРУА

1 МРУА Smem

2 МРУА dst

Операнды Smem: операнд памяти данных одиночного действия.
dst: A (аккумулятор A).
B (аккумулятор B).

Код операции

1:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	1	0	0	0	1	I	A	A	A	A	A	A	A

2:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	0	D	1	0	0	0	1	1	0	0

Выполнение

1: (Smem) × (A(32–16)) → B
(Smem) → T

2: (T) × (A(32–16)) → dst

Статусные биты

Изменяется под влиянием FRCT и OVM.

Влияет на OVdst (OVb в синтаксисе 1).

Описание

Данная команда умножает старшую часть аккумулятора A (биты 32–16) на значение операнда Smem или на содержимое T, и сохраняет результат в dst или аккумуляторе B. T обновляется в фазе чтения.

Слова

1 слово.

Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Циклы

1 цикл.

Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Классы

Синтаксис 1: Класс 3A.

Синтаксис 1: Класс 3B.

Синтаксис 2: Класс 1.

Пример 1

MPYA *AR2

Перед командой			
A	FF	8765	1111
B	00	0000	0320
T	1234		
FRCT	0		
AR2	0200		

После команды			
A	FF	8765	1111
B	FF	D743	6558
T	5678		
FRCT	0		
AR2	0200		

Память данных

0200h	5678

0200h	5678

Пример 2

MPYA B

Перед командой				После команды			
A	FF	8765	1111	A	FF	8765	1111
B	00	0000	0320	B	FF	DF4D	B2A3
T				4567	T		
FRCT				0	FRCT		

20.3.64 MPYU Smem, dst

Операнды Smem: операнд памяти данных одиночного доступа.
dst: A (аккумулятор A).
B (аккумулятор B).

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	1	0	0	1	0	D	I	A	A	A	A	A	A	A

Выполнение unsigned(T) x unsigned(Smem) → dst

Статусные биты Результат зависит от FRCT и OVM.

Команда влияет на OVdst.

Описание Данная команда умножает содержимое T как числа без знака на содержимое операнда Smem без знака, и сохраняет результат в dst. В данной команде умножитель действует как умножитель 17х17-бит со сброшенными самыми старшими битами обоих операндов. Эта команда особенно применяется для расчета произведений многократной точности, таких, как умножение 32-разрядных чисел и получением 64-разрядного произведения.

Слова 1 слово.

Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Циклы 1 цикл.

Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Классы Класс 3А.

Класс 3В.

Пример MPYU *AR0-, A

Перед выполнением				После выполнения			
A	FF	8000	0000	A	00	3F80	0000
T			4000	T			0400
FRCT			0	FRCT			0
AR0			1000	AR0			0FFF
Память данных							
1000h			FE00	0100h			FE00

20.3.65 MVDD Xmem, Ymem

Операнды Xmem, Ymem: операнды памяти данных двойного доступа.

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	0	0	1	0	1	X	X	X	X	Y	Y	Y	Y

Выполнение (Xmem) → Ymem

Статусные биты Не используются.

Описание Данная команда копирует содержание ячейки памяти данных, адресованной операндом Xmem, в ячейку памяти данных, адресованную операндом Ymem.

Слова 1 слово.
Циклы 1 цикл.
Классы Класс 14.
Пример MVDD *AR3+, *AR5+

Перед командой		После команды	
AR3	8000	AR3	8001
AR5	0200	AR5	0201
Память данных			
0200h	ABCD	0200h	1234
8000h	1234	8000h	1234

20.3.66 MVDK Smem, dmad

Операнды Smem: операнд памяти данных одиночного доступа.
 $0 \leq dmad \leq 65\,535$

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	1	1	1	0	0	0	1	I	A	A	A	A	A	A	A
16 битная константа																

Выполнение (dmad) → EAR
If (RC) ≠ 0
Then
 (Smem) → Dmem addressed by EAR
 (EAR) + 1 → EAR
Else
 (Smem) → Dmem addressed by EAR

Статусные биты Не используются.

Описание Данная команда копирует содержимое операнда Smem в ячейку памяти данных, адресованную 16-разрядным непосредственным значением dmad (адрес находится в EAB адресного регистра EAR). Данная команда может быть использована с командами одиночного повторения для перемещения последовательных слов в память данных (с применением косвенной адресации). Количество перемещаемых слов больше на единицу количества в счетчике повторений на начало выполнения команды. Как только конвейер повторений начинает работу, команда становится одноцикловой.

Слова 2 слова.
Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.
Циклы 2 цикла.
Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.
Классы Класс 19А.
Класс 19В.

Пример 1 MVDK 10, 8000h

Перед командой		После команды	
DP	004	DP	004
Память данных			
020Ah	1234	020Ah	1234
8000h	ABCD	8000h	1234

Пример 2 MVDK *AR3–, 1000h

Перед командой		После команды	
AR3	01FF	AR3	01FE
Память данных			
1000h	ABCD	1000h	1234
01FFh	1234	01FFh	1234

20.3.67 MVDM dmad, MMR

Операнды MMR: отображаемый в памяти регистр.
 $0 \leq \text{dmad} \leq 65\,535$

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	1	0	0	1	0	I	A	A	A	A	A	A	A
16 битная константа															

Выполнение dmad → DAR
If (RC) ≠ 0
Then
 (Dmem addressed by DAR) → MMR
 (DAR) + 1 → DAR
Else
 (Dmem addressed by DAR) → MMR

Статусные биты Не используются.

Описание Данная команда копирует данные из ячеек памяти данных dmad (адрес находится в адресном регистре DAR) в отображаемые в памяти регистры MMR. Значение ячейки памяти данных адресуется 16-разрядным непосредственным значением. Как только конвейер повторений начинает работу, команда становится одноцикловой.

Слова 2 слова.

Циклы 2 цикла.

Классы Класс 19А.

Пример MVDM 300h, ВК

Перед командой		После команды	
ВК	ABCD	ВК	1234
Память данных			
0300h	1234	0300h	1234

20.3.68 MVDP Smem, pmad

Операнды	Smem: операнд памяти данных одиночного доступа. $0 \leq \text{pmad} \leq 65\,535$																																																
Код операции	<table><tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td><td>I</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td></tr><tr><td colspan="16">16 битная константа</td></tr></table>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	0	1	1	1	1	1	0	1	I	A	A	A	A	A	A	A	16 битная константа															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																		
0	1	1	1	1	1	0	1	I	A	A	A	A	A	A	A																																		
16 битная константа																																																	
Выполнение	$\text{pmad} \rightarrow \text{PAR}$ If (RC) $\neq$ 0 Then (Smem) $\rightarrow$ Pmem addressed by PAR (PAR) + 1 $\rightarrow$ PAR Else (Smem) $\rightarrow$ Pmem addressed by PAR																																																
Статусные биты	Не используются.																																																
Описание	Данная команда копирует 16-битное значение операнда Smem в ячейку памяти программ, адресованную 16-разрядным непосредственным значением pmad. Данная команда может быть использована как повторяющаяся для перемещения последовательных слов памяти данных (с применением косвенной адресации) в непрерывное пространство памяти программ, адресуемое 16-разрядным непосредственным значением. Блоки источника и приемника не обязательно должны быть встроенными или располагаться вне кристалла. При выполнении команды с повторениями прерывания запрещены. Как только конвейер повторений начинает работу, команда становится одноцикловой.																																																
Слова	2 слова. Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.																																																
Циклы	4 цикла. Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.																																																
Классы	Класс 20A. Класс 20B.																																																
Пример	MVDP 0, 0FE00h																																																

Перед командой		После команды	
DP	004	DP	004
Память программ			
0200h	0123	0200h	0123
Память данных			
FE00h	FFFF	FE00h	0123

20.3.69 MVKD dmad, Smem

Операнды Smem: операнд памяти данных одиночного доступа.
 $0 \leq \text{dmad} \leq 65\,535$

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	1	1	1	0	0	0	0	I	A	A	A	A	A	A	A
16 битная константа																

Выполнение dmad $\rightarrow$ DAR
 If (RC) $\neq$ 0
 Then
 (Dmem addressed by DAR) $\rightarrow$ Smem
 (DAR) + 1 $\rightarrow$ DAR
 Else
 (Dmem addressed by DAR) $\rightarrow$ Smem

Статусные биты Не используются.

Описание Данная команда перемещает данные из одной ячейки памяти данных в другую ячейку памяти данных. Ячейка памяти данных источника адресована 16-разрядным непосредственным операндом dmad а перемещается в Smem. Данная команда может быть использована с командами повторения одиночной инструкции для перемещения последовательных слов в памяти данных (с применением косвенной адресации). Количество перемещаемых слов больше на единицу количества в счетчике повторений на начало выполнения команды. Как только конвейер повторений начинает работу, команда становится одноцикловой.

Слова 2 слова.
 Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Циклы 2 цикла.
 Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Классы Класс 19А.

Класс 19В.

Пример 1 MVKD 300h, 0

Перед командой		После команды	
DP	004	DP	004
Память данных			
0200h	ABCD	0200h	1234
0300h	1234	0300h	1234

Пример 2 MVKD 1000h, *+AR5

Перед командой		После команды	
AR5	01FF	AR5	0200
Память данных			
1000h	1234	1000h	1234
0200h	ABCD	0200h	1234

20.3.70 MVMD MMR, dmad

Операнды MMR: отображаемый в памяти регистр.
0 ≤ dmad ≤ 65 535

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	1	1	1	0	0	1	1	I	A	A	A	A	A	A	A
16 битная константа																

Выполнение dmad → EAR
If (RC) ≠ 0
Then
 (MMR) → Dmem addressed by EAR
 (EAR) + 1 → EAR
Else
 (MMR) → Dmem addressed by EAR

Статусные биты Не используются.

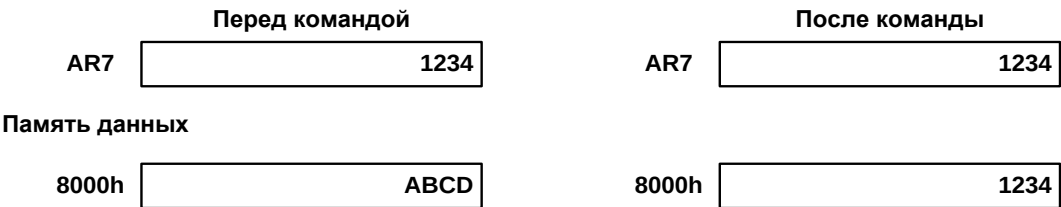
Описание Данная команда перемещает данные из отображаемого в памяти регистра MMR в ячейку памяти данных. Адрес в памяти данных назначается 16-разрядным непосредственным значением dmad. Как только конвейер повторений начинает работу, команда становится одноцикловой.

Слова 2 слова.

Циклы 2 цикла.

Классы Класс 19А.

Пример MVMD AR7, 8000h



20.3.71 MVMM MMRx, MMRy

Операнды MMRx: AR0–AR7, SP
MMRy: AR0–AR7, SP

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	0	0	1	1	1	M	M	R	X	M	M	R	Y
Регистр									MMRX/MMRY							
AR0									0000							
AR1									0001							
AR2									0010							
AR3									0011							
AR4									0100							
AR5									0101							
AR6									0110							
AR7									0111							
SP									1000							

Выполнение (MMRx) → MMRy

Статусные биты Не используются.

Описание	Данная команда перемещает содержимое отображаемого в памяти регистра MMRx в регистр MMRy. Разрешены только девять операндов: AR0–AR7 и SP. Операция чтения из MMRx выполняется в фазе чтения (в отличии от прототипа, где указана фаза декодирования). Операция записи в MMRy выполняется в фазе записи (в отличии от прототипа, где указана фаза доступа). Примечание – Данная команда не повторяемая.
Слова	1 слово.
Циклы	1 цикл.
Классы	Класс 1.
Пример	MVMM SP, AR1

Перед командой		После команды	
AR1	3EFF	AR1	0200
SP	0200	SP	0200

20.3.72 MVPD pmad, Smem

Операнды	Smem: операнд памяти данных одиночного доступа. $0 \leq \text{pmad} \leq 65\,535$																																																
Код операции	<table><tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>I</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td></tr><tr><td colspan="16">16 битная константа</td></tr></table>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	0	1	1	1	1	1	0	0	I	A	A	A	A	A	A	A	16 битная константа															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																		
0	1	1	1	1	1	0	0	I	A	A	A	A	A	A	A																																		
16 битная константа																																																	
Выполнение	pmad → PAR If (RC) ≠ 0 Then (Pmem addressed by PAR) → Smem (PAR) + 1 → PAR Else (Pmem addressed by PAR) → Smem																																																
Статусные биты	Не используются.																																																
Описание	Данная команда перемещает слово в ячейку памяти программ, адресованную 16-разрядным непосредственным значением pmad. Данная инструкция может быть использована как повторяющаяся команда для перемещения последовательных слов, адресуемых 16-разрядным непосредственным адресом памяти программ в непрерывное пространство памяти данных, адресованное Smem. Блоки источника и приемника не обязательно должны быть встроенными или располагаться вне кристалла. При выполнении команды с повторениями прерывания запрещены. Как только конвейер повторений начинает работу, команда становится одноцикловой.																																																
Слова	2 слова. Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem																																																
Циклы	3 цикла. Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.																																																

Классы Класс 21А.
 Класс 21В.
Пример 1 MVDP 0FE00h, 5

Перед командой		После команды	
DP	006	DP	006
Память программ			
FE00h	8A55	FE00h	8A55
Память данных			
0305h	FFFF	0305h	8A55

Пример 2 MVDP 2000h, *AR7–0

Перед командой		После команды	
AR0	0002	AR0	0002
AR7	0FFE	AR7	0FFC
Память программ			
2000h	1234	2000h	1234
Память данных			
0FFEh	ABCD	0FFEh	1234

20.3.73 NEG src [, dst]

Операнды src, dst: А (аккумулятор А).
 В (аккумулятор В).

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	S	D	1	0	0	0	0	1	0	0

Выполнение (src) * –1 → dst
Статусные биты Команда под влиянием OVM.
 Результат влияет на С и OVdst (или OVsrc, если dst = src).
Описание Данная команда рассчитывает дополнительный код (дополнение до двух) содержимого src (также А или В) и сохраняет результат в dst или src, если dst не указан. Данная команда сбрасывает значение бита переноса, С, для любых ненулевых значений аккумулятора. Если аккумулятор равен 0, значение бита переноса - 1. Если аккумулятор равен FF 8000 0000h, операция отрицания вызывает переполнение, поскольку дополнение до двух FF 8000 0000h превышает значения младших 32 битов аккумулятора. Если OVM = 1, значением dst назначается 00 7FFF FFFFh. Если OVM = 0, значением dst назначается 00 8000 0000h. Бит OV для dst устанавливается, чтобы информировать о переполнении в каждом случае.

Слова 1 слово.
Циклы 1 цикл.
Классы Класс 1.
Пример 1 NEG A, В

A	FF	FFFF	F228
B	00	0000	1234
OVA			

A	FF	FFFF	F228
B	00	0000	0DD8
OVA			

Пример 2 NEG B, A

Перед командой			
A	00	0000	1234
B	00	8000	0000
OVA	0		

После команды			
A	FF	8000	0000
B	00	8000	0000
OVA	0		

Пример 3 NEG A

Перед командой			
A	80	0000	0000
OVA	1		
OVM	0		

После команды			
A	80	8000	0000
OVA	1		
OVM	0		

Пример 4 NEG A

Перед командой			
A	80	0000	0000
OVA	0		
OVM	1		

После команды			
A	00	7FFF	FFFF
OVA	1		
OVM	1		

20.3.74 NOP

Операнды Отсутствуют.

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	0	0	1	0	0	1	0	1	0	1

Выполнение Отсутствует.

Статусные биты Не используются.

Описание Пустая операция (нет действий). Увеличивается только РС. Это может быть полезным при создании в конвейере задержек выполнения.

Слова 1 слово.

Циклы 1 цикл.

Классы Класс 1.

Пример NOP
Действия отсутствуют.

20.3.75 NORM src [, dst]

Операнды src, dst: A (аккумулятор A).
B (аккумулятор B).

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	S	D	1	0	0	0	1	1	1	1

Выполнение (src) << TS → dst

Статусные биты Выполнение изменяется под влиянием SXM и OVM. Команда влияет на OVdst (или OVsrc, когда dst = src).

Описание Значение со знаком, содержащееся в src, нормализуется и результат сохраняется в dst или src, если dst не установлен. Нормализация числа с фиксированной запятой разделяет число на дробную часть (мантиссу) и порядок (экспоненту) путем нахождения количества разрядов расширяющих знак числа.

Данная команда за один цикл нормализует значение аккумулятора одновременно с командой EXP, которая рассчитывает экспоненту этого числа. Значение сдвига определяется T(5–0) и кодируется как знаковое число в дополнительном коде. Действующие значения сдвига – от 16 до 31. Для осуществления нормализации сдвигатель использует значение сдвига (в T) в фазе выполнения; нормализация происходит в фазе выполнения.

Слова 1 слово.

Циклы 1 цикл.

Классы Класс 1.

Пример 1 NORM A

	Перед командой				После команды		
A	FF	FFFF	F001	A	FF	8008	0000
T			0013	T			0013

Пример 2 NORM B, A

	Перед командой				После команды		
A	FF	FFFF	F001	A	00	4214	1414
B	21	0A0A	0A0A	B	21	0A0A	0A0A
T			0FF9	T			0FF9

20.3.76 OR

- 1 OR Smem, src
- 2 OR #lk [, SHFT], src [, dst]
- 3 OR #lk, 16, src [, dst]
- 4 OR src [, SHIFT], [, dst]

Операнды src, dst: A (аккумулятор A).
B (аккумулятор B).

Smem: Операнд памяти данных одиночного доступа.

0 ≤ SHFT ≤ 15

–16 ≤ SHIFT ≤ 15

0 ≤ lk ≤ 65 535

Код
операции

1:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	0	1	S	I	A	A	A	A	A	A	A

2:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	S	D	0	1	0	0	S	H	F	T
16 битная константа															

3:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	S	D	0	1	1	0	0	1	0	0
16 битная константа															

4:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	S	D	1	0	1	S	H	I	F	T

Выполнение

- 1: (Smem) OR (src(15–0)) → src
src(39–16) без изменений
- 2: lk << SHFT OR (src) → dst
- 3: k << 16 OR (src) → dst
- 4: (src or [dst]) OR (src) << SHIFT → dst

Статусные
биты

Не используются.

Описание

Эта команда логического сложения ИЛИ (OR) между src и операндом памяти данных одиночного доступа Smem, непосредственным значением lk со сдвигом влево на 16 бит, dst или самим собой. Результат создается в dst или src, если dst не указан. По предписанию команды значения могут быть сдвинуты. При положительных сдвигах (влево) значения младших разрядов обнуляются, а значения старших разрядов не дополняются знаком. Для отрицательных сдвигов (вправо) значения старших разрядов не дополняются знаком.

Слова

Синтаксисы 1 и 4: 1 слово.
Синтаксисы 2 и 3: 2 слова.
Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Циклы

Синтаксисы 1 и 4: 1 цикл.
Синтаксисы 2 и 3: 2 цикла.
Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Классы

Синтаксис 1: Класс 3А.
Синтаксис 1: Класс 3В.
Синтаксисы 2 и 3: Класс 2.
Синтаксис 4: Класс 1.

Пример 1 OR *AR3+, A

Перед командой			После команды				
A	00	00FF	1200	A	00	00FF	1700
AR3			0100	AR3			0101
Память данных							
0100h			1500	0100h			1500

Пример 2 OR A, +3, B

Перед командой				После команды			
A	00	0000	1200	A	00	0000	1200
B	00	0000	1800	B	00	0000	9800

20.3.77 ORM #lk, Smem

Операнды Smem: Операнд памяти данных одиночного доступа.
0 ≤ lk ≤ 65 535

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	1	1	0	1	0	0	1	I	A	A	A	A	A	A	A
	16 битная константа															

Выполнение lk OR (Smem) → Smem

Статусные
биты Не используются.

Описание Эта команда логического поразрядного сложения ИЛИ (OR) между операндом памяти данных одиночного доступа Smem и 16-разрядной константой lk, результат сохраняется в Smem. Данная команда выполняет операцию типа память-память.

Примечание – Данная команда не повторяемая.

Слова 2 слова.
Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Циклы 2 цикла.
Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Классы Класс 18А.
Класс 18В.

Пример ORM 0404h, *AR4+

Перед командой		После команды	
AR4	0100	AR4	0101
Память данных			
0100h	4444	0100h	4444

20.3.78 POLY Smem

Операнды Smem : операнд памяти данных одиночного доступа.

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	1	1	0	1	1	0	I	A	A	A	A	A	A	A

Выполнение Round (A(32–16) x (T) + (B)) → A
(Smem) << 16 → B

Статусные биты Выполнение изменяется под влиянием FRCT, OVM и SXM.
Команда влияет на OVA.

Описание Данная команда сдвигает содержимое операнда памяти данных одиночного доступа Smem на 16 бит влево и сохраняет результат в аккумуляторе В. Одновременно, эта команда умножает старшую часть аккумулятора А (биты 32–16) на содержание Т, складывает произведение с аккумулятором В, округляет результат операции, и сохраняет конечный результат в аккумуляторе А. Эта команда может быть полезна для вычисления полинома при осуществлении расчетов, затрачивая 1 цикл при расчете слагаемого полинома.

Слова 1 слово.
Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Циклы 1 цикл.
Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Классы Класс 3А.
Класс 3В.

Пример POLY *AR3+%

Перед командой				После команды			
A	00	1234	0000	A	00	0627	0000
B	00	0001	0000	B	00	2000	0000
T				T	5678		
AR3	0200			AR3	0201		
Память данных							
0200h	2000			0200h	2000		

20.3.79 POPD Smem

Операнды Smem: операнд памяти данных одиночного доступа.

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	0	0	1	0	1	1	I	A	A	A	A	A	A	A

Выполнение (TOS) → Smem
(SP) + 1 → SP

Статусные биты Не используются.

Описание	Данная команда перемещает содержимое ячейки памяти данных, адресуемое SP в ячейку памяти, определяемую Smem. SP увеличивается на 1.
Слова	1 слово. Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.
Циклы	1 цикл. Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.
Классы	Класс 17A. Класс 17B.
Пример	POPD 10

Перед командой		После команды	
DP	008	DP	008
SP	0300	SP	0301
Память данных			
0300h	0092	0300h	0092
040Ah	0055	040Ah	0092

20.3.80 POPM MMR

Операнды	MMR: отображаемый в памяти регистр.																																
Код операции	<table><tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>I</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td></tr></table>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	1	0	0	0	1	0	1	0	I	A	A	A	A	A	A	A
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																		
1	0	0	0	1	0	1	0	I	A	A	A	A	A	A	A																		
Выполнение	(TOS) → MMR (SP) + 1 → SP																																
Статусные биты	Не используются.																																
Описание	Данная команда перемещает содержимое ячейки памяти данных, адресуемое SP в определенный регистр MMR. SP инкрементируется.																																
Слова	1 слово.																																
Циклы	1 цикл.																																
Классы	Класс 17A.																																
Пример	POPM AR5																																

Перед командой		После команды	
AR5	0055	AR5	0060
SP	03F0	SP	03F1
Память данных			
03F0h	0060	03F0h	0060

20.3.81 PORTR PA, Smem

Операнды Smem: операнд памяти данных одиночного доступа.
 $0 \leq PA \leq 65\,535$

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	1	1	1	0	1	0	0	I	A	A	A	A	A	A	A
Адрес порта																

Выполнение (PA) → Smem

Статусные биты Не используются.

Описание Данная команда считывает 16-разрядное значение с внешнего порта ввода-вывода PA (16-разрядный непосредственный адрес) в ячейку памяти данных определяемую Smem. Сигнал IS устанавливается в состояние низкого потенциала для индикации доступа по вводу-выводу, временные соотношения IOSTRB и READY такие же, как при чтении внешней памяти данных.

Слова 2 слова.

Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Циклы 2 цикла (зависит от внешней операции ввода-вывода)

Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Классы Класс 27A.

Класс 27B.

Пример PORTR 05, INDAT ; INDAT .equ 60h

Перед командой		После команды	
DP	000	DP	000
Память ввода-вывода			
0005h	7FFA	0005h	7FFA
Память данных			
0060h	0000	0060h	7FFA

20.3.82 PORTW Smem, PA

Операнды Smem: операнд памяти данных одиночного доступа.
 $0 \leq PA \leq 65\,535$

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	1	1	1	0	1	0	1	I	A	A	A	A	A	A	A
Адрес порта																

Выполнение (Smem) → PA

Статусные биты Не используются.

Описание	Данная команда записывает 16-разрядное значение из ячейки памяти данных определенную значением Smem во внешний порт ввода-вывода по адресу PA. Сигнал IS устанавливается в состояние низкого потенциала для отображения доступа по вводу-выводу, временные соотношения IOSTRB и READY такие же, как при записи во внешнюю память данных.
Слова	2 слова. Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.
Циклы	2 цикла (зависит от внешней операции ввода-вывода). Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.
Классы	Класс 28A. Класс 28B.
Пример	PORTW OUTDAT, 5h ; OUTDAT .equ 07h

Перед командой		После команды	
DP	001	DP	001
Память ввода-вывода			
0005h	0000	0005h	7FFA
Память данных			
0087h	7FFA	0087h	7FFA

20.3.83 PSHD Smem

Операнды	Smem: операнд памяти данных одиночного доступа.															
Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	1	0	0	1	0	1	1	I	A	A	A	A	A	A	A
Выполнение	(SP) - 1 → SP (Smem) → TOS															
Статусные биты	Не используются.															
Описание	После уменьшения SP на 1, данная команда сохраняет содержание ячейки памяти Smem в ячейку памяти данных, адресованную SP. Чтение SP осуществляется во время фазы чтения (в прототипе – декодирования); сохранение происходит во время фазы записи (в прототипе – доступа).															
Слова	1 слово. Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.															
Циклы	1 цикл. Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.															
Классы	Класс 16A. Класс 16B.															

Пример

PSHD *AR3+

Перед командой		После команды	
AR3	0200	AR3	0201
SP	8000	SP	7FFF
Память данных			
0200h	07FF	0200h	07FF
7FFFh	0092	7FFFh	07FF

20.3.84 PSHM MMR

Операнды

MMR: отображаемый в памяти регистр.

Код операции

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	0	1	0	I	A	A	A	A	A	A	A

Выполнение

(SP) - 1 → SP
(MMR) → TOS

Статусные биты

Не используются.

Описание

После уменьшения SP на 1, данная команда сохраняет содержимое регистра MMR в ячейке памяти данных, адресованной SP.

Слова

1 слово.

Циклы

1 цикл.

Классы

Класс 16A.

Пример

PSHM BRC

Перед командой		После команды	
BRC	1234	BRC	1234
SP	2000	SP	1FFF
Память данных			
1FFFh	07FF	1FFFh	1234

20.3.85 RC[D] cond [, cond [, cond]]

Операнды

В таблице перечислены условия (операнд условий) для данной команды.

Операнд условий	Описание	Код условия	Операнд условий	Описание	Код условия
BIO	$\overline{\text{BIO}}_{\text{Low}}$	0000 0011	NBIO	$\overline{\text{BIO}}_{\text{high}}$	0000 0010
C	C=1	0000 1100	C	C=0	0000 1000
TC	TC=1	0011 0000	TC	TC=0	0010 0000
AEQ	(A) = 0	0100 0101	BEQ	(B) = 0	0100 1101
ANEQ	(A) ≠ 0	0100 0100	BNEQ	(B) ≠ 0	0100 1100
AGT	(A) > 0	0100 0110	BGT	(B) > 0	0100 1110
AGEQ	(A) ≥ 0	0100 0010	BGEQ	(B) ≥ 0	0100 1010
ALT	(A) < 0	0100 0011	BLT	(B) < 0	0100 1011
ALEQ	(A) ≤ 0	0100 0111	BLEQ	(B) ≤ 0	0100 1111
AOV	A переполнение	0111 0000	BOV	B переполнение	0111 1000
ANOV	A отсутствие переполнения	0110 0000	BNOV	B отсутствие переполнения	0110 1000
UNC	Безусловный	0000 0000			

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	1	1	Z	0	С	С	С	С	С	С	С	С

Выполнение If (cond(s))
Then
(TOS) → PC
(SP) + 1 → SP
Else
(PC) + 1 → PC

Статусные биты Не используются.

Описание При выполнении условия, заданного операндом условия, эта команда заменяет значение PC значением памяти данных из TOS и увеличивает SP на 1. Если условие не выполняется, то команда просто увеличивает PC на 1.
При задержанной команде возврата (определяется по наличию индекса D), две однословные команды или одна двухсловная команда, следующие за командой перехода, извлекаются из памяти программ и выполняются.
Два командных слова, следующих за данной командой не влияют на значения условий, проходящих проверку.
Данная команда осуществляет проверку множественных условий до передачи управления другому разделу программы. Команда осуществляет проверку условий по отдельности или в совокупности с другими условиями. Необходимо сочетать условия только из одной группы, как указано ниже:
Группа 1. Можно выбрать не более двух условий. Каждое условие должно принадлежать разной категории (категория А или В); нельзя выбрать два условия из одной категории. Например, можно осуществлять проверку EQ и OV одновременно, но нельзя осуществлять проверку GT и NEQ в одно и то же время. Аккумулятор должен быть одним для обоих условий; нельзя одной командой осуществлять проверку условий для двух аккумуляторов. Например, можно осуществлять проверку AGT и AOV одновременно, но нельзя осуществлять проверку AGT и BOV в одно и то же время.
Группа 2. Можно выбрать не более трех условий. Каждое из этих условий должно быть из разных категорий (категория А, В или С); нельзя выбирать два условия из одной категории. Например, можно осуществлять проверку TC, C и BIO одновременно, но нельзя осуществлять проверку NTC, C и NC в одно и то же время.

Условия для данной команды

Группа 1		Группа 2		
Категория А	Категория В	Категория А	Категория В	Категория С
EQ	OV	TC	C	BIO
NEQ	NOV	NTC	NC	NBIO
LT				
LEQ				
GT				
GEO				

Примечание – Данная команда не повторяемая.

Слова 1 слово.

Циклы	5 циклов (истинное условие). 3 цикла (ложное условие). 3 цикла (с задержкой).
Классы	Класс 32.
Пример	RC AGEQ, ANOV ; возврат осуществляется если содержание аккумулятора ; А положительно и значение бита OVA равно нулю

Перед командой		После команды	
PC	0807	PC	2002
OVA	0	OVA	0
SP	0308	SP	0309
Память данных			
0308h	2002	0308h	2002

20.3.86 READA Smem

Операнды	Smem: операнд памяти данных одиночного доступа.																																															
Код операции	<table><tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>I</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td></tr></table>																15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	0	1	1	1	1	1	1	0	I	A	A	A	A	A	A	A
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																	
0	1	1	1	1	1	1	0	I	A	A	A	A	A	A	A																																	
Выполнение	A → PAR If ((RC) ≠ 0) (Pmem (addressed by PAR)) → Smem (PAR) + 1 → PAR (RC) – 1 → RC Else (Pmem (addressed by PAR)) → Smem																																															
Статусные биты	Не используются.																																															
Описание	Команда перемещает слово из ячейки памяти программ, адресуемой аккумулятором A в ячейку памяти данных, определенную Smem. Как только конвейер повторений начинает работу, команда становится одноцикловой. Для адресации ячейки памяти программ используется аккумулятор A(15-0). Данная команда может быть использована как повторяющаяся для перемещения последовательных слов (начиная с адреса, указанного в аккумуляторе A) в непрерывное пространство памяти данных с использованием косвенной адресации. Блоки источника и приемника не обязательно должны быть встроенными или располагаться вне кристалла.																																															
Слова	1 слово. Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.																																															
Циклы	5 циклов. Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.																																															
Классы	Класс 25A. Класс 25B.																																															

Пример READA 6

Перед командой				После команды			
A	00 0000 0023			A	00 0000 0023		
DP	004			DP	004		
Память программ							
0023h	0306			0023h	0306		
Память данных							
0206h	0075			0206h	0306		

20.3.87 RESET

Операнды Отсутствуют.

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	1	1	1	1	1	0	0	0	0	0

Выполнение В эти поля PMST, ST0 и ST1 загружаются указанные значения:

(IPTR) << 7 _ PC	0 → OVA	0 → OVB
1 → C	1 → TC	0 → ARP
0 → DP	1 → SXM	0 → ASM
0 → BRAF	0 → HM	1 → XF
0 → C16	0 → FRCT	0 → CMPT
0 → CPL	1 → INTM	0 → IFR
0 → OVM		

Статусные биты Изменение статусных бит представлено в разделе выполнения.

Описание Данная команда представляет собой немаскируемый программный сброс, который может быть испрльзован в любое время для помещения микросхемы в известное состояние. Когда команда сброса выполнена, возникают операции, указанные в разделе Выполнение. Вывод MP/MC не фиксируется во время программного сброса. Инициализация IPTR и периферийных регистров отличается от инициализации с использованием контактной площадки RS. Данная команда не зависит от INTM; напротив, она устанавливает в INTM значение 1 с целью запрета прерываний.

Примечание – Данная команда не повторяемая.

Слова 1 слово.

Циклы 3 цикла.

Классы Класс 35.

Пример RESET

Перед командой		После команды	
PC	0025	PC	0080
INTM	0	INTM	1
IPTR	1	IPTR	1

20.3.88 RET[D]

Операнды Отсутствуют.

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	1	1	Z	0	0	0	0	0	0	0	0	0

Выполнение (TOS) → PC
(SP) + 1 → SP

Статусные биты Не используются.

Описание Данная команда заменяет значение PC 16-разрядным значением из TOS. SP увеличивается на 1. Если возврат задержанный (определяется по наличию индекса D), то две однословные команды или одна двухсловная, следующие за командой перехода, извлекаются из памяти программ и выполняются.

Примечание – Данная команда не повторяемая.

Слова 1 слово.

Циклы 5 циклов.

3 цикла (задержанная).

Классы Класс 32.

Пример RET

Перед командой		После команды	
PC	2112	PC	1000
SP	0300	SP	0301
Память данных			
0300h	1000	0300h	1000

20.3.89 RETE[D]

Операнды Отсутствуют.

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	Z	0	1	1	1	0	1	0	1	1

Выполнение (TOS) → PC
(SP) + 1 → SP
0 → INTM

Статусные биты Влияет на INTM.

Описание Данная команда заменяет значение PC 16-разрядным значением из TOS. Выполнение продолжается с этого адреса. SP увеличивается на 1. Данная команда автоматически сбрасывает маску прерывания (INTM) в ST1. (сбрасывание этого бита разрешает прерывания.) Если возврат задержанный (определяется по наличию индекса D), то две однословные команды или одна двухсловная, следующие за командой перехода, извлекаются из памяти программ и выполняются.

Примечание – Данная команда не повторяемая.

Слова 1 слово.
Циклы 5 циклов.
 3 цикла (задержанная).
Классы Класс 32.
Пример RETE

Перед командой		После команды	
PC	01C3	PC	0110
SP	2001	SP	2002
ST1	xCxx	ST1	x4xx
Память данных			
2001h	0110	2001h	0110

20.3.90 RETF[D]

Операнды Отсутствуют.

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	Z	0	1	0	0	1	1	0	1	1

Выполнение (RTN) → PC
 (SP) + 1 → SP
 0 → INTM

Статусные биты Влияет на INTM.

Описание В прототипе данная команда заменяет значение PC 16-разрядным значением в RTN. RTN содержит информацию об адресе, в котором должна осуществить возврат программа управления прерываниями. RTN загружается в PC во время возврата вместо чтения PC из стека.
 В процессоре реализация данной команды не отличается от реализации команды RETE.

Примечание – Данная команда не повторяемая.

Слова 1 слово.
Циклы 3 цикла.
 1 цикл (задержанная).
Классы Класс 33.
Пример RETF

Перед командой		После команды	
PC	01C3	PC	0110
SP	2001	SP	2002
ST1	xCxx	ST1	x4xx
Память данных			
2001h	0110	2001h	0110

20.3.91 RND src [, dst]

Операнды src , dst: A (аккумулятор A).
B (аккумулятор B).

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	S	D	1	0	0	1	1	1	1	1

Выполнение (src) + 8000h → dst

Статусные биты Исполнение зависит от OVM.

Описание Данная команда округляет содержимое src (A или B) добавлением 2^{15} . Округленное значение сохраняется в dst или src, если dst не указан.

Примечание – Данная команда не повторяемая, она не реализована в прототипе и потому ее нет и в процессоре. Наличие ее кода в потоке команд воспринимается как холостая инструкция NOP.

20.3.92 ROL src

Операнды src: A (аккумулятор A).
B (аккумулятор B).

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	0	S	1	0	0	1	0	0	0	1

Выполнение (C) → src(0)
(src(30–0)) → src(31–1)
(src(31)) → C
0 → src(39–32)

Статусные биты Результат выполнения зависит от C.
Инструкция влияет на C.

Описание Данная команда циклически сдвигает каждый бит src влево на 1 бит. Значение бита переноса C, бывшее до выполнения этой команды сдвигается в самые младшие разряды src. Затем самые старший разряд src сдвигаются в C. Значения защитных битов src сбрасываются.

Слова 1 слово.

Циклы 1 цикл.

Классы Класс 1.

Пример ROL A

Перед командой				После команды			
A	5F	B000	1234	A	00	6000	2468
C	0			C	1		

20.3.93 ROLTC src

Операнды src: A (аккумулятор A).
B (аккумулятор B).

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	0	S	1	0	0	1	0	0	1	0

Выполнение (TC) → src(0)
(src(30–0)) → src(31–1)
(src(31)) → C
0 → src(39–32)

Статусные биты Команда изменяет бит C.
Результат зависит от TC.

Описание Данная команда циклически сдвигает каждый бит src влево на 1 бит. Значение бита TC бывшее до выполнения этой команды сдвигается в самый младший разряд src. Затем самый старший разряды src сдвигается в C. Значения защитных бит src сбрасываются.

Слова 1 слово.

Циклы 1 цикл.

Классы Класс 1.

Пример ROLTC A

	Перед командой				После команды		
A	81	C000	5555	A	00	8000	AAAB
C			x	C			1
TC			1	TC			1

20.3.94 ROR src

Операнды src: A (аккумулятор A).
B (аккумулятор B).

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	0	S	1	0	0	1	0	0	0	0

Выполнение (C) → src(31)
(src(31–1)) → src(30–0)
(src(0)) → C
0 → src(39–32)

Статусные биты Команда влияет на значение C.
Результат зависит от прежнего значения C.

Описание Данная команда циклически сдвигает каждый бит src вправо на 1 бит. Значение бита переноса C, бывшее до выполнения этой команды сдвигается в самый старший разряд src. Затем самый младший разряд src сдвигается в C. Значения защитных бит src сбрасываются.

Слова 1 слово.

Циклы 1 цикл.

Классы Класс 1.

Пример ROR A

	Перед командой				После команды		
A	7F	B000	1235	A	00	5800	091A
C			0	C			1

20.3.95 RPT

1 RPT Smem

2 RPT #K

3 RPT #lk

Операнды Smem: операнд памяти данных одиночного доступа.

$0 \leq K \leq 255$

$0 \leq lk \leq 65\,535$

Код операции 1:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	1	1	1	1	A	A	A	A	A	A	A

2:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	1	1	0	0	K	K	K	K	K	K	K	K

3:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	1	1	1	0	0	0	0
16 битная константа															

Выполнение 1: (Smem) → RC

2: K → RC

3: lk → RC

Статусные биты Не используются.

Описание В счетчик повторений (RC) загружается число итераций выполнения команды. Число итераций (n) задается в 16-разрядном операнде памяти данных Smem или 8- или 16-разрядной константой, K или lk, соответственно. Команда, следующая за повторяющейся командой, выполняется n + 1 раз. Доступ к RC отсутствует во время его декрементирования.

Примечание – Сама данная команда не повторяемая.

Слова Синтаксисы 1 и 2: 1 слово.

Синтаксис 3: 2 слова.

Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Циклы Синтаксис 1: 3 цикла.

Синтаксис 2: 1 цикл.

Синтаксис 3: 2 цикла.

Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Классы Синтаксис 1: Класс 5А.

Синтаксис 1: Класс 5В.

Синтаксис 2: Класс 1.

Синтаксис 3: Класс 2.

Пример 1 RPT DAT127 ; DAT127 .EQU 0FFF

Перед командой		После команды	
RC	0	RC	000C
DP	031	DP	031
Память данных			
0FFFh	000C	0FFFh	000C

Пример 2 RPT #2 ; Повторение следующей инструкции 3 раза

Перед командой		После команды	
RC	0	RC	0002

Пример 3 RPT #1111h ; Повторение следующей инструкции 4370 раз

Перед командой		После команды	
	0	RC	1111

20.3.96 RPTB[D] pmad

Операнды $0 \leq \text{pmad} \leq 65\,535$

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	0	Z	0	0	1	1	1	0	0	1	0
16 битная константа																

Выполнение $1 \rightarrow \text{BRAf}$
 If (delayed) then
 $(\text{PC}) + 4 \rightarrow \text{RSA}$
 Else
 $(\text{PC}) + 2 \rightarrow \text{RSA}$
 $\text{pmad} \rightarrow \text{REA}$

Статусные биты Команда влияет на значение BRAF.

Описание Данная команда повторяет блок команд то количество раз, какое установлено счетчиком повторений блока (BRC), являющимся регистром, отображаемым в памяти. BRC должен быть загружен до выполнения этой команды. Когда команда выполнена, в регистр начального адреса повторений блока (RSA) загружается $\text{PC} + 2$ (или $\text{PC} + 4$ при использовании задержанной команды), а в регистр конечного адреса повторений блока (REA) загружается конечный адрес памяти программ (pmad).

Эта команда является прерываемой. Команды одиночного повторения могут быть включены в группу повторения блока команд. Повторяющиеся блоки могут быть вложенными друг в друга, но при возврате в соответствующий блок необходимо убедиться в следующем:

соответствующие BRC, RSA и REA были сохранены(при необходимости) и затем восстановлены.

текущий флаг повторений блока (BRAf) установлен должным образом.

В случае задержанной команды повторения блока (определяется по наличию индекса D), две однословные команды или одна двухсловная, следующие за командой перехода, извлекаются из памяти программ и выполняются.

Примечание – Повторение блока может быть прекращено сбрасыванием значения бита BRAF. Данная команда не повторяемая.

Слова 2 слова.
Циклы 4 цикла.
 2 цикла (задержанная).
Классы Класс 29А.
Пример 1 ST #99, BRC
 RPTB end_block – 1
 ; end_block = Bottom of Block

Перед командой		После команды	
PC	1000	PC	1002
BRC	1234	BRC	0063
RSA	5678	RSA	1002
REA	9ABC	REA	end block - 1

Пример 2 ST #99, BRC ; выполнение блока 100 раз
 RPTBD end_block – 1
 MVDM POINTER, AR1
 ; initialize pointer
 ; end_block ; Bottom of Block

Перед командой		После команды	
PC	1000	PC	1004
BRC	1234	BRC	0063
RSA	5678	RSA	1004
REA	9ABC	REA	end block - 1

20.3.97 RPTZ dst, #lk

Операнды dst: A (аккумулятор A).
 B (аккумулятор B).
 $0 \leq lk \leq 65\,535$

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	0	0	D	0	1	1	1	0	0	0	1
	16 битная константа															

Выполнение $0 \rightarrow \text{dst}$
 $lk \rightarrow \text{RC}$
Статусные биты Не используются.

Описание	Данная команда сбрасывает значение dst и повторяет следующую команду $n + 1$ раз, где n – значение счетчика повторений (RC). Значение RC становится равным 16-разрядной константе lk.
Слова	2 слова.
Циклы	2 цикла.
Классы	Класс 2.
Пример	RPTZ A, 1023 ; Repeat the next instruction 1024 times STL A, *AR2+

Перед командой			После команды		
A	0F	FE00	8000	A	00 0000 0000
RC			0000	RC	03FF

20.3.98 RSBX N, SBIT

Операнды	$0 \leq \text{SBIT} \leq 15$ $N = 0 \text{ or } 1$															
Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	N	0	1	0	1	1	S	B	I	T
Выполнение	$0 \rightarrow \text{STN}(\text{SBIT})$															
Статусные биты	Не используются.															
Описание	Данная команда сбрасывает значение заданного бита в статусном регистре 0 или 1. N назначает номер статусного регистра для внесения изменений, а SBIT задает номер изменяемого бита. Название поля статусного регистра может быть использовано в качестве операнда вместо операндов N и SBIT (см. Пример1). Примечание – Данная команда не повторяемая.															
Слова	1 слово.															
Циклы	1 цикл.															
Классы	Класс 1.															
Пример 1	RSBX SXM ; SXM подразумевает: $n=1$ и $\text{SBIT}=8$															

Перед командой		После команды	
ST1	35CD	ST1	34CD

Пример 2 RSBX 1,8

Перед командой		После команды	
ST1	35CD	ST1	34CD

20.3.99 SACCD src, Xmem, cond

Операнды	src: A (аккумулятор A). B (аккумулятор B). Xmem: операнд памяти данных двойного доступа

В таблице перечислены условия (в соответствии с кодом операнда условий cond) для данной команды.

Операнд условий	Описание	Код условия	Операнд условий	Описание	Код условия
AEQ	(A) = 0	0101	BEQ	(B) = 0	1101
ANEQ	(A) ≠ 0	0100	BNEQ	(B) ≠ 0	1100
AGT	(A) > 0	0110	BGT	(B) > 0	1110
AGEQ	(A) ≥ 0	0010	BGEQ	(B) ≥ 0	1010
ALT	(A) < 0	0011	BLT	(B) < 0	1011
ALEQ	(A) ≤ 0	0111	BLEQ	(B) ≤ 0	1111

Код

операции

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	1	1	1	S	X	X	X	X	C	O	N	D

Выполнение

If (cond)

Then

(src) << (ASM – 16) → Xmem

Else

(Xmem) → (Xmem)

Статусные биты

Изменяется под влиянием ASM и SXM.

Описание

Если условие истинно, эта команда сохраняет src со сдвигом влево на (ASM – 16). Значение сдвига находится в ячейке памяти, определяемой Xmem. Если условие ложное, то команда считывает Xmem и записывает значение обратно в Xmem по тому же адресу; таким образом, Xmem остается прежним. Независимо от условия Xmem всегда считывается и обновляется.

Слова

1 слово.

Циклы

1 цикл.

Классы

Класс 15.

Пример 1

SACCD A, *AR3+0%, ALT

Перед командой				После команды			
A	FF	FE00	4321	A	FF	FE00	4321
ASM	01			ASM	01		
AR0	0002			AR0	0002		
AR3	0202			AR3	0204		
Память данных							
0202h	0101			0202h	FC00		

20.3.100 SAT src

Операнды

src: A (аккумулятор A).
B (аккумулятор B).

Код

операции

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	1	1	1	S	X	X	X	X	C	O	N	D

Выполнение

Насыщенный (src) → src

Статусные биты	Команда влияет на OVsrc.
Описание	Независимо от значения OVM, данная команда обеспечивает насыщение содержания src на 32 бита.
Слова	1 слово.
Циклы	1 цикл.
Классы	Класс 1.
Пример 1	SAT B

	Перед командой	После команды
B	71 2345 6789	00 7FFF FFFF
OVb	x	1

Пример 2 SAT A

	Перед командой	После команды
A	F8 1234 5678	FF 8000 0000
OvA	x	1

Пример 3 SAT B

	Перед командой	После команды
B	00 0012 3456	00 0012 3456
OVb	x	0

20.3.101 SFTA src, SHIFT [, dst]

Операнды src, dst A (аккумулятор A)
B (аккумулятор B)
 $-16 \leq \text{SHIFT} \leq 15$

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	S	D	0	1	1	S	H	I	F	T

Выполнение If SHIFT < 0
Then
 $(\text{src}((-\text{SHIFT}) - 1)) \rightarrow C$
 $(\text{src}(39-0)) \ll \text{SHIFT} \rightarrow \text{dst}$
 If SXM = 1
 Then
 $(\text{src}(39)) \rightarrow \text{dst}(39-(39 + (\text{SHIFT} + 1)))$ [or $\text{src}(39-(39 + (\text{SHIFT} + 1)))$],
 if dst is not specified]
 Else
 $0 \rightarrow \text{dst}(39-(39 + (\text{SHIFT} + 1)))$ [or $\text{src}(39-(39 + (\text{SHIFT} + 1)))$],
 if dst is not specified]
 Else
 $(\text{src}(39 - \text{SHIFT})) \rightarrow C$
 $(\text{src}) \ll \text{SHIFT} \rightarrow \text{dst}$
 $0 \rightarrow \text{dst}((\text{SHIFT} - 1)-0)$ [or $\text{src}((\text{SHIFT} - 1)-0)$, if dst is not specified]

Статусные биты	Результат зависит от SXM и OVM. Команда влияет на C и OVdst (или OVsrc, если dst = src).
Описание	Данная команда арифметически сдвигает src и сохраняет результат в dst или src, если dst не указан. Выполнение команды зависит от значения SHIFT: если значение SHIFT меньше 0, то: 1 src((-SHIFT) - 1) копируется в бит переноса C. - если SXM равен 1, команда выполняет арифметический сдвиг вправо и старшие биты src сдвигаются в dst(39-(39 + (SHIFT + 1))). - если SXM равно 0, 0 записывается в dst(39-(39 + (SHIFT + 1))). если значение SHIFT больше, чем 0, то: a) src(39 - SHIFT) копируется в бит переноса C. б) команда осуществляет арифметический сдвиг влево. в) 0 записывается в dst((SHIFT - 1)-0).
Слова	1 слово.
Циклы	1 цикл.
Классы	Класс 1.
Пример 1	SFTA A, -5, B

Перед командой				После команды			
A	FF	8765	0055	A	FF	8765	0055
B	00	4321	1234	B	FF	FC3B	2802
C				C			
SXM				SXM			
			x				1
			1				1

Пример 2 SFTA B, +5

Перед командой				После команды			
B	80	AA00	1234	B	15	4002	4680
C				C			
OVM				OVM			
SXM				SXM			
			0				1
			0				0
			0				0

20.3.102 SFTC src

Операнды src: A (аккумулятор A).
B (аккумулятор B).

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	0	S	1	0	0	1	0	1	0	0

Выполнение If (src) = 0
Then
1 → TC
Else
If (src(31)) XOR (src(30)) = 0
Then (two significant sign bits)
0 → TC
(src) << 1 → src
Else (only one sign bit)
1 → TC

Статусные биты	Команда определяет значение бита ТС.
Описание	Если src имеет два совпадающих старших знаковых бита, то команда сдвигает 32-разрядный src влево на 1бит. Если есть два совпадающих знаковых бита, то значение бита тестирования (ТС) сбрасывается; в противном случае устанавливается в 1.
Слова	1 слово.
Циклы	1 цикл.
Классы	Класс 1.
Пример 1	SFTC A

Перед командой			После команды		
A	FF	FFFF	F001	A	FF
ТС			x	ТС	0

20.3.103 SFTL src, SHIFT [, dst]

Операнды	src, dst:	A (аккумулятор A). B (аккумулятор B).

$-16 \leq \text{SHIFT} \leq 15$

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	0	S	D	1	1	1	S	H	I	F	T

Выполнение	If SHIFT < 0 Then $\text{src}((-\text{SHIFT}) - 1) \rightarrow C$ $\text{src}(31-0) \ll \text{SHIFT} \rightarrow \text{dst}$ $0 \rightarrow \text{dst}(39-(31 + (\text{SHIFT} + 1)))$ If SHIFT = 0 Then $0 \rightarrow C$ Else $\text{src}(31 - (\text{SHIFT} - 1)) \rightarrow C$ $\text{src}((31 - \text{SHIFT})-0) \ll \text{SHIFT} \rightarrow \text{dst}$ $0 \rightarrow \text{dst}((\text{SHIFT} - 1)-0)$ [or $\text{src}((\text{SHIFT} - 1)-0)$, if dst is not specified] $0 \rightarrow \text{dst}(39-32)$ [or $\text{src}(39-32)$, if dst is not specified]

Статусные биты	Операция влияет на значение C.

Описание	Данная команда реализует логический сдвиг src и сохраняет результат в dst или src, если в мнемонике команды dst не указан. Значения защитных бит dst или src, если dst пропущен, сбрасываются. Выполнение команды зависит от значения SHIFT: если значение SHIFT меньше 0, то: 1 src((−SHIFT) − 1) копируется в бит переноса, C. 2 Команда выполняет логический сдвиг вправо. 3 0 записывается в dst(39−(31 + (SHIFT + 1))). если значение SHIFT больше 0, то: а) src(31 − (SHIFT − 1)) копируется в бит переноса, C. б) команда выполняет логический сдвиг влево. в) 0 записывается в dst((SHIFT − 1)−0).

Слова 1 слово.
Циклы 1 цикл.
Классы Класс 1.
Пример 1 SFTL A, -5, B

Перед командой				После команды			
A	FF	8765	0055	A	FF	8765	0055
B	FF	8000	0000	B	00	043B	2802
C			0	C			1

Пример 2 SFTL B, +5

Перед командой				После команды			
B	80	AA00	1234	B	00	4002	4680
C			0	C			1

20.3.104 SQDST Xmem, Ymem

Операнды Xmem, Ymem: операнды памяти данных двойного доступа.

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	0	0	0	1	0	X	X	X	X	Y	Y	Y	Y

Выполнение $(A(32-16)) \times (A(32-16)) + (B) \rightarrow B$
 $((Xmem) - (Ymem)) \ll 16 \rightarrow A$

Статусные биты Результат выполнения зависит от OVM, FRCT и SXM.
 Команда влияет на значение C, OVA и OVB.

Описание Используемая в режиме одиночного повторения, данная команда вычисляет квадрат расстояния между двумя векторами. Старшая часть аккумулятора A (биты 32–16) возводится в квадрат, произведение складывается с аккумулятором B, и результат записывается в аккумулятор B. Ymem вычитается из Xmem, разница сдвигается на 16 бит влево, и результат записывается в аккумулятор A. Значение, которое возводится в квадрат (A(32–16)) – это значение аккумулятора до выполнения вычитания в данной команде.

Слова 1 слово.
Циклы 1 цикл.
Классы Класс 7.
Пример 1 SQDST *AR3+, AR4+

Перед командой				После команды			
A	FF	ABCD	0000	A	FF	FFAB	0000
B	00	0000	0000	B	00	1BB1	8229
FRCT			0	FRCT			0
AR3			0100	AR3			0101
AR4			0200	AR4			0201

Память данных							
0100h			0055	0100h			0055
0200h			00AA	0200h			00AA

1 SQR Smem, dst

2 SQR A, dst

Операнды Smem: Операнд памяти данных одиночного доступа
dst: A (аккумулятор A).
B (аккумулятор B).

Код операции 1:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	1	1	D	I	A	A	A	A	A	A	A

2:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	0	D	1	0	0	0	1	1	0	1

Выполнение 1: (Smem) → T
(Smem) x (Smem) → dst
2: (A(32–16)) x (A(32–16)) → dst

Статусные биты Результат зависит от OVM и FRCT.
Команда влияет на OVsrc.

Описание Данная команда вычисляет квадрат операнда памяти данных одиночного доступа Smem или старшей части аккумулятора A (биты 32–16) и записывает результат в dst. T не изменяется во время использования аккумулятора A; в противном случае, Smem сохраняется в T.

Слова 1 слово.

Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Циклы 1 цикл.

Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Классы Синтаксис 1: Класс 3A.

Синтаксис 1: Класс 3B.

Синтаксис 2: Класс 1.

Пример 1 SQR 30, B

Перед командой				После команды			
B	00	0000	01F4	B	00	0000	00E1
T				T	000F		
FRCT	0			FRCT	0		
DP	006			DP	006		
Память данных							
031Eh	000F			031Eh	000F		

Пример 2 SQR A, B

Перед командой				После команды			
A	00	000F	0000	A	FF	8765	0055
B	00	0101	0101	B	00	0000	01C2
FRCT	1			FRCT	1		

20.3.105 SQURA Smem, src

Операнды Smem: Операнд памяти данных одиночного доступа.
src: A (аккумулятор A).
B (аккумулятор B).

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	1	1	1	0	0	S	I	A	A	A	A	A	A	A

Выполнение (Smem) → T
(Smem) x (Smem) + (src) → src

Статусные биты Результат выполнения инструкции зависит от OVM и FRCT.
Инструкция влияет на OVsrc.

Описание Данная команда сохраняет значение Smem в T, затем она вычисляет квадрат Smem и складывает произведение с src. Результат записывается в src.

Слова 1 слово.
Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Циклы 1 цикл.
Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Классы Класс 3A.
Класс 3B.

Пример 1 SQURA 30, B

Перед командой				После команды			
B	00	0320	0000	B	00	0320	00E1
T				T	000F		
FRCT	0			FRCT	0		
DP	006			DP	006		
Память данных							
031Eh	000F			031Eh	000F		

Пример 2 SQURA *AR3+, A

Перед командой				После команды			
B	00	0000	01F4	B	00	0000	02D5
T				T	000F		
FRCT	0			FRCT	0		
DP	031E			DP	031F		
Память данных							
031Eh	000F			031Eh	000F		

20.3.106 SQURS Smem, src

Операнды Smem: Операнд памяти данных одиночного доступа.
src: A (аккумулятор A).
B (аккумулятор B).

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	1	1	1	0	1	S	I	A	A	A	A	A	A	A

Выполнение (Smem) → T
(src) – (Smem) x (Smem) → src

Статусные биты На результат влияют OVM и FRCT.
Команда влияет на значение OVsrc.

Описание Данная команда сохраняет значение Smem в T, затем она вычисляет квадрат Smem и вычитает произведение из src. Результат записывается в src.

Слова 1 слово.
Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Циклы 1 цикл.
Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Классы Класс 3A.
Класс 3B.

Пример 1 SQURS 9, A

Перед командой				После команды			
A	00	014B	5DB0	A	00	0000	0320
T	8765			T	1234		
FRCT	0			FRCT	0		
DP	006			DP	006		
Память данных							
0309h	1234			0309h	1234		

Пример 2 SQURS *AR3, B

Перед командой				После команды			
B	00	014B	5DB0	B	00	0000	02D5
T	8765			T	1234		
FRCT	0			FRCT	0		
AR3	0309			AR3	0309		
Память данных							
0309h	1234			0309h	1234		

20.3.107 SRCCD Xmem, cond

Операнды Xmem: Операнд памяти данных двойного доступа.

В таблице перечислены условия (операнд cond) для данной команды.

Операнд условий	Описание	Код условия	Операнд условий	Описание	Код условия
AEQ	(A) = 0	0101	BEQ	(B) = 0	1101
ANEQ	(A) ≠ 0	0100	BNEQ	(B) ≠ 0	1100
AGT	(A) > 0	0110	BGT	(B) > 0	1110
AGEQ	(A) ≥ 0	0010	BGEQ	(B) ≥ 0	1010
ALT	(A) < 0	0011	BLT	(B) < 0	1011
ALEQ	(A) ≤ 0	0111	BLEQ	(B) ≤ 0	1111

Код

операции

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	1	1	0	1	X	X	X	X	C	O	N	D

Выполнение If (cond)

Then

(BRC) → Xmem

Else

(Xmem) → Xmem

Статусные биты Не используются.

Описание

Если условие истинно, то данная команда сохраняет содержимое счетика повторений блока (BRC) в Xmem. Если условие ложное, то команда считывает Xmem и записывает значение в Xmem обратно по тому же адресу; Таким образом, Xmem остается прежним. Независимо от условия, Xmem всегда считывается и обновляется.

Слова 1 слово.

Циклы 1 цикл.

Классы Класс 15.

Пример SRCCD *AR5-, AGT

Перед командой				После команды			
A	00	70FF	FFFF	A	00	70FF	FFFF
AR5	0202			AR5	0201		
BRC	4321			BRC	4321		
Память данных							
0202h	1234			0202h	4321		

20.3.108 SSBX N, SBIT

Операнды $0 \leq \text{SBIT} \leq 15$
 $N = 0$ or 1

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	N	1	1	0	1	1	S	B	I	T

Выполнение $1 \rightarrow \text{STN}(\text{SBIT})$

Статусные биты Не используются.

Описание Данная команда присваивает значение логической единицы указанному биту в статусном регистре 0 или 1. N назначает номер статусного регистра для внесения изменений, а SBIT задает бит для изменения. Название поля статусного регистра может быть использовано в качестве операнда вместо операндов N и SBIT (см. Пример 1).

Примечание – Данная команда не повторяемая.

Слова 1 слово.

Циклы 1 цикл.

Классы Класс 1.

Пример 1 SSBX SXM ; SXM means: N=1, SBIT=8

	Перед командой	После команды
ST1	34CD	35CD

Пример 2 SSBX 1,8

	Перед командой	После команды
ST1	34CD	35CD

20.3.109 ST

1 ST T, Smem

2 ST TRN, Smem

3 ST #lk, Smem

Операнды Smem: операнд памяти данных одиночного доступа.
 $-32\ 768 \leq lk \leq 32\ 767$

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1:	1	0	0	0	1	1	0	0	I	A	A	A	A	A	A	A
2:	1	0	0	0	1	1	0	1	I	A	A	A	A	A	A	A
3:	0	1	1	1	0	1	1	0	I	A	A	A	A	A	A	A
16 битная константа																

Выполнение	1: (T) → Smem 2: (TRN) → Smem 3: lk → Smem
Статусные биты	Не используются.
Описание	Данная команда сохраняет содержимое T, регистр перехода (TRN), или 16-разрядную константу lk в ячейке памяти данных Smem.
Слова	Синтаксисы 1 и 2: 1 слово. Синтаксис 3: 2 слова. Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.
Циклы	Синтаксисы 1 и 2: 1 цикл. Синтаксис 3: 2 цикла. Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.
Классы	Синтаксисы 1 and 2: Класс 10A. Синтаксисы 1 and 2: Класс 10B. Синтаксис 3: Класс 12A. Синтаксис 3: Класс 12B.
Пример 1	ST FFFFh, 0

Перед командой		После команды	
DP	004	DP	004
Память данных			
0200h	0101	0200h	FFFF

Пример 2 ST TRN, 5

Перед командой		После команды	
DP	004	DP	004
TRN	1234	TRN	1234
Память данных			
0205h	0030	0205h	1234

Пример 3 ST T, *AR7–

Перед командой		После команды	
T	4210	T	4210
AR7	0321	AR7	0320
Память данных			
0321h	1200	0321h	4210

20.3.110 STH

- 1 STH src, Smem
- 2 STH src, ASM, Smem
- 3 STH src, SHFT, Xmem
- 4 STH src [, SHIFT], Smem

Операнды	src: A (аккумулятор A).																																															
	B (аккумулятор B).																																															
	Smem: Операнд памяти данных одиночного доступа.																																															
	Xmem: Операнд памяти данных двойного доступа.																																															
	$0 \leq \text{SHFT} \leq 15$ $-16 \leq \text{SHIFT} \leq 15$																																															
Код операции	1:																																															
	<table><tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>S</td><td>I</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td></tr></table>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	1	0	0	0	0	0	1	S	I	A	A	A	A	A	A	A															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																
	1	0	0	0	0	0	1	S	I	A	A	A	A	A	A	A																																
	2:																																															
<table><tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>S</td><td>I</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td></tr></table>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	1	0	0	0	0	1	1	S	I	A	A	A	A	A	A	A																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																	
1	0	0	0	0	1	1	S	I	A	A	A	A	A	A	A																																	
3:																																																
<table><tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>S</td><td>X</td><td>X</td><td>X</td><td>X</td><td>S</td><td>H</td><td>F</td><td>T</td></tr></table>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	1	0	0	1	1	0	1	S	X	X	X	X	S	H	F	T																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																	
1	0	0	1	1	0	1	S	X	X	X	X	S	H	F	T																																	
4:																																																
<table><tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>I</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>S</td><td>0</td><td>1</td><td>1</td><td>S</td><td>H</td><td>I</td><td>F</td><td>T</td></tr></table>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	0	1	1	0	1	1	1	1	I	A	A	A	A	A	A	A	0	0	0	0	1	1	0	S	0	1	1	S	H	I	F	T
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																	
0	1	1	0	1	1	1	1	I	A	A	A	A	A	A	A																																	
0	0	0	0	1	1	0	S	0	1	1	S	H	I	F	T																																	
Выполнение	1: (src(31–16)) → Smem 2: (src) << (ASM – 16) → Smem 3: (src) << (SHFT – 16) → Xmem 4: (src) << (SHIFT – 16) → Smem																																															
Статусные биты	Результат зависит от SXM.																																															
Описание	Эта команда сохраняет старшую часть src (биты 31–16) в ячейке памяти данных Smem. src сдвигается влево (в соответствии с требованиями ASM, SHFT или SHIFT) и биты 31–16 сдвинутого значения записываются в память данных (Smem или Xmem). Если SXM = 0, бит 39 src копируется в самые старшие биты ячейки памяти данных. Если SXM = 1, значение src расширенное знаком (39 бит) сохраняется в ячейке памяти данных после сдвига вправо с использованием защитных бит. Аккумулятор src остается неизменным. Примечания – Следующие синтаксисы ассемблируются как различный синтаксис в определенных условиях. 1 Синтаксис 3: если SHFT = 0, код операции команды ассемблируется как Синтаксис 1. 2 Синтаксис 4: если SHIFT = 0, код операции команды ассемблируется как Синтаксис 1. Синтаксис 4: если $0 < \text{SHIFT} \leq 15$ косвенный адрес приравнивается к режиму Xmem, код операции команды ассемблируется как Синтаксис 3.																																															

Слова	Синтаксисы 1, 2 и 3: 1 слово. Синтаксис 4: 2 слова. Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.
Циклы	Синтаксисы 1, 2 и 3: 1 цикл. Синтаксис 4: 2 цикла. Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.
Классы	Синтаксисы 1, 2, и 3: Класс 10А. Синтаксисы 1 и 2: Класс 10В. Синтаксис 4: Класс 11А. Синтаксис 4: Класс 11В.
Пример 1	STH A, 10

Перед командой			После команды		
A	FF	8765 4321	A	FF	8765 4321
DP		004	DP		004
Память данных					
020Ah		1234	020Ah		8765

Пример 2 STH B, -8, *AR7-

Перед командой			После команды		
B	FF	8421 1234	B	FF	8421 1234
AR7		0321	AR7		0320
Память данных					
0321h		ABCD	0321h		FF84

Пример 3 STH A, -4, 10

Перед командой			После команды		
A	FF	8421 1234	A	FF	8421 1234
SXM		1	SXM		1
DP		004	DP		004
Память данных					
020Ah		7FFF	020Ah		F842

20.3.111 STL

- 1 STL src, Smem
- 2 STL src, ASM, Smem
- 3 STL src, SHFT, Xmem
- 4 STL src [, SHIFT], Smem

Операнды src: A (аккумулятор А).
B (аккумулятор В).

Smem: операнд памяти данных одиночного доступа.

Xmem: операнд памяти данных двойного доступа.

$0 \leq \text{SHFT} \leq 15$

$-16 \leq \text{SHIFT} \leq 15$

Код

операции

1:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0	S	I	A	A	A	A	A	A

2:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	1	0	0	S	I	A	A	A	A	A	A

3:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	1	0	0	S	X	X	X	X	S	H	F	T

4:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	1	1	1	I	A	A	A	A	A	A	A
0	0	0	0	1	1	0	S	1	0	0	S	H	I	F	T

Выполнение 1: (src(15–0)) → Smem

2: (src) << ASM → Smem

3: (src) << SHFT → Xmem

4: (src) << SHIFT → Smem

Статусные биты

Результат зависит от SXM.

Описание

Данная команда сохраняет младшую часть src (биты 15–0) в ячейку памяти данных Smem. Сдвигается влево (в соответствии с требованиями ASM, SHFT или SHIFT) и биты 15–0 сдвинутого значения помещаются в память данных (Smem или Xmem). Когда значение сдвига положительно, то в самые младшие биты вдвигаются нули.

Примечания

Следующие синтаксисы ассемблируются как различный синтаксис в неизменных условиях.

1 Синтаксис 3: если SHFT = 0, код операции команды ассемблируется как Синтаксис 1.

2 Синтаксис 4: если SHIFT = 0, код операции команды ассемблируется как Синтаксис 1.

Синтаксис 4: если $0 \leq \text{SHIFT} \leq 15$ косвенный модификатор приравняется к режиму Xmem, код операции команды ассемблируется как Синтаксис 3.

Слова	Синтаксисы 1, 2 и 3: 1 слово. Синтаксис 4: 2 слова. Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.
Циклы	Синтаксисы 1, 2 и 3: 1 цикл. Синтаксис 4: 2 цикла. Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.
Классы	Синтаксисы 1, 2 и 3: Класс 10А. Синтаксисы 1, 2 и 3: Класс 10В. Синтаксис 4: Класс 11А. Синтаксис 4: Класс 11В.
Пример 1	STL A, 11

Перед командой				После команды			
A	FF	8765	4321	A	FF	8765	4321
DP				DP			
Память данных							
020Bh				020Bh			
	1234				4321		

Пример 2 STL B, –8, *AR7–

Перед командой				После команды			
B	FF	8421	1234	B	FF	8421	1234
SXM	1			SXM	1		
AR7	004			AR7	004		
Память данных							
0321h	0099			0321h	2112		

Пример 3 STL A, 7, 11

Перед командой				После команды			
A	FF	8421	1234	A	FF	8421	1234
DP				DP			
Память данных							
020Bh				020Bh			
	0101				1A00		

20.3.112 STLM src, MMR

Операнды src: A (аккумулятор A).
B (аккумулятор B).
MMR: отображаемый в памяти регистр.

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	0	0	1	0	0	S	I	A	A	A	A	A	A	A

Выполнение (src(15–0)) → MMR

Статусные биты	Не используются.
Описание	Эта команда сохраняет младшую часть src (биты 15–0) в адресуемый регистр отображенный в памяти MMR. Значения девяти самых старших бит действительного адреса сбрасываются независимо от текущего значения DP или от старших девяти бит ARx. Эта команда позволяет сохранять src в любой ячейке памяти на нулевой странице данных без внесения изменений в поле DP статусного регистра ST0.
Слова	1 слово.
Циклы	1 цикл.
Классы	Класс 10А.
Пример 1	STLM A, BRC

Перед командой				После команды			
A	FF	8765	4321	A	FF	8765	4321
BRC(1Ah)			1234	BRC			4321

Пример 2 STLM B, *AR1–

Перед командой				После команды			
B	FF	8421	1234	B	FF	8421	1234
AR1			3F17	AR1			0016
AR7(17h)			0099	AR7			1234

20.3.113 STM #lk, MMR

Операнды MMR: Отображаемый в памяти регистр.
 $-32\ 768 \leq lk \leq 32\ 767$

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	1	1	1	0	1	1	1	I	A	A	A	A	A	A	A
16 битная константа																

Выполнение lk → MMR

Статусные биты Отсутствуют.

Описание Эта команда сохраняет 16-разрядную константу lk в регистр MMR или ячейку памяти на 0 странице данных без внесения изменений в поле DP статусного регистра ST0. Значения девяти самых старших битов действительного адреса сбрасываются независимо от текущего значения DP или от старших девяти битов ARx.

Слова 2 слова.

Циклы 2 цикла.

Классы Класс 12А.

Пример 1 STM 0FFFFh, IMR

Перед командой		После команды	
IMR	FF01	IMR	FFFF

Пример 2 STM 8765h, *AR7+

Перед командой		После команды	
AR0	0000	AR0	8765
AR7	8010	AR7	0011

20.3.114 ST src, Ymem || ADD Xmem, dst

Операнды src, dst: A (аккумулятор A).
B (аккумулятор B).
Xmem, Ymem: операнды памяти данных двойного доступа.
dst_: если dst = A, то dst_ = B; если dst = B, то dst_ = A

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	0	0	0	0	S	D	X	X	X	X	Y	Y	Y	Y

Выполнение (src) << (ASM - 16) → Ymem
(dst_) + (Xmem) << 16 → dst

Статусные биты Выполнение инструкции зависит от OVM, SXM и ASM.
Операция влияет на C и OVdst.

Описание Эта команда сохраняет src, сдвинутый на (ASM – 16) в ячейку памяти данных Ymem. Параллельно, эта команда складывает содержимое dst_ со сдвинутым влево на 16 разрядов операндом памяти данных Xmem, и сохраняет результат в dst. Если src равно dst, значение сохраняемое в Ymem является значением src до выполнения команды.

Слова 1 слово.

Циклы 1 цикл.

Классы Класс 14.

Пример ST A, *AR3
||ADD *AR5+0%, B

Перед командой				После команды			
A	FF	8421	1000	A	FF	8021	1000
B	00	0000	1111	B	FF	0422	1000
OVM	0			OVM	0		
SXM	1			SXM	1		
ASM	1			ASM	1		
AR0	0002			AR0	0002		
AR3	0200			AR3	0200		
AR5	0300			AR5	0302		
Память данных							
0200h	0101			0309h	0842		
0300h	8001			0309h	8001		

20.3.115 ST||LD

- 1 ST src, Ymem
|| LD Xmem, dst
- 2 ST src, Ymem
|| LD Xmem, T

Операнды

src, dst: A (аккумулятор A).
B (аккумулятор B).

Xmem, Ymem: операнды памяти данных двойного доступа.

Код операции

1:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	1	0	S	D	X	X	X	X	Y	Y	Y	Y

2:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0	0	1	S	0	X	X	X	X	Y	Y	Y	Y

- Выполнение
1. (src) << (ASM - 16) → Ymem
(Xmem) << 16 → dst
2. (src) << (ASM - 16) → Ymem
(Xmem) → T

Статусные биты

Выполнение зависит от значений OVM и ASM.
Команды влияет на значение C.

Описание

Эта команда сохраняет src, сдвинутое на (ASM – 16) разрядов в ячейку памяти данных Ymem. Параллельно, эта команда загружает 16 битный операнд памяти данных Xmem в dst или T. Если src равно dst, значение, сохраняемое в Ymem, является значением src до выполнения команды.

Слова

1 слово.

Циклы

1 цикл.

Классы

Класс 14.

Пример 1

ST B, *AR2–
||LD *AR4+, A

Перед командой				После команды			
A	00	0000	001C	A	FF	8001	0000
B	FF	8421	1234	B	FF	8421	1234
SXM	1			SXM	0		
ASM	1C			ASM	1		
AR2	01FF			AR2	01FE		
AR4	0200			AR4	0201		
Память данных							
01FFh	xxxx			01FFh	F842		
0200h	8001			0200h	8001		

Пример 2 ST A, *AR3
 ||LD *AR4, T

Перед командой				После команды			
A	FF	8421	1234	A	FF	8421	1234
T				T	80FF		
ASM	1			ASM	1		
AR3	0200			AR3	0200		
AR4	0100			AR4	0100		
Память данных							
0200h	0001			0200h	0842		
0100h	80FF			0100h	80FF		

Пример 3 ST A, *AR2+
 ||LD *AR2–, A
В примере 3, LD считывает операнд-источник из ячейки памяти, указанной AR2 прежде, чем ST делает запись в ту же ячейку. ST считывает операнд-источник аккумулятора A прежде, чем LD загрузит новое значение в аккумулятор A.

20.3.116 ST src, Ymem

|| MAC[R] Xmem, dst

Операнды	src, dst:	A (аккумулятор A). B (аккумулятор B).																
	Xmem, Ymem:	Операнды памяти данных двойного доступа.																
Код операции		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
		1	1	0	1	0	R	S	D	X	X	X	X	Y	Y	Y	Y	
Выполнение	(src << (ASM – 16)) → Ymem																	
	If (Rounding)																	
	Then																	
	Round ((Xmem) × (T) + (dst)) → dst																	
Статусные биты	Else																	
	(Xmem) → (T) + (dst) → dst																	
Описание	Выполнение зависит от OVM, SXM, ASM и FRCT.																	
	Результат влияет на значение C и OVdst.																	
Описание	Эта команда сохраняет src, сдвинутое на (ASM – 16) разрядов в ячейке памяти данных Ymem. Параллельно, эта команда умножает содержимое T на операнд памяти данных Xmem, складывает со значением dst (с округлением или без него) и сохраняет результат в dst. Если src равно dst, значение, сохраняемое в Ymem, является значением src до выполнения команды																	
	При использовании индекса R команда округляет результат операции умножения путем добавления 2 ¹⁵ к результату и сбрасыванием значений самых младших битов (биты 15–0).																	

Слова 1 слово.
Циклы 1 цикл.
Классы Класс 14.
Пример 1 ST A, *AR4–
||MAC *AR5, B

Перед командой				После команды			
A	00	0011	1111	A	00	0011	1111
B	00	0000	1111	B	00	010C	9511
T	0400			T	0400		
ASM	5			ASM	5		
FRCT	0			FRCT	0		
AR4	0100			AR4	00FF		
AR5	0200			AR5	0200		
Память данных							
100h	1234			100h	0222		
200h	4321			200h	4321		

Пример 2 ST A, *AR4+
||MACR *AR5+, B

Перед командой				После команды			
A	00	0011	1111	A	00	0011	1111
B	00	0000	1111	B	00	010D	0000
T	0400			T	0400		
ASM	1C			ASM	1C		
FRCT	0			FRCT	0		
AR4	0100			AR4	0101		
AR5	0200			AR5	0201		
Память данных							
100h	1234			100h	0001		
200h	4321			200h	4321		

20.3.117 ST src, Ymem || MAS[R] Xmem, dst

Операнды src, dst: A (аккумулятор A).
B (аккумулятор B).

Xmem, Ymem: Операнды памяти данных двойного доступа.

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	0	1	1	R	S	D	X	X	X	X	Y	Y	Y	Y

Выполнение $(src \ll (ASM - 16)) \rightarrow Ymem$
If (Rounding)
 Then
 $Round ((dst) - (Xmem) \times (T)) \rightarrow dst$
Else
 $(dst) - (Xmem) \times (T) \rightarrow dst$

Статусные биты Выполнение зависит от значения OVM, SXM, ASM и FRCT.
Команда влияет на значение C и OVdst.

Описание Эта команда сохраняет src, сдвинутое на (ASM – 16) разрядов в ячейке памяти данных Ymem. Параллельно, эта команда умножает содержимое T на операнд памяти данных Xmem, вычитает результат из dst (с округлением или без него) и сохраняет результат в dst. Если src равно dst, значение, сохраняемое в Ymem, является значением src до выполнения команды
При использовании индекса R команда округляет результат операции умножения путем добавления 2¹⁵ к результату и сбрасыванием значений самых младших битов (биты 15–0).

Слова 1 слово.

Циклы 1 цикл.

Классы Класс 14.

Пример 1 ST A, *AR4+
 ||MAS *AR5, B

Перед командой				После команды			
A	00	0011	1111	A	00	0011	1111
B	00	0000	1111	B	FF	FEF3	8D11
T	0400			T	0400		
ASM	5			ASM	5		
FRCT	0			FRCT	0		
AR4	0100			AR4	0101		
AR5	0200			AR5	0200		
Память данных							
0100h	1234			0100h	0222		
0200h	4321			0200h	4321		

Пример 2 ST A, *AR4+
 ||MASR *AR5+, B

Перед командой				После команды					
A	00	0011	1111	A	00	0011	1111		
B	00	0000	1111	B	FF	FEF4	0000		
T				0400	T				0400
ASM				0001	ASM				0001
FRCT				0	FRCT				0
AR4				0100	AR4				0101
AR5				0200	AR5				0201
Память данных									
0100h				1234	0100h				0022
0200h				4321	0200h				4321

20.3.118 ST src, Ymem || MPY Xmem, dst

Операнды src, dst: A (аккумулятор A).
B (аккумулятор B).

Xmem, Ymem: Операнды памяти данных двойного доступа.

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	0	0	1	1	S	D	X	X	X	X	Y	Y	Y	Y

Выполнение (src << (ASM - 16)) → Ymem
(T) x (Xmem) → dst

Статусные биты Выполнение находится под влиянием OVM, SXM, ASM и FRCT.
Команда влияет на C и OVdst.

Описание Эта команда сохраняет src, сдвинутое на (ASM – 16) разрядов в ячейке памяти данных Ymem. Параллельно, эта команда умножает содержимое T на операнд памяти данных двойного доступа Xmem и сохраняет результат в dst. Если src равно dst, значение, сохраняемое в Ymem, является значением src до выполнения команды.

Слова 1 слово.

Циклы 1 цикл.

Классы Класс 14.

Пример 1 ST A, *AR3+
||MPY *AR5+, B

Перед командой				После команды			
A	FF	8421	1234	A	FF	8421	1234
B	xx	xxxx	xxxx	B	00	2000	0000
T			4000	T			4000
ASM			00	ASM			00
FRCT			1	FRCT			1
AR3			0200	AR3			0201
AR5			0300	AR5			0301
Память данных							
0200h			1111	0200h			8421
0300h			4000	0300h			4000

20.3.119 ST src, Ymem || SUB Xmem, dst

Операнды src, dst: A (аккумулятор A).
B (аккумулятор B).

Xmem, Ymem: Операнды памяти данных двойного доступа
dst_ : если dst = A, то dst_ = B; если dst = B, то dst_ = A.

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	0	0	0	1	S	D	X	X	X	X	Y	Y	Y	Y

Выполнение (src << (ASM - 16)) → Ymem
(Xmem) << 16 – (dst_) → dst

Статусные биты Результат операции зависит от OVM, SXM и ASM.
Операция влияет на C и OVdst.

Описание	Эта команда сохраняет src, сдвинутое на (ASM – 16) разрядов в ячейке памяти данных Ymem. Параллельно, эта команда вычитает содержимое dst_ из 16-разрядного операнда памяти данных двойного доступа Xmem, сдвинутого влево на 16 бит и сохраняет результат в dst. Если src равно dst, значение, сохраняемое в Ymem, является значением src до выполнения команды.
Слова	1 слово.
Циклы	1 цикл.
Классы	Класс 14.
Пример 1	ST A, *AR3– SUB *AR5+0%, B

Перед командой				После команды			
A	FF	8421	0000	A	FF	8421	0000
B	00	1000	0001	B	FF	FBE0	0000
ASM	01			ASM	01		
SXM	1			SXM	1		
AR0	0002			AR0	0002		
AR3	01FF			AR3	01FE		
AR5	0300			AR5	0302		
Память данных							
01FFh	1111			01FFh	0842		
0300h	8001			0300h	8001		

20.3.120 STRCD Xmem, cond

Операнды	Xmem: Операнд памяти данных двойного доступа. В таблице перечислены условия (операнд cond) для данной команды.

Операнд условий	Описание	Код условия	Операнд условий	Описание	Код условия
AEQ	(A) = 0	0101	BEQ	(B) = 0	1101
ANEQ	(A) ≠ 0	0100	BNEQ	(B) ≠ 0	1100
AGT	(A) > 0	0110	BGT	(B) > 0	1110
AGEQ	(A) ≥ 0	0010	BGEQ	(B) ≥ 0	1010
ALT	(A) < 0	0011	BLT	(B) < 0	1011
ALEQ	(A) ≤ 0	0111	BLEQ	(B) ≤ 0	1111

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	0	1	1	1	0	0	X	X	X	X	C	0	N	D

Выполнение	If (cond) (T) → Xmem Else (Xmem) → Xmem

Статусные биты	Не используются.

Описание	Если условие истинно, то команда сохраняет содержимое Т в ячейке памяти данных Хmem. Если условие ложное, то команда считывает Хmem и записывает значение Хmem обратно по тому же адресу; таким образом, Хmem остается прежним. Независимо от условия Хmem всегда считывается и обновляется.
Слова	1 слово.
Циклы	1 цикл.
Классы	Класс 15.
Пример 1	STRCD *AR5–, AGT

Перед командой				После команды			
A	00	70FF	FFFF	A	00	70FF	FFFF
T	4321			T	4321		
AR5	0202			AR5	0201		
Память данных							
0202h	1234			0202h	4321		

20.3.121 SUB

- 1 SUB Smem, src
- 2 SUB Smem, TS, src
- 3 SUB Smem, 16, src [, dst]
- 4 SUB Smem [, SHIFT], src [, dst]
- 5 SUB Xmem, SHFT, src
- 6 SUB Xmem, Ymem, dst
- 7 SUB #lk [, SHFT], src [, dst]
- 8 SUB #lk, 16, src [, dst]
- 9 SUB src [, SHIFT], [, dst]
- 10 SUB src, ASM [, dst]

Операнды	src, dst:	A (аккумулятор A). B (аккумулятор B).
	Smem:	операнд памяти данных одиночного доступа.
	Xmem, Ymem:	операнды памяти данных двойного доступа.
		$-32\ 768 \leq lk \leq 32\ 767$
		$0 \leq SHFT \leq 15$
		$-16 \leq SHIFT \leq 15$

Код

операции

1:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	S	I	A	A	A	A	A	A	A

2:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	1	0	S	I	A	A	A	A	A	A	A

3:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	S	D	I	A	A	A	A	A	A	A

4:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	1	1	1	I	A	A	A	A	A	A	A
0	0	0	0	1	1	S	D	0	0	1	S	H	I	F	T

5:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	0	0	1	D	X	X	X	X	Y	Y	Y	Y

6:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	S	D	0	0	0	1	S	H	F	T
16 битная константа															

7:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	S	D	0	0	0	1	S	H	F	T
16 битная константа															

8:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	S	D	0	1	1	0	0	0	0	1
16 битная константа															

9:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	S	D	0	0	1	S	H	I	F	T

10:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	1	S	D	1	0	0	0	0	0	0	1

Выполнение

- 1: (src) – (Smem) → src
- 2: (src) – (Smem) << TS → src
- 3: (src) – (Smem) << 16 → dst
- 4: (src) – (Smem) << SHIFT → dst
- 5: (src) – (Xmem) << SHFT → src
- 6: (Xmem) << 16 – (Ymem) << 16 → dst
- 7: (src) – lk << SHFT → dst
- 8: (src) – lk << 16 → dst
- 9: (dst) – (src) << SHIFT → dst
- 10: (dst) – (src) << ASM → dst

Статусные биты

Результат зависит от SXM и OVM.
Команда влияет на C и OVdst (или OVsrc, если dst = src).
Для Синтаксиса 3: если в результате вычитания возникает заем, то значение бита переноса C обнуляется. Для других синтаксисов C не зависит от результата.

Описание	Данная команда вычитает 16-разрядное значение из содержимого выбранного аккумулятора или из 16-разрядного операнда $Xmem$ в режиме двойной адресации памяти данных. Вычитаемым 16-разрядным значением может быть одно из следующих: 1 содержимое операнда памяти данных одиночного доступа ($Smem$); 2 содержимое операнда памяти данных двойного доступа ($Ymem$); 3 16-разрядный непосредственный операнд ($\#lk$); 4 значение сдвига в src. Если dst задан, то команда сохраняет результат в dst. Если dst не определен, то команда сохраняет результат в src. Большинство вторых операндов может быть сдвинуто. При сдвиге влево: 1 значения младших битов сбрасываются; 2 старшие биты: – дополняются знаком, если $SXM = 1$; – сбрасываются, если $SXM = 0$. При сдвиге вправо, старшие биты: – дополняются знаком, если $SXM = 1$; – сбрасываются, если $SXM = 0$.
Слова	Примечания Следующие синтаксисы ассемблируются в разные синтаксисы в соответствующих условиях. Синтаксис 4: если $dst = src$ и $SHIFT = 0$, код операции команды ассемблируется как Синтаксис 1. Синтаксис 4: если $dst = src$, $SHIFT \leq 15$, и режим косвенной адресации $Smem$ включен в $Xmem$, то код операции команды ассемблируется как Синтаксис 1. Синтаксисы 1, 2, 3, 5, 6, 9 и 10: 1 слово. Синтаксисы 4, 7 и 8: 2 слова. Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с $Smem$.
Циклы	Синтаксисы 1, 2, 3, 5, 6, 9 и 10: 1 цикл. Синтаксисы 4, 7 и 8: 2 цикла. Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с $Smem$.
Классы	Синтаксисы 1, 2, 3 и 5: Класс 3А. Синтаксисы 1, 2 и 3: Класс 3В. Синтаксис 4: Класс 4А. Синтаксис 4: Класс 4В. Синтаксис 6: Класс 7. Синтаксисы 7 и 8: Класс 2. Синтаксисы 9 и 10: Класс 1.
Пример 1	SUB *AR1+, 14, A

Перед командой				После команды			
A	00	0000	1200	A	FF	FAC0	1200
C			x	C			0
SXM			1	SXM			1
AR1			0100	AR1			0101
Память данных							
0100h			1500	0100h			1500

Пример 2 SUB A, –8, B

Перед командой				После команды			
A	00	0000	1200	A	00	0000	1200
B	00	0000	1800	B	00	0000	17EE
C				x			
SXM				1			

Пример 3 SUB #12345, 8, A, B

Перед командой				После команды			
A	00	0000	1200	A	00	0000	1200
B	00	0000	1800	B	FF	FFCF	D900
C				x			
SXM				1			

Пример 4 ST B, *AR2–
||LD *AR4+, A

A	00	0000	001C
B	FF	8421	1234
SXM	1		
ASM	1C		
AR2	01FF		
AR4	0200		

A	00	4214	1414
B	FF	8421	1234
SXM	1		
ASM	1C		
AR2	01FE		
AR4	0201		

Память данных

01FFh	xxxx
0200h	8001

01FFh	F842
0200h	8001

Пример 5 ST A, *AR3
||LD *AR4, T

Перед командой				После команды			
A	FF	8421	1234	A	FF	8421	1234
T	3456			T	80FF		
ASM	1			ASM	1		
AR3	0200			AR3	0200		
AR4	0100			AR4	0100		
Память данных							
0200h	0001			0200h	0842		
0100h	80FF			0100h	80FF		

Пример 6 ST A, *AR2+
||LD *AR2–, A

В примере 6, LD читает операнд-источник из ячейки памяти указанной AR2 до того, как ST осуществит запись в ту же ячейку. ST считывает операнд-источник аккумулятора A прежде, чем LD осуществляет загрузку аккумулятора A.

20.3.122 SUBB Smem, src

Операнды src: A (аккумулятор A).
B (аккумулятор B).

Smem: операнд памяти данных одиночного доступа.

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	1	1	1	D	I	A	A	A	A	A	A	A

Выполнение (src) – (Smem) – (логическое отрицание C) → src

Статусные биты Результат зависит от OVM и C.

Команда влияет на C и OVsrc.

Описание Эта команда вычитает из src содержимое 16-разрядного операнда Smem и логическую инверсию бита переноса C (то есть заем), без расширения знака.

Слова 1 слово.

Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Циклы 1 цикл.

Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Классы Класс 3A.

Класс 3B.

Пример 1 SUBB 5, A

Перед командой				После команды			
A	00 0000 0006			A	FF FFFF FFFF		
C	0			C	0		
DP	008			DP	008		
Память данных							
0405h	0006			0405h	0006		

Пример 2 SUBB *AR1+, B

Перед командой				После команды					
B	FF	8000	0006	B	FF	8000	0000		
C				1	C				1
OVM				1	OVM				1
AR1				0405	AR1				0406
Память данных									
0405h				0006	0405h				0006

20.3.123 SUBC Smem, src

Операнды	Smem: Операнд памяти данных одиночного доступа. src: A (аккумулятор A). B (аккумулятор B).																																
Код операции	<table><tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>S</td><td>I</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td></tr></table>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	0	0	0	1	1	1	1	S	I	A	A	A	A	A	A	A
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																		
0	0	0	1	1	1	1	S	I	A	A	A	A	A	A	A																		
Выполнение	(src) – ((Smem) << 15) → ALU output If ALU output ≥ 0 Then ((ALU output) << 1) + 1 → src Else (src) << 1 → src																																
Статусные биты	Выполнение команды зависит от SXM. Команда влияет на C и OVsrc.																																
Описание	Данная команда вычитает из содержимого src 16-разрядный операнд памяти данных одиночного доступа Smem, сдвинутый влево на 15 бит. Если результат больше 0, он сдвигается на 1 бит влево, к результату добавляется 1, и результат сохраняется в src. В противном случае, данная команда сдвигает содержимое src на 1 бит влево и сохраняет результат в src. Команда поддерживает деление чисел. Предполагается, что делитель и делимое в этой команде имеют положительное значение. Бит SXM влияет на выполнение операции следующим образом: если SXM = 1, делитель должен иметь нулевое значение старшего бита. если SXM = 0, любое 16-разрядное значение делителя дает ожидаемый результат. Делимое, находящееся в src, изначально должно быть положительным (значение бита 31 равно 0) и должно оставаться положительным после сдвига аккумулятора, которое происходит на первом этапе выполнения команды. Эта команда влияет на OVA или OVB (в зависимости от src) но не зависит от OVM; поэтому, src не создает положительных или отрицательных насыщений во время выполнения команды.																																
Слова	1 слово. Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.																																
Циклы	1 цикл. Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.																																
Классы	Класс 3A. Класс 3B.																																
Пример 1	SUBC 2, A																																

Перед командой			После команды		
A	00	0000 0004	A	00	0000 0008
C		x	C		0
DP		006	DP		006
Память данных					
0302h		0001	0302h		0001

Пример 2 RPT #15
SUBC *AR1, B

Перед командой				После команды			
В	00	0000	0041	В	00	0002	0009
С	x			С	1		
AR1	1000			AR1	1000		
Память данных							
1000h	0007			1000h	0007		

20.3.124 SUBS Smem, src

Операнды Smem: Операнд памяти данных одиночного доступа.
src: A (аккумулятор A).
B (аккумулятор B).

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	0	0	0	1	0	1	S	I	A	A	A	A	A	A	A

Выполнение (src) – без знака (Smem) → src

Статусные биты Изменяется под влиянием OVM.
Влияет на C и OVsrc.

Описание Команда вычитает содержимое 16-разрядного операнда Smem из содержания src. Smem считается 16-разрядным числом без знака независимо от значения SXM. Результат сохраняется в src.

Слова 1 слово.
Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Циклы 1 цикл.
Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem

Классы Класс 3A.
Класс 3B.

Пример SUBS *AR2-, B

Перед командой				После команды			
В	00 0000 0002			В	FF FFFF 0FFC		
С	x			С	0		
AR2	0100			AR2	00FF		
Память данных							
0100h	F006			0100h	F006		

20.3.125 TRAP K

Операнды $0 \leq K \leq 31$

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	1	1	0	1	0	0	1	1	0	K	K	K	K	K

Выполнение (SP) - 1 → SP
(PC) + 1 → TOS
Вектор прерывания, определяемый K → PC

Статусные биты Не используются.

Описание Эта команда передает управление на подпрограмму обработки прерывания по вектору, определенному операндом K. Команда позволяет использовать программное обеспечение для выполнения любой программы обработки прерываний. Перечень прерываний и соответствующих им значений K см. в Приложении В.

Данная команда записывает значение PC + 1 в ячейку памяти данных, адресованную SP. Это позволяет реализовать возврат процессора на прерванную программу после обработки прерывания путем загрузки счетчика команд значением из ячейки памяти данных, адресованную SP. Команда является немаскируемой и не зависит от INTM, а также не влияет на INTM.

Примечание – Данная команда не повторяемая.

Слова 1 слово.

Циклы 3 цикла.

Классы Класс 35.

Пример 1 TRAP 10h

Перед командой		После команды	
PC	1233	PC	FFC0
SP	03FF	SP	03FE
Память данных			
03FEh	9653	03FEh	1234

20.3.126 WRITA Smem

Операнды Smem: операнд памяти данных одиночного доступа.

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	1	1	1	1	1	1	1	I	A	A	A	A	A	A	A

Выполнение A → PAR
If (RC) ≠ 0
Then
 (Smem) → (Pmem addressed by PAR)
 (PAR) + 1 → PAR
 (RC) - 1 → RC
Else
 (Smem) → (Pmem addressed by PAR)

Статусные биты	Не используются.
Описание	Данная команда передает слово из ячейки памяти данных, указанной Smem, в ячейку памяти программ. Адрес памяти программ определяется аккумулятором A(15-0). Данная команда может быть использована как повторяющаяся для перемещения последовательных слов (с использованием косвенной адресации) в непрерывное пространство памяти программ, адресованное PAR, путем инкрементирования PAR. Исходное значение устанавливается 16-тью младшими битами аккумулятора A. Блоки источника и приемника не обязательно должны быть встроенными или располагаться вне кристалла. Как только конвейер повторений начинает работу, команда становится одноцикловой. Содержимое аккумулятора A не зависит от данной команды.
Слова	1 слово. Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.
Циклы	5 циклов. Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.
Классы	Класс 26A. Класс 26B.
Пример	WRITA 5

Перед командой			После команды		
A	00 0000 0257		A	00 0000 0257	
DP		032	DP		032
Память программ					
0257h		0306	0257h		4339
Память данных					
1005h		4339	1005h		4339

20.3.127 XC n, cond [, cond [, cond]]

Операнды	n = 1 or 2
	В таблице перечислены условия (операнд условий) для данной команды.

Операнд условий	Описание	Код условия	Операнд условий	Описание	Код условия
BIO	$\overline{\text{BIO}}_{\text{Low}}$	0000 0011	NBIO	$\overline{\text{BIO}}_{\text{high}}$	0000 0010
C	C=1	0000 1100	C	C=0	0000 1000
TC	TC=1	0011 0000	TC	TC=0	0010 0000
AEQ	(A) = 0	0100 0101	BEQ	(B) = 0	0100 1101
ANEQ	(A) ≠ 0	0100 0100	BNEQ	(B) ≠ 0	0100 1100
AGT	(A) > 0	0100 0110	BGT	(B) > 0	0100 1110
AGEQ	(A) ≥ 0	0100 0010	BGEQ	(B) ≥ 0	0100 1010
ALT	(A) < 0	0100 0011	BLT	(B) < 0	0100 1011
ALEQ	(A) ≤ 0	0100 0111	BLEQ	(B) ≤ 0	0100 1111
AOV	A переполнение	0111 0000	BOV	B переполнение	0111 1000
ANOV	A отсутствие переполнения	0110 0000	BNOV	B отсутствие переполнения	0110 1000
UNC	Безусловный	0000 0000			

Код	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
операции	1	1	1	1	1	1	N	1	C	C	C	C	C	C	C	C

Выполнение	Синтаксис	Код n
	1	0
	2	1

If (cond)

Then

Next n instructions are executed

Else

Execute NOP for next n instructions

Статусные биты Не используются.

Описание Выполнение этой команды зависит от значения n и выбранных условий: если n = 1 и условие(я) удовлетворяется, то выполняется однословная команда, следующая за данной командой.

если n = 2 и условие(я) удовлетворяется, то выполняется одна двухсловная или две однословные команды, следующие за данной командой.

если условие(я) не удовлетворяется, то 1 или 2 инструкции, в зависимости от значения n, следующие за данной инструкцией, не выполняются.

Данная команда осуществляет проверку множественных условий перед выполнением и может осуществлять проверку условий как по одному, так в сочетании с другими. Можно сочетать условия только из одной группы:

Группа 1: можно выбрать не более двух условий. Каждое из этих условий должно принадлежать различным категориям (категория А или В); нельзя выбрать два условия из одной категории. Например, можно проверить EQ и OV одновременно, но нельзя проверить одновременно GT и NEQ. Аккумулятор должен быть одним для обоих условий; нельзя осуществлять проверку условий для двух аккумуляторов в рамках одной команды. Например, можно проверить AGT и AOV одновременно, но нельзя проверить одновременно AGT и BOV.

Группа 2: можно выбрать не более трех условий. Каждое из этих условий должно принадлежать разным категориям (категория А, В или С); нельзя выбрать два условия из одной категории. Например, можно осуществить проверку ТС, С и BIO одновременно, но нельзя одновременно проверять NTC, С и NC.

Условия для данной команды

Группа 1		Группа 2		
Категория А	Категория В	Категория А	Категория В	Категория С
EQ	OV	TC	C	BIO
NEQ	NOV	NTC	NC	NBIO
LT				
LEQ				
GT				
GEO				

Эта команда и два слова команды, следующие за ней, непрерываемые.

Примечание – Условия тестируются до того, как команда выполнится. Поэтому, если одна двухсловная или две однословные команды изменяют анализируемые условия, то это никак не повлияет на выполнение данной команды.

Данная команда не повторяемая.

Слова 1 слово.
Циклы 1 цикл.
Классы Класс 1.
Пример XC 1, ALEQ
MAR *AR1+
ADD A, DAT100

Перед командой				После команды			
A	FF	FFFF	FFFF	A	FF	FFFF	FFFF
AR1	0032			AR1	0033		

20.3.128 XOR

- 1 XOR Smem, src
- 2 XOR #lk [, SHFT], src [, dst]
- 3 XOR #lk, 16, src [, dst]
- 4 XOR src [, SHIFT] [, dst]

Операнды src, dst: A (аккумулятор A).
B (аккумулятор B).
Smem: Операнд памяти данных одиночного доступа.
 $0 \leq SHFT \leq 15$
 $-16 \leq SHIFT \leq 15$
 $0 \leq lk \leq 65\,535$

Код операции 1:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	1	0	S	I	A	A	A	A	A	A	A

2:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	S	D	0	1	0	1	S	H	F	T
16 битная константа															

3:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	S	D	0	1	1	0	0	1	0	1
16 битная константа															

4:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	S	D	1	1	0	S	H	I	F	T

Выполнение 1: (Smem) XOR (src) → src
2: lk << SHFT XOR (src) → dst
3: lk << 16 XOR (src) → dst
4: (src) << SHIFT XOR (dst) → dst
Статусные биты Не используются.

Описание	Эта команда выполняет операцию исключающего ИЛИ над содержанием ячейки памяти данных Smem (со сдвигом, указанным в команде) и содержимым выбранного аккумулятора, и сохраняет результат в dst или src, как указано. При сдвиге влево значения младших битов сбрасываются, а старшие биты не расширяются знаком. При сдвиге вправо знак не расширяется.
Слова	Синтаксисы 1 и 4: 1 слово. Синтаксисы 2 и 3: 2 слова. Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.
Циклы	Синтаксисы 1 и 4: 1 цикл. Синтаксисы 2 и 3: 2 цикла. Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.
Классы	Синтаксис 1: Класс 3А. Синтаксис 1: Класс 3В. Синтаксисы 2 и 3: Класс 2. Синтаксис 4: Класс 1.
Пример 1	XOR *AR3+, A

Перед командой				После команды			
A	00 00FF 1200			A	00 00FF 0700		
AR3	0100			AR3	0101		
Память данных							
0100h	1500			0100h	1500		

Пример 2 XOR A, +3, B

Перед командой				После команды			
A	00	0000	1200	A	00	0000	1200
B	00	0000	1800	B	00	0000	8800

20.3.129 XORM #lk, Smem

Операнды Smem: операнд памяти данных одиночного доступа.
 $0 \leq lk \leq 65535$

Код операции	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0	1	1	0	1	0	1	0	I	A	A	A	A	A	A	A

Выполнение lk XOR (Smem) → Smem

Статусные биты Не используются.

Описание Эта команда выполняет операцию исключающего ИЛИ над содержанием ячейки памяти данных Smem и 16-разрядной константой lk. Результат записывается в Smem.

Примечание – Данная команда не повторяется.

Слова 2 слова.

Необходимо добавить 1 слово при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Циклы 2 цикла.

Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Классы Класс 18A.

Класс 18B.

Пример XORM 0404h, *AR4–

Перед командой		После команды	
AR4	0100	AR4	00FF
Память данных			
0100h	4444	0100h	4040

21 Контроллер McBSP (DSP)

McBSP обеспечивает:

- полнодуплексный режим работы;
- буферное FIFO на восемь слов по проему и передаче;
- независимую кадровую и битовую синхронизация по приему и передаче;
- возможность прямого подключения к стандартизованным промышленным кодекам, аналоговым интерфейсным приборам, последовательным ЦАП и АЦП приборам и прочим последовательным устройствам;
- возможность выбора между внешней и внутренними источниками кадровой и тактовой синхронизации.

Также в McBSP имеются следующие возможности:

- прямое подключение к:
 - T1/E1;
 - MVIP и ST-BUS совместимые устройства, включая:
 - формирователи MVIP;
 - формирователи H.100;
 - формирователи SCSA.
 - IOM-2 совместимые устройства;
 - AC97 совместимые устройства (обеспечивается необходимая многофазная синхронизация);
 - IIS совместимые устройства;
 - устройства SPI.
- многоканальные прием и передача до 128 каналов;
- произвольная длина слова от 4-х до 32-х бит включительно;
- компадирование/декомпадирование по m- и A-законам;
- реверсивный порядок передачи бит в слове (MSB или LSB);
- программируемая полярность для кадровой и битовой синхронизации;
- программируемые битовая и кадровая синхронизации;
- частота битовой синхронизации до 50 МГц.

21.1 Общее описание McBSP

С точки зрения подключения McBSP состоит из блока данных и блока управления. Структурно McBSP разделен на следующие блоки:

- управления;
- формирования внутренней кадровой и битовой синхронизации;
- передачи;
- приема.

Внешние устройства подключаются через последовательные трехпроводные интерфейсы независимые для приемника и передатчика.

BSPx_TFR – сигнал кадровой синхронизации передатчика

BSPx_TCLK – сигнал битовой синхронизации передатчика

BSPx_TX – сигнал данных передатчика

BSPx_RFR – сигнал кадровой синхронизации приемника

BSPx_RCLK – сигнал битовой синхронизации приемника

BSPx_RX – сигнал данных приемника

BSPx_RTFR – сигнал кадровой синхронизации совмещенный для приемника и передатчика

BSPx_RTCLK – сигнал битовой синхронизации совмещенный для приемника и передатчика

Для обмена данными с процессором в McBSP присутствует параллельный интерфейс.

ЦП или контроллер DMA читают принятые данные через регистр принимаемых данных (DRRL и DDRH) и записывает данные на передачу через регистр передаваемых данных (DXRL и DXRH). Данные записанные в DXRL и DXRH проходя через FIFO передатчика XBR выдвигаются через вывод DX из сдвигового регистра передатчика XSR. Аналогично данные принимаемые через вывод DR задвигаются в сдвиговый регистр приемника RSR откуда попадают в FIFO приемника RBR, далее копируются в DRRL и DRRH и могут быть считаны ЦП или DMA-контроллером. Данный способ позволяет осуществлять одновременный прием и передачу данных и обмен с управляющим устройством.

Обмен данными между McBSP и ЦП или DMA-контроллером может осуществляться байтами (8 бит) словами (16 бит), двойными словами (32 бит) (выбирается управляющим устройством), при этом выбор записывается слово атомарно (единым словом) или набирается из слов меньшего размера управляется через регистры управления McBSP.

Остальные регистры McBSP предназначены для задания режима работы McBSP, таких как управление режимами кадровой и битовой синхронизации, многоканального управления, кодирования, декодирования данных и пр.

Кроме того, в McBSP присутствуют статусные выходы для информирования ЦП или контроллера DMA о различных событиях и посредством программируемых выводов прерываний, независимых для приемника и передатчика.

21.2 Регистры McBSP

21.2.1 McBSP1

Таблица 21-1 – Регистр McBSP1

Адрес RISC	Адрес DSP	Значение SPSA	Название	Описание
0x3000_0040	0x0020	-	DRRL	Регистр приемника (младшее слово)
0x3000_0042	0x0021	-	DRRH	Регистр приемника (старшее слово)
0x3000_0044	0x0022	-	DXRL	Регистр передатчика (младшее слово)
0x3000_0046	0x0023	-	DXRH	Регистр передатчика (старшее слово)
0x3000_0048	0x0024	-	SPSA*	Регистр номера управляющего регистра

Адрес RISC	Адрес DSP	Значение SPSA	Название	Описание
0x3000_004C	0x0026	0x00	SPCRL	Регистр общего управления
0x3000_004C	0x0026	0x01	SPCRH	Регистр общего управления
0x3000_004C	0x0026	0x02	RCRL	Регистр управления приемником
0x3000_004C	0x0026	0x03	RCRH	Регистр управления приемником
0x3000_004C	0x0026	0x04	XCRL	Регистр управления передатчиком
0x3000_004C	0x0026	0x05	XCRH	Регистр управления передатчиком
0x3000_004C	0x0026	0x06	SRGRL	Регистр управления генераторами кадровой и битовой синхронизации
0x3000_004C	0x0026	0x07	SRGRH	Регистр управления генераторами кадровой и битовой синхронизации
0x3000_004C	0x0026	0x08	MCRL	Регистр управления многоканальными режимами приемника
0x3000_004C	0x0026	0x09	MCRH	Регистр управления многоканальными режимами передатчика
0x3000_004C	0x0026	0x0A	RCERL	Регистр маски приемника
0x3000_004C	0x0026	0x0B	RCERH	Регистр маски приемника
0x3000_004C	0x0026	0x0C	XCERL	Регистр маски передатчика
0x3000_004C	0x0026	0x0D	XCERH	Регистр маски передатчика
0x3000_004C	0x0026	0x0E	PCR	Регистр управления выводами
0x3000_004C	0x0026	0x0F	SPSR	Регистр состояния

21.2.2 McBSP2

Таблица 21-2 – Регистр McBSP2

Адрес RISC	Адрес DSP	Значение SPSA	Название	Описание
0x3000_0050	0x0028	-	DRRL	Регистр приемника (младшее слово)
0x3000_0052	0x0029	-	DRRH	Регистр приемника (старшее слово)
0x3000_0054	0x002A	-	DXRL	Регистр передатчика (младшее слово)
0x3000_0056	0x002B	-	DXRH	Регистр передатчика (старшее слово)
0x3000_0058	0x002C	-	SPSA*	Регистр номера управляющего регистра
0x3000_005C	0x002E	0x00	SPCRL	Регистр общего управления
0x3000_005C	0x002E	0x01	SPCRH	Регистр общего управления
0x3000_005C	0x002E	0x02	RCRL	Регистр управления приемником
0x3000_005C	0x002E	0x03	RCRH	Регистр управления приемником
0x3000_005C	0x002E	0x04	XCRL	Регистр управления передатчиком
0x3000_005C	0x002E	0x05	XCRH	Регистр управления передатчиком
0x3000_005C	0x002E	0x06	SRGRL	Регистр управления генераторами кадровой и битовой синхронизации
0x3000_005C	0x002E	0x07	SRGRH	Регистр управления генераторами кадровой и битовой синхронизации

Адрес RISC	Адрес DSP	Значение SPSA	Название	Описание
0x3000_005C	0x002E	0x08	MCRL	Регистр управления многоканальными режимами приемника
0x3000_005C	0x002E	0x09	MCRH	Регистр управления многоканальными режимами передатчика
0x3000_005C	0x002E	0x0A	RCERL	Регистр маски приемника
0x3000_005C	0x002E	0x0B	RCERH	Регистр маски приемника
0x3000_005C	0x002E	0x0C	XCERL	Регистр маски передатчика
0x3000_005C	0x002E	0x0D	XCERH	Регистр маски передатчика
0x3000_005C	0x002E	0x0E	PCR	Регистр управления выводами
0x3000_005C	0x002E	0x0F	SPSR	Регистр состояния

21.2.3 McBSP3

Таблица 21-3 – Регистр McBSP3

Адрес RISC	Адрес DSP	Значение SPSA	Название	Описание
0x3000_0060	0x0030	-	DRRL	Регистр приемника (младшее слово)
0x3000_0062	0x0031	-	DRRH	Регистр приемника (старшее слово)
0x3000_0064	0x0032	-	DXRL	Регистр передатчика (младшее слово)
0x3000_0066	0x0033	-	DXRH	Регистр передатчика (старшее слово)
0x3000_0068	0x0034	-	SPSA*	Регистр номера управляющего регистра
0x3000_006C	0x0036	0x00	SPCRL	Регистр общего управления
0x3000_006C	0x0036	0x01	SPCRH	Регистр общего управления
0x3000_006C	0x0036	0x02	RCRL	Регистр управления приемником
0x3000_006C	0x0036	0x03	RCRH	Регистр управления приемником
0x3000_006C	0x0036	0x04	XCRL	Регистр управления передатчиком
0x3000_006C	0x0036	0x05	XCRH	Регистр управления передатчиком
0x3000_006C	0x0036	0x06	SRGRL	Регистр управления генераторами кадровой и битовой синхронизации
0x3000_006C	0x0036	0x07	SRGRH	Регистр управления генераторами кадровой и битовой синхронизации
0x3000_006C	0x0036	0x08	MCRL	Регистр управления многоканальными режимами приемника
0x3000_006C	0x0036	0x09	MCRH	Регистр управления многоканальными режимами передатчика
0x3000_006C	0x0036	0x0A	RCERL	Регистр маски приемника
0x3000_006C	0x0036	0x0B	RCERH	Регистр маски приемника
0x3000_006C	0x0036	0x0C	XCERL	Регистр маски передатчика
0x3000_006C	0x0036	0x0D	XCERH	Регистр маски передатчика
0x3000_006C	0x0036	0x0E	PCR	Регистр управления выводами
0x3000_006C	0x0036	0x0F	SPSR	Регистр состояния

21.3 Конфигурирование последовательного порта

Настройка последовательного порта осуществляется через регистры управления SPCR и PCR. В них содержатся биты управления, а также биты состояния McBSP.

Также регистр PCR предназначен для управления назначением выводов McBSP.

21.3.1 SPCRН

Таблица 21-4 – Регистр SPCRН

Номер	15	14...12	11, 10	9, 8
Доступ	R/W,+0	R/W,+0	R/W,+0	R/W,+0
Сброс				
	FREE	XINTM[2:0]	XEVNT1[1:0]	XEVNT0[1:0]

Номер	7	6...4	3, 2	1, 0
Доступ	R/W,+0	R/W,+0	R/W,+0	R/W,+0
Сброс				
	SOFT	RINTM[2:0]	REVNT1[1:0]	REVNT0[1:0]

Таблица 21-5 – Описание бит регистра SPCRН

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15	FREE	Режим FreeRun (DEBUG Mode): 0 – режим FreeRun отключен; 1 – режим FreeRun включен
14...12	XINTM[2:0]	Выбор источника прерывания передатчика: 000 – XRDY; 001 – начало блока данных (XSOB); 010 – окончание блока данных (XEOB); 011 – ошибка синхронизации передатчика XSYNCERR; 100 – FIFO передатчика полупуст; 101 – FIFO передатчика полуполон; 110 – FIFO передатчика пуст; 111 – FIFO передатчика полон
11, 10	XEVNT1[1:0]	Выбор источника события передатчика: 00 – XRDY; 01 – FIFO передатчика полуполон; 10 – FIFO передатчика почти полон; 11 – FIFO передатчика полон
9, 8	XEVNT0[1:0]	Выбор источника события передатчика: 00 – XRDY; 01 – FIFO передатчика полупуст; 10 – FIFO передатчика не почти полон; 11 – FIFO передатчика не полон
7	SOFT	Режим Halt (DEBUG Mode): 0 – режим Halt отключен; 1 – режим Halt включен

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
6...4	RINTM[2:0]	Выбор источника прерывания приемника: 000 – RRDY; 001 – начало блока данных (RSOB); 010 – окончание блока данных (REOB); 011 – ошибка синхронизации приемника RSYNCERR; 100 – FIFO приемника полупуст; 101 – FIFO приемника полуполон; 110 – FIFO приемника пуст; 111 – FIFO приемника полон
3, 2	REVNT1[1:0]	Выбор источника события передатчика: 00 – RRDY; 01 – FIFO приемника полуполон; 10 – FIFO приемника почти полон; 11 – FIFO приемника полон
1, 0	REVNT0[1:0]	Выбор источника события приемника: 00 – RRDY; 01 – FIFO приемника полупуст; 10 – FIFO приемника не почти полон; 11 – FIFO приемника не полон

21.3.2 SPCRL

Таблица 21-6 – Регистр SPCRL

Номер	15	14	13, 12	11, 10	9, 8
Доступ	R/W,+0	R/W,+0	R/W,+1	R/W,+0	R/W,+0
Сброс					
	DXENA	LW_ACC	JBOUND[1:0]	XJUST[1:0]	RJUST[1:0]

Номер	7	6	5, 4	3	2	1	0
Доступ	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0
Сброс							
	DLB	ABIS	CLKSTP[1:0]	nXRST	nRRST	nFGRST	nCGRST

Таблица 21-7 – Описание бит регистра SPCRL

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15	DXENA	Выключатель DX: 0 – выключатель DX выключен; 1 – выключатель DX включен
14	LW_ACC	Способ записи/чтения данных: 0 – слово считается считанным/записанным после любого чтения/записи; 1 – слово считается считанным/записанным когда считаны/записаны все байты

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
13, 12	JBOUND[1:0]	Граница выравнивания байтов/слов: 00 – 1 байт; 01 – 2 байта; 1х – 4 байта
11, 10	XJUST[1:0]	Выравнивание передаваемых данных: 0х – выравнивание по правой границе; 1х – выравнивание по левой границе (в соответствии с JBOUND)
9, 8	RJUST	00 – выравнивание по правой границе; 01 – выравнивание по правой границе с расширением знака (MSB); 10 – выравнивание по левой границе (в соответствии с JBOUND) с заполнением нулями старших и младших разрядов; 11 – выравнивание по левой границе (в соответствии с JBOUND) с расширением знака (MSB) и заполнением нулями младших разрядов
7	DLB	Режим КЗ: 0 – режим КЗ выключен (штатный режим работы); 1 – режим КЗ включен
6	ABIS	Режим A-bis: 0 – режим A-bis отключен; 1 – режим A-bis включен
5, 4	CLKSTP[1:0]	Режим останова сигнала битовой синхронизации: 0х – режим останова сигнала битовой синхронизации отключен (штатный режим работы для не-SPI режима); 10 – режим работы с остановом сигнала битовой синхронизации без задержки; 11 – режим работы с остановом сигнала битовой синхронизации с задержкой сигнала синхронизации на полпериода
3	nXRST	Сброс передатчика: 0 – передатчик в состоянии сброса; 1 – передатчик в штатном режиме работы
2	nRRST	Сброс приемника: 0 – приемник в состоянии сброса; 1 – приемник в штатном режиме работы
1	nFGRST	Сброс генератора кадровой синхронизации: 0 – генератор кадровой синхронизации в состоянии сброса; 1 – генератор кадровой синхронизации в штатном режиме работы
0	nCGRST	Сброс генератора битовой синхронизации: 0 – генератор битовой синхронизации в состоянии сброса; 1 – генератор битовой синхронизации в штатном режиме работы

21.3.3 SPSRH

Таблица 21-8 – Регистр SPSRH

Номер	15	14	13	12	11	10	9	8
Доступ		R,+0	R,+0	R,+1	R,+0	R,+0	R,+1	R,+1
Сброс								
	-	XFIFO_F	XFIFO_H	XFIFO_E	XLOST	XSERR	XEMPTY	XRDY

Номер	7	6	5	4	3	2	1	0
Доступ		R,+0	R,+0	R,+1	R,+0	R,+0	R,+0	R,+0
Сброс								
	-	RFIFO_F	RFIFO_H	RFIFO_E	RLOST	RSERR	REMPY	RRDY

Таблица 21-9 – Описание бит регистра SPSRH

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15	-	Зарезервировано
14	XFIFO_F	Признак заполненности FIFO передатчика (XBR): 0 – XBR не полон; 1 – XBR полон
13	XFIFO_H	Признак половинной заполненности FIFO передатчика (XBR): 0 – XBR полупуст (более чем наполовину пуст); 1 – XBR полуполон (заполнен более чем на половину)
12	XFIFO_E	Признак заполненности FIFO передатчика (XBR): 0 – XBR не пуст; 1 – XBR пуст
11	XLOST	Признак перезаписи данных в регистре DXR: 0 – отсутствие потери данных; 1 – произведена перезапись данных в DXR до их перемещения в FIFO
10	XSERR	Признак ошибки синхронизации передатчика: 0 – ошибки синхронизации передатчика отсутствуют; 1 – имелаась ошибка при синхронизации передатчика
9	XEMPTY	Опустошение передатчика: 0 – XSR не пуст; 1 – XSR пуст
8	XRDY	Готовность передатчика принять данные: 0 – новые данные передатчиком приняты быть не могут; 1 – передатчик готов принять данные новые
7	-	Зарезервировано
6	RFIFO_F	Признак заполненности FIFO приемника (RBR): 0 – RBR не полон; 1 – RBR полон
5	RFIFO_H	Признак половинной заполненности FIFO приемника (RBR): 0 – RBR полупуст (более чем наполовину пуст); 1 – RBR полуполон (заполнен более чем на половину)

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
4	RFIFO_E	Признак заполненности FIFO приемника (RBR): 0 – RBR не пуст; 1 – RBR пуст
3	RLOST	Признак перезаписи данных в регистре RSR: 0 – отсутствие потери данных; 1 – произведена перезапись данных в RSR до их перемещения в FIFO
2	RSERR	Признак ошибки синхронизации приемника: 0 – ошибки синхронизации приемника отсутствуют; 1 – произошла ошибка при синхронизации приемника
1	RFULL	Переполнение приемника: 0 – приемник не полон; 1 – приемник полон и не может принять новые данные из линии
0	RRDY	Наличие принятых данных: 0 – новых данных нет; 1 – приемник содержит новые данные

21.3.4 PCRL

Таблица 21-10 – Регистр PCRL

Номер	15	14	13	12	11	10	9	8
Доступ			R/W,+0	R/W,+0	R/W,+1	R/W,+1	R/W,+1	R/W,+1
Сброс								
	-	-	XIOEN	RIOEN	FSXM	FSRM	CLKXM	CLKRM

Номер	7	6	5	4	3	2	1	0
Доступ		R,+0	R,+0	R,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0
Сброс								
		CLKS_ST	DX_ST	DR_ST	FSXP	FSRP	CLKXP	CLKRP

Таблица 21-11 – Описание бит регистра PCRL

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15...14	-	Зарезервировано
13	XIOEN	Назначение выводов передатчика: 0 – выводы передатчика используются как GPIO; 1 – выводы передатчика подключены к контактным площадкам
12	RIOEN	Назначение выводов приемника: 0 – выводы приемника используются как GPIO; 1 – выводы приемника подключены к контактным площадкам
11	FSXM	Режим кадровой синхронизации передатчика: 0 – кадровая синхронизации от внешнего источника; 1 – кадровая синхронизации от внутреннего генератора

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
10	FSRM	Режим кадровой синхронизации приемника: 0 – кадровая синхронизации от внешнего источника; 1 – кадровая синхронизации от внутреннего генератора
9	CLKXM	Режим битовой синхронизации передатчика: 0 – битовая синхронизации от внешнего источника; 1 – битовая синхронизации от внутреннего генератора
8	CLKRM	Режим битовой синхронизации приемника: 0 – битовая синхронизации от внешнего источника; 1 – битовая синхронизации от внутреннего генератора
7	-	Зарезервировано
6	CLKS_ST	Отражает текущее состояние вывода CLKS. * Фиксируется в регистре на частоте работы параллельного интерфейса
5	DX_ST	Отражает текущее состояние вывода DX. * Фиксируется в регистре на частоте работы параллельного интерфейса
4	DR_ST	Отражает текущее состояние вывода DR. * Фиксируется в регистре на частоте работы параллельного интерфейса
3	FSXP	Полярность сигнала кадровой синхронизации передатчика: 0 – активный высокий уровень кадровой синхронизации; 1 – активный низкий уровень кадровой синхронизации
2	FSRP	Полярность сигнала кадровой синхронизации приемника: 0 – активный высокий уровень кадровой синхронизации; 1 – активный низкий уровень кадровой синхронизации
1	CLKXP	Активный фронт сигнала битовой синхронизации передатчика: 0 – активный передний фронт битовой синхронизации; 1 – активный задний фронт битовой синхронизации
0	CLKRP	Активный фронт сигнала битовой синхронизации приемника: 0 – активный передний фронт битовой синхронизации; 1 – активный задний фронт битовой синхронизации

21.4 Управление приемом и передачей

Управление приемом и передачей данных управляется посредством регистров управления приемником RCR и управления передатчиком XCR

21.4.1 RCRH, XCRH

Таблица 21-12 – Регистры RCRH, XCRH

Номер	15	14...8	7...5	4...0
Доступ		R/W,+0	R/W,+31	R/W,+15
Сброс				
	MPHASE	FLEN_P1[6:0]	CODEC[2:0]	WLEN_P1[4:0]

Таблица 21-13 – Описание бит регистров RCRH, XCRH

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15	MPHASE	Выбор одно- и многофазного (двухфазного) режимов работы: 0 – однофазный режим; 1 – многофазный режим
14...8	FLEN_P1[6:0]	Размер кадра фазы №2. Размер кадра в словах -1. (реальный диапазон 1–128 слов в кадре)
7...5	CODEC[2:0]	Способ кодирования/декодирования: 000 – кодирование/декодирование отключено; 001 – кодирование/декодирование отключено (зарезервировано); 010 – включен режим компадирования/декомпадирования по u-Law; 011 – включен режим компадирования/декомпадирования по A-Law; 100 – кодирование/декодирование отключено (зарезервировано); 101 – кодирование/декодирование отключено (зарезервировано); 110 – кодирование/декодирование отключено (зарезервировано); 111 – кодирование/декодирование отключено (зарезервировано)
4...0	WLEN_P1[4:0]	Размер слова данных фазы №2 Размер слова данных в битах -1 (реальный диапазон 4–32 бит в слове) работа блока при размере слова менее четырех бит не гарантируется

21.4.2 RCRL, XCRL

Таблица 21-14 – Регистры RCRL, XCRL

Номер	15	14...8	7	6, 5	4...0
Доступ	R/W,+0	R/W,+31	R/W,+0	R/W,+1	R/W,+15
Сброс					
	FIG	FLEN_P0[6:0]	MSB1st	DELAY[1:0]	WLEN_P0[4:0]

Таблица 21-15– Описание бит регистров RCRL, XCRL

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15	FIG	Режим игнорирования кадровой синхронизации: 0 – каждый импульс кадровой синхронизации перезапускает обмен; 1 – импульсы кадровой синхронизации кроме первого игнорируются
14...8	FLEN_P0[6:0]	Размер кадра фазы №1. Размер кадра в словах -1. (реальный диапазон 1–128 слов в кадре)
7	MSB1st	Порядок передачи бит в слове: 0 – начиная с младшего бита (LSB); 1 – начиная со старшего бита (MSB)

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
6...5	DELAY[1:0]	Задержка данных относительно кадровой синхронизации: 0–3 битовых интервала; * Нулевая задержка соответствует началу передачи битов одновременно с импульсом кадровой синхронизации
4...0	WLEN_P0[4:0]	Размер слова данных фазы №1. Размер слова данных в битах -1 (реальный диапазон 4-32 бит в слове). * Работа блока при размере слова менее четырех бит не гарантируется

21.5 Порядок приема и передачи данных

Как показано на структурной схеме, прием и передача осуществляется через FIFO.

Принимаемые данные поступают на вывод RX и вдвигаются в RSR. После получения заданного количества битов данные перемещаются в FIFO приемника RBR, если он не полон, и далее копируются в DRR, и остаются там до тех пор, пока не будут считаны через параллельный интерфейс

Передаваемые данные записываются через параллельный интерфейс в регистр DXR, откуда перемещаются в FIFO передатчика, если он не заполнен. После этого данные попадают в сдвиговый регистр передатчика XSR, откуда выдвигаются через вывод TX.

21.6 Сброс последовательного порта

Последовательный порт может быть сброшен следующими способами:

- аппаратный сброс (вывод HRESETn);
- программный внешний сброс (вывод SRESETn);
- программный внутренний сброс (биты управления регистра SPCR).

21.7 Программный внутренний сброс (биты управления регистра SPCR)

В McBSP предусмотрен независимый сброс отдельных его блоков посредством регистра управления SPCR.

Программно могут быть сброшены:

- внутренний генератор битовой синхронизации (SPCR.nFGRST);
- внутренний генератор кадровой синхронизации (SPCR.nCGRST);
- приемник (SPCR.nRRST);
- передатчик (SPCR.nXRST).

Программный внутренний сброс переводит соответствующий блок McBSP в исходное состояние синхронно после подачи его активного уровня (выставления соответствующего бита в регистре управления SPCR). После снятия сигнала аппаратного сброса выход блока из состояния сброса осуществляется также синхронно.

Примечание – Перед сменой параметров приемника/передатчика рекомендуется перевести их в состояние программного внутреннего сброса, сменить параметры, вывести из состояния программного внутреннего сброса

21.8 Обработка статусов

Признаки RRDY и XRDY определяют состояния готовности приемника и передатчика соответственно. Запись и чтение данных последовательного порта могут быть синхронизированы опросом этих битов, или используя их в качестве источников запросов DMA или запросов прерывания ЦП.

21.9 Состояние приемника: RRDY, RRINT, RFULL, RSYNCERR

Признак RRDY = 1 сигнализирует о наличии несчитанных данных в регистре DRRL и DDRH, которые могут быть считаны ЦП или DMA-контроллером. Сразу после чтения, а также после сброса McBSP или приемника McBSP, данный признак сбрасывается в 0. Кроме данного признака есть программируемый вывод состояния приемника, на который может быть выведен данный признак (SPCRH.RINTM=000b).

21.10 Состояние передатчика

Признак XRDY = 1 сигнализирует о готовности передатчика принять новое слово данных в регистры DXRL и DXRH. Данный признак снимается сразу после записи в него данных через параллельный интерфейс и выставляется после копирования полной порции данных из регистра DXRL и DXRH в FIFO передатчика XBR. Также имеется программируемый вывод состояния передатчика, на который может быть выведен данный признак (SPCRH.XINTM=000b).

21.11 События и прерывания

В McBSP имеются шесть программируемых выводов для информирования управляющее ЦП (XINT и RINT) или DMA-контроллер (XEVT[x2] и REVT[x2]) о возникновении события.

Прерывания приемника (RINT) и передатчика (XINT) информируют ЦП об изменении состояния последовательного порта. Данные выводы программируемы и для каждого доступны восемь настроек, определяемых полем конфигурации (RINTM и XINTM соответственно) источников событий:

1 RINTM/XINTM = 000b. Прерывание по каждому слову перемещаемому между интерфейсным регистром (DRR/DXR) и FIFO буфером (RBR/XBR) приемника и передатчика определяемым битами SPCR.RRDY/SPCR.XRDY соответственно.

2 RINTM/XINTM = 001b. Прерывание по признаку конца текущей фазы приема/передачи.

3 RINTM/XINTM = 010b. Прерывание по признаку кадровой синхронизации.

4 RINTM/XINTM = 011b. Прерывание по признаку обнаружения ошибки кадровой синхронизации приемника (RSYNCERR) и передатчика (XSYNCERR) соответственно.

5 RINTM/XINTM = 100b. Заполнено менее половины FIFO приемника/передатчика.

- 6 RINTM/XINTM = 101b. Заполнено половина или более FIFO приемника/передатчика.
- 7 RINTM/XINTM = 110b. FIFO приемника/передатчика опустошено (не содержит данных).
- 8 RINTM/XINTM = 111b. FIFO приемника/передатчика полностью заполнено.

Для управления выводом событий через выходы XEVNT0/1 и REVENT0/1 применяются поля XEVNT0/1 и REVNT0/1 соответственно. В отличие от выводов прерываний данные выходы не имеют промежуточного регистра на выходе блока. Назначение сигналов аналогично выводам RINT/XINT.

21.12 Настройка битовой и кадровой синхронизации

Рисунок 21-1 демонстрирует типовой режим работы битовой и кадровой синхронизации. Сигналы RCLK и TCLK определяют границы битов на прием и на передачу соответственно. Аналогично сигналы TFR/RFR определяют границу слова.

В McBSP предусмотрена независимая настройка параметров синхронизации данных для приемника и передатчика:

- Полярность синхросигналов RFR, TFR, RCLK и TCLK;
- Выбор между одно- и многофазным режимами;
- Количество слов в кадре для каждой фазы;
- Количество битов в слове для каждой фазы;
- Перезапуск обмена по каждому импульсу кадровой синхронизации или их игнорирование;
- Задержка данных относительно сигнала кадровой синхронизации от 0 до 3х битовых интервалов;
- Выравнивание данных по правой или левой границе, и расширение знака;
- Выравнивание передаваемых/принимаемых по границе байта (8 бит), слова (16 бит) или двойного слова (32 бит).

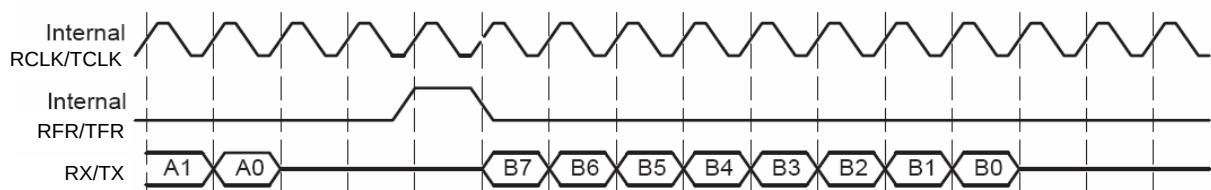


Рисунок 21-1 – Работа кадровой и битовой синхронизации

Битовая и кадровая синхронизации приемника и передатчика могут задаваться от внутреннего или от внешнего источника. Управление режимами осуществляется через биты режима в регистре PCR. Также на источник кадровой и битовой синхронизации влияет бит КЗ (DLB) при котором битовая и кадровая синхронизации приемника задаются от битовой и кадровой синхронизации передатчика.

Когда кадровая синхронизация приемника (RFR) и передатчика (TFR) задаются от внешнего источника (PCR.FSRM=PCR.FSRM=0), то McBSP определяет их наличие по отрицательному фронту сигналов битовой синхронизации приемника (RCLK) и передатчика (TCLK) соответственно.

Когда же кадровая синхронизация приемника (RFR) и передатчика (TFR) задаются от внутреннего источника ($PCR.FSRM=PCR.FSRM=1$), то McBSP выставляет их на линию по положительному фронту сигналов битовой синхронизации приемника (RCLK) и передатчика (TCLK) соответственно.

Входные данные захватываются по отрицательному фронту сигнала RCLK, а передаваемые данные выставляются по положительному фронту сигнала TCLK.

Биты FSXP, FSRP, CLKXP и CLKRP определяют полярность сигналов TFR, RFR, TCLK и RCLK. Все внутренние сигналы кадровой синхронизации имеют активный уровень «1», данные выставляются по положительному фронту битовой синхронизации, а принимаются по отрицательному. Для согласования уровней и активных фронтов с внешней линией производится инверсия сигналов кадровой и битовой синхронизации в соответствии с выставленными битами управления полярностью сигналов. Инверсия производится посредством вентилях типа XOR.

21.12.1 Фазы кадровой синхронизации

Кадровая синхронизация обозначает начало обмена в McBSP. Поток данных, следующий за кадровой синхронизацией, может иметь одну или две фазы (фаза 1 и фаза 2). Количество фаз определяется битом управления в регистрах XCR и RCR (бит MPHASE). Размер кадра и размер слова для каждой фазы задается независимо посредством полей FLEN_P0/ FLEN_P1 и WLEN_P0/WLEN_P1 соответственно.

Размер кадра определяется числом слов в фазе кадра (от 1 до 128 слов), а размер слова определяется длиной слова в пределах фазы. Размер слова может быть задан произвольно в диапазоне от 4-х до 32-х бит*.

Примечание – размер слова может быть задан и менее 4-х, но при этом работа блока не гарантируется.

21.12.2 Упаковка данных заданием размеров кадра и слова.

Для эффективного управления упаковкой данных можно воспользоваться изменением размеров кадра и слова.

Пример такой упаковки приведен ниже.

К примеру, необходимо передавать данные по четыре байта (восемь бит) в кадре с одной фазой. Данный режим может быть запрограммирован при помощи следующих установок:

- MPHASE = 0b , однофазный кадр;
- FLEN_P0 = 0000011b (0x03), кадр из четырех слов;
- WLEN_P0 = 0000111b (0x07), слово из восьми бит.

В данном случае между ЦП или контроллером DMA и McBSP будет совершено восемь обменов восьмибитными словами – четыре чтения из DRR и четыре записи в DXR.

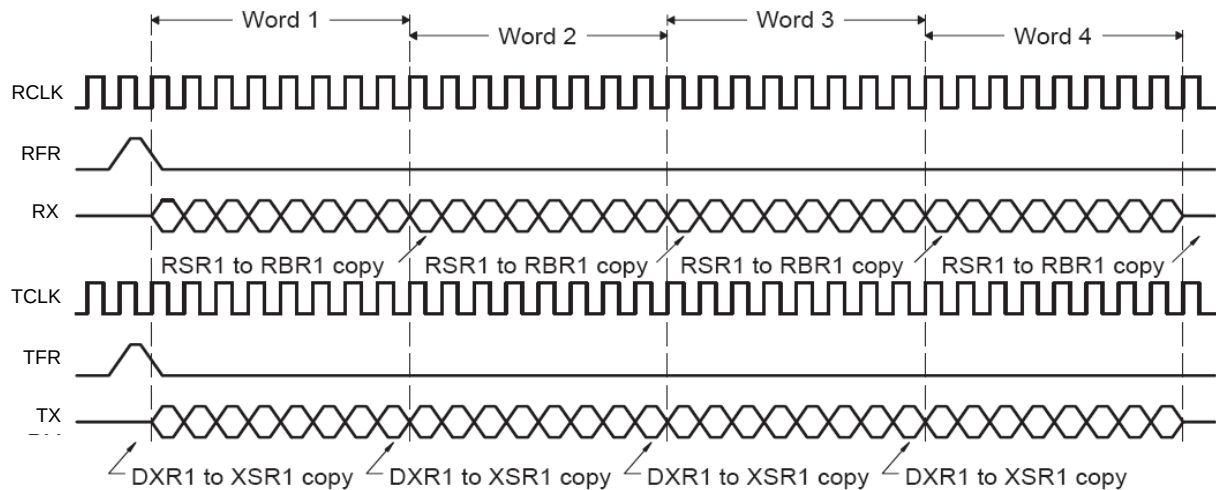


Рисунок 21-2 – Пример упаковки данных методом задания размера (несколько слов)

В тоже время можно тот же массив данных принять другим способом – как одно 32-х разрядное слово.

- MPHASE = 0b , однофазный кадр
- FLEN_P0 = 0000000b (0x00), кадр из одного слова
- WLEN_P0 = 11111b (0x1F), слово из 32х бит

В данном случае будет произведено всего два обмена с ЦП или контроллером DMA – по одному 32-разрядному слову на прием и на передачу.

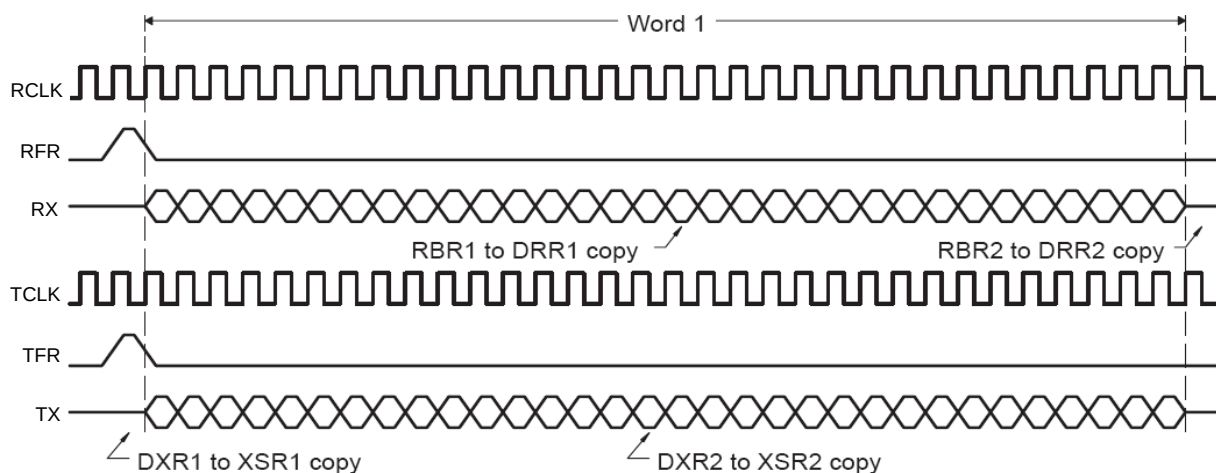


Рисунок 21-3 – Пример упаковки данных методом задания размера (одно слово)

Как видно из примера управление разрядностью передаваемых слов и их количеством позволяет проводить упаковку и распаковку данных, тем самым снижая нагрузку на ЦП или контроллер DMA.

21.13 Задержка данных

Начало импульса кадровой синхронизации определяет первый цикл, когда может начаться передача пакета. Реальный же пакет при необходимости может быть отправлен/принят с некоторой задержкой. Данная задержка определяется полями DELAY в регистрах управления XCR и RCR соответственно. Диапазон программно задаваемых

задержек от нуля до трех битовых интервалов включительно. По сбросу установлена задержка в один битовый интервал.

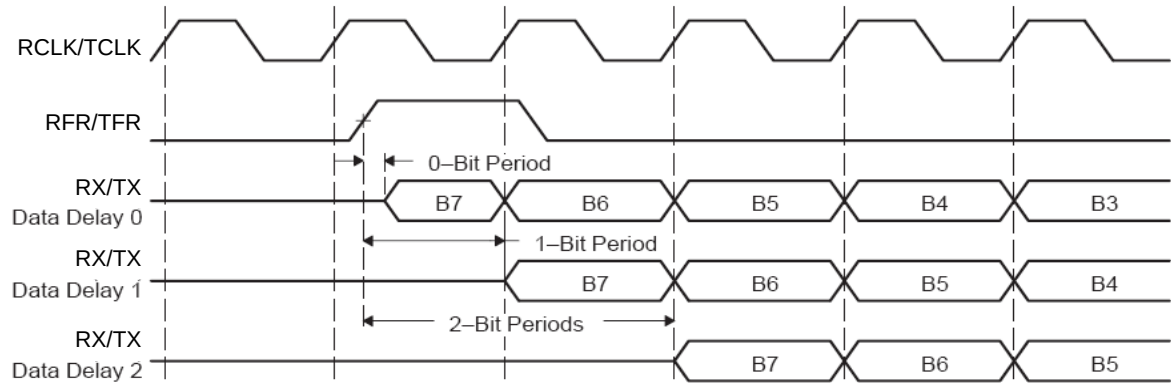


Рисунок 21-4 – Задержка данных

Обычно кадровый импульс обнаруживается и захватывается по фронту сигнала битовой синхронизации. Таким образом, данные могут быть приняты или переданы одним битовым интервалом позже. Однако, в случае если установлена нулевая задержка приема/передачи, данные должны начать передаваться и приниматься в том же битовом интервале что и импульс кадровой синхронизации. При приеме данная проблема решается работой приемника по отрицательному фронту сигнала битовой синхронизации. Однако, передача должна начаться с началом сигнала кадровой синхронизации. Таким образом, первый бит передаваемых данных должен уже находиться в регистре XSR и транслироваться на вывод DX.

Другой часто используемый режим - это задержка данных на два битовых интервала. Данный режим позволяет подключать последовательный порт к различным формирователям кадров типа T1 с предварением пакета стартовым битом.

Во время приема подобного пакета (если установлена задержка в два битовых интервала) последовательный порт, по сути, отбрасывает стартовый бит. При передаче в этом случае передатчик оставляет стартовый бит в высокоимпедансном состоянии. Подразумевается, что формирователь кадров вставляет собственный стартовый бит или что данный бит формируется сторонним устройством. К тому же, можно установить подтяжку к «0» или к «1» вывода DX для поведением в этом состоянии.

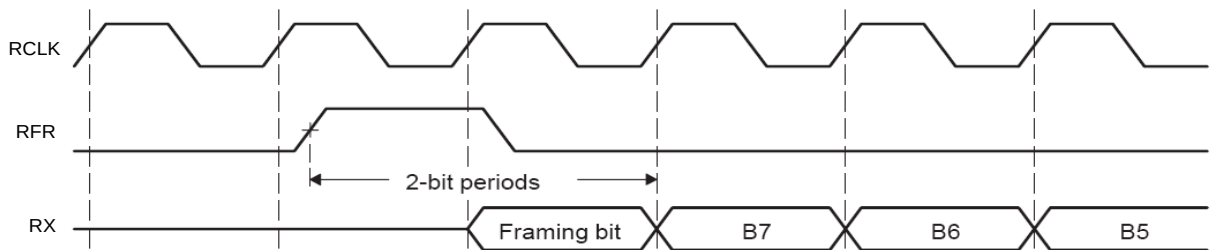


Рисунок 21-5 – Задержка данных на два битовых интервала

21.14 Пример многофазного кадра (AC97)

Хорошим примером использования многофазного кадра является применение аудиокодека стандарта AC97, использующего двухфазные кадры, где первая фаза содержит одно слово из 16 бит, за которым следуют 12 слов длиной 20 бит.

Для работы с таким кадром можно использовать следующую настройку последовательного порта:

- FSRP1 = 0b, активный высокий уровень;
- MPHASE = 1b, многофазный кадр;
- FLEN_P0 = 0000000b (0x00), кадр из 1го слова;
- WLEN_P0 = 01111b (0x0F), слово из 16х бит;
- FLEN_P1 = 0001011b (0x0B), кадр из 12 слов;
- WLEN_P1 = 10011b (0x13), слова из 20 бит;
- DELAY = 01b (0x1), задержка 1 битовый интервал.

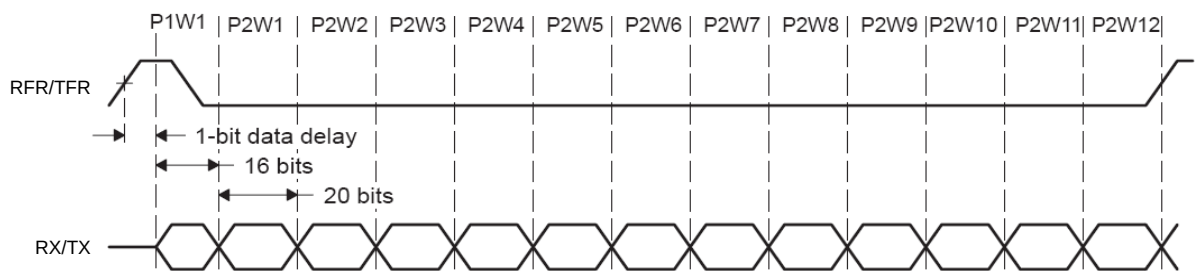


Рисунок 21-6 – Многофазный кадр

Примечательно, что в кадрах данного формата импульс кадровой синхронизации перекрывает первое слово кадра. В McBSP кадрковая синхронизация осуществляется по переходу кадрowego синхроимпульса из неактивного состояния в активное и, таким образом, сигнал кадровой синхронизации длительностью менее длины кадра эквивалентен импульсу в один битовый интервал.

21.15 Штатный режим работы McBSP

Во время последовательного обмена обычно существуют периоды неактивности между последовательными пакетами. Для каждого кадра формируется импульс кадровой синхронизации. Если McBSP не находится в состоянии сброса или останова и настроен на желаемый режим работы обычный последовательный режим работы может быть запущен установкой битов MPHASE = 0, для однофазного кадра, и заданным числом слов в кадре FLEN_P0. Кроме того, необходимо, чтобы была установлена длина слова WLEN_P0. Для работы с двухфазными кадрами необходимо установить MPHASE = 1 и установить значения длины кадра и слова для фазы 2 в регистрах FLEN_P1 и WLEN_P1 соответственно.

Примечания

* Длины кадра и слова указываются с вычетом 1, т.е. значение «0x00» в соответствующем поле определяет 1 битовое слово или однословный кадр

** При длине слова менее четырех бит работа устройства не гарантируется.

На рисунке ниже (Рисунок 21-7) приведен пример однофазного кадра, содержащего одно восьмибитное слово. Данная диаграмма (и последующие тоже, если не указано иначе) подразумевает следующие настройки приема/передачи:

- FSRP1 = 0b, активный высокий уровень;
- CLKRP1 = 0b, передача по положительному фронту, прием по отрицательному;
- MPHASE = 0b, однофазный кадр;
- FLEN_P0 = 0000000b (0x00), кадр из 1-го слова;
- WLEN_P0 = 00111b (0x0F), слово из 8 бит;
- DELAY = 01b (0x1), задержка 1 битовый интервал.

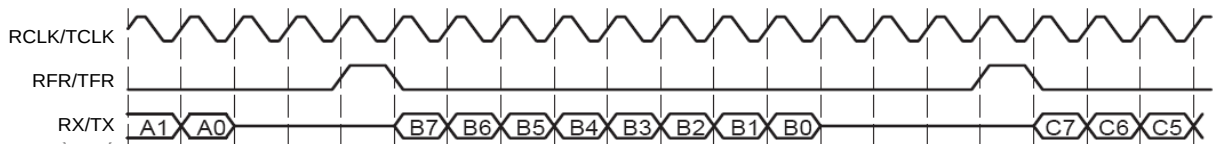


Рисунок 21-7 – Работа приемника

Ниже показан пример приема данных по последовательному порту (Рисунок 21-8). Сразу по переходу сигнала кадровой синхронизации из неактивного состояния в активное по отрицательному фронту сигнала SCLR приемник обнаруживает начало кадра. Далее, данные задвигаются в сдвиговый регистр (RSR) после выжиданий установленной задержки (RCR.DELAY). После приема слова данные из регистра RSR переносятся в FIFO приемника (RBR), откуда перемещаются в регистр приемника DRR с выставлением признака готовности данных (RRDY), откуда в свою очередь могут быть считаны ЦП или контроллером DMA. При чтении регистра DRR ЦП или DMA-контроллером признак готовности данных снимается.

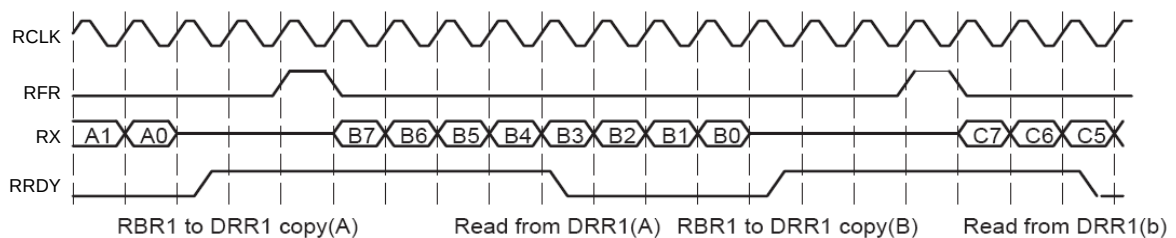


Рисунок 21-8 – Выставление признака готовности приемника

После получения импульса кадровой синхронизации через задержку (XCR.DELAY) передатчик начинает выдвигать содержимое регистра XSR через вывод DX. В регистре XSR для этого должны уже находиться данные на отправку которые попадают в него посредством FIFO передатчик, куда в свою очередь помещаются из регистра передатчика DXR, доступного для записи из ЦП или DMA-контроллера. При записи в регистр DXR ЦП или DMA-контроллером регистр помечается как занятый с одновременным снятием признака готовности передатчика к приему новых данных (XRDY). Признак готовности выставляется после того как данные из регистра DXR перемещены в FIFO передатчика.

Ниже на диаграмме приведен пример работы передатчика.

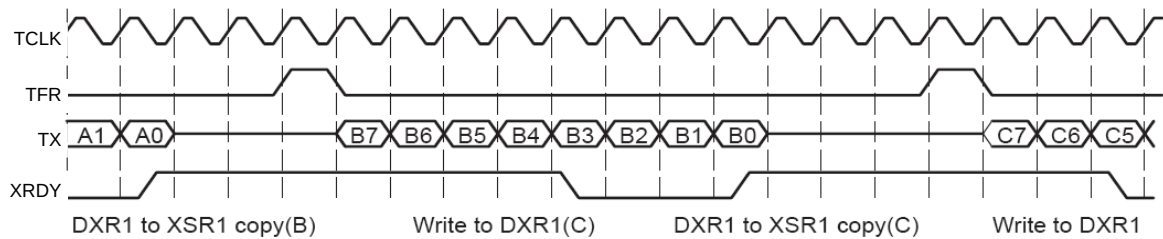


Рисунок 21-9 – Работа передатчика

Частота следования кадров определяется периодом между сигналами кадровой синхронизации:

$$F_f = F_b / N,$$

$$N = D + N_b + N_z,$$

где F_f – частота следования кадров;

F_b – частота следования битов;

N – число битовых интервалов между импульсами кадровой синхронизации.

Частота следования кадров при той же частоте следования битов может быть увеличена за счет уменьшения количества битов и в пределе ограничивается только числом значащих битов кадра. Таким образом, максимальная частота следования кадров определяется как:

$$F_f = F_b / N_b,$$

где F_f – частота следования кадров;

F_b – частота следования битов;

N_b — число битов в кадре.

Рисунок 21-10 демонстрирует пример работы McBSP на предельной частоте следования кадров.

При максимальной частоте следования кадров биты данных следуют непрерывным потоком. А при задержке = 1 импульс кадровой синхронизации перекрывает последний бит предыдущего кадра.

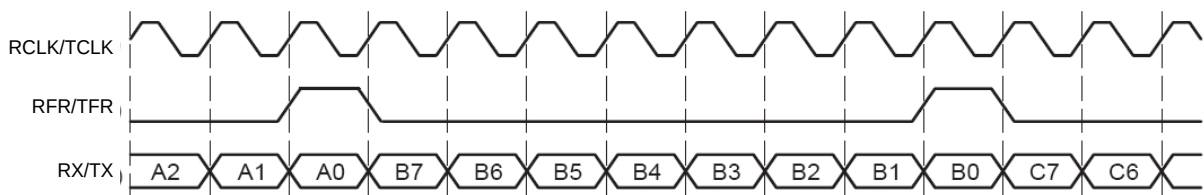


Рисунок 21-10 – Работа передатчика на предельной частоте следования кадров

В действительности допускаются непрерывные потоки данных и поэтому нет необходимости стробировать импульсом кадровой синхронизации все кадры. Теоретически достаточно только первого кадрового импульса для запуска многокадровой передачи. В McBSP предусмотрен подобный режим работы игнорированием импульса кадровой синхронизации кроме первого. Данный режим включается программно установкой бита FIG = 1.

21.16 Режим игнорирования кадровой синхронизации

McBSP может быть настроен для пропуска кадровой синхронизации приема и передачи. Бит $[X/R]CR.FIG = 0$ задает режим слежения за импульсами кадровой синхронизации, в то время как $[X/R]CR.FIG = 1$ задает режим их игнорирования, кроме первого. Пользователь может использовать второй режим для упаковки данных, либо для пропуска нежелательных импульсов кадровой синхронизации.

21.17 Упаковка данных при помощи режима пропуска кадровой синхронизации

Ранее был описан один из методов упаковки данных при помощи изменения размера принимаемого и количества слов в кадре. Этот пример показывает, как можно производить упаковку, если в кадре содержится несколько слов. Если же в кадре содержится одно слово, то данный метод не применим. Вместо него можно воспользоваться режимом пропуска импульсов синхронизации. Положим, что каждый кадр состоит из одного слова размером 8 бит. Таким образом, каждый кадр требует одного чтения и одной записи. Для снижения нагрузки на ЦП или DMA-контроллер можно осуществлять упаковку к примеру до 32-битных слов. Для этого наряду с изменением длины слова до 32-х бит необходимо установить бит $[X/R]CR.FIG = 1$ и, таким образом, поток данных будет восприниматься как поток 32-разрядных слов, что существенно снизит частоту обменов между McBSP и ЦП или DMA-контроллером. Диаграмма работы модуля в данном режиме представлена ниже (Рисунок 21-11, Рисунок 21-12).

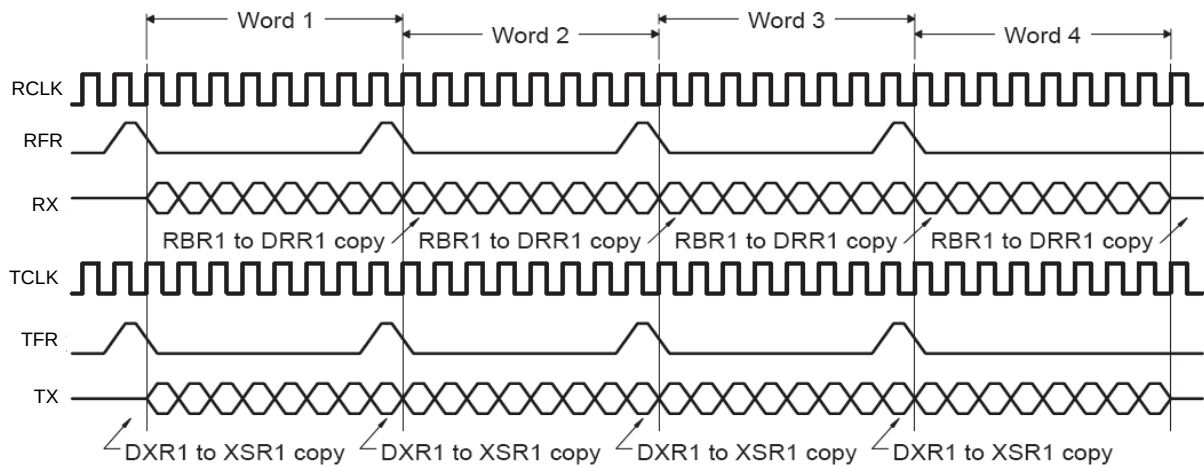


Рисунок 21-11 – Пример упаковки данных при помощи режима кадровой синхронизации (несколько слов)

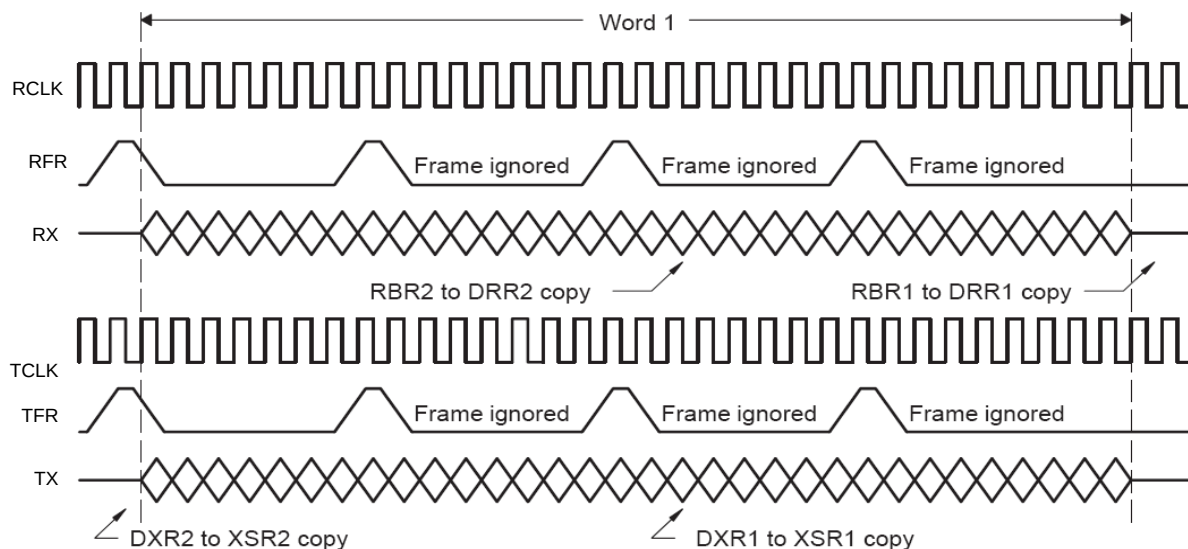


Рисунок 21-12 – Пример упаковки данных при помощи режима кадровой синхронизации (одно слово)

21.18 Пропуск нежелательных кадров при помощи режима пропуска кадровой синхронизации

В предыдущем разделе было показано, как при помощи режима пропуска кадровых импульсов можно производить упаковку данных. Также данный режим может использоваться для пропуска нежелательных или несвоевременных кадровых синхроимпульсов.

Если кадровые синхроимпульсы не игнорируются, то каждый новый синхроимпульс заново запускает передачу/прием, независимо от того, пришел ли он в ожидаемое время или нет. Если кадровый синхроимпульс пришел ранее чем завершилась передача/прием слова данных, то это слово отбрасывается.

В противном случае, если кадровые импульсы не игнорируются, прием/передача продолжают без разрывов между словами данных. Если же кадровый синхроимпульс кроме первого, запускающего прием/передачу, не оказывают никакого влияния на порядок приема/передачи данных. Однако, если при этом кадровый синхроимпульс придет в момент когда он не ожидается, то будет выставлен флаг ошибки синхронизации XSYNCERR и RSYNCERR.

Ниже приведены диаграммы обоих режимов – с пропуском кадровых синхроимпульсов и без него.

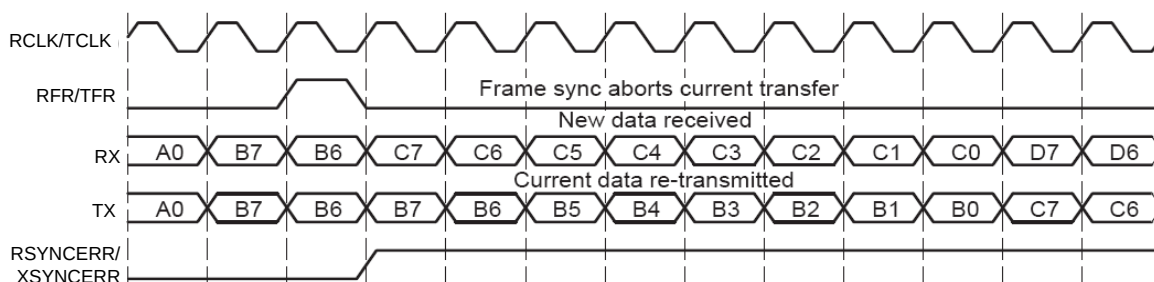


Рисунок 21-13 – Работа модуля в обычном режиме

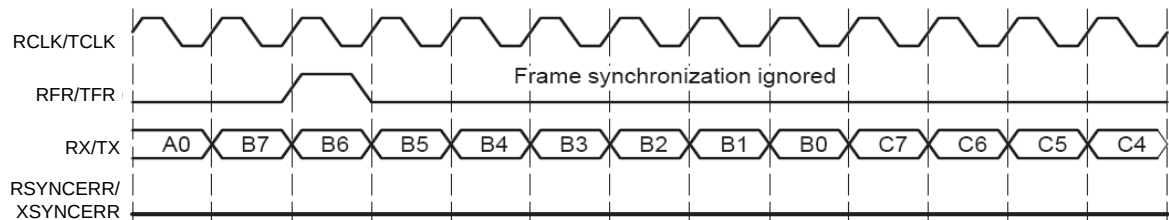


Рисунок 21-14 – Работа модуля в режиме пропуска кадровых синхроимпульсов

21.19 Особые ситуации работы последовательного порта

В контроллере McBSP существует пять событий, которые могут привести к системной ошибке:

1 **Переполнение приемника (RFULL = 1).** Данное событие отражает состояние ошибки когда начинается прием нового слова при том что сдвиговый регистр приемника (RSR) не был вычитан (заполнены все регистры и буферы приемника). Если при этом на вход приемника (DR) поступит новое слово, то это приведет к ошибке и при этом с поступлением новых банных слово, находившееся в RSR, будет затираться, приводя к потере данных.

2 **Несвоевременная кадровая синхронизация приемника (RSYNCERR = 1).** Это событие происходит при режиме слежения за кадровыми синхроимпульсами (FIG = 0) и поступлении импульса кадровой синхронизации ранее чем закончится кадр согласно запрограммированному размеру кадра и слова. Наступление данного события приводит к прерыванию приема текущего слова и началу приема/передачи нового.

3 **Перезапись передаваемых данных.** Этот случай происходит когда ЦП или контроллер DMA перезаписывает данные в регистре передатчика (DXR) до того как они будут перемещены в FIFO передатчика (XBR). Данная ситуация не обнаруживается встроенными средствами McBSP,

4 **Опустошение передатчика (XEMPTY = 1).** Данное событие возникает, когда сдвиговый регистр передатчика (XSR) пуст и начинается передача первого бита слова.

5 **Несвоевременная кадровая синхронизация передатчика (XSYNCERR = 1).** Это событие происходит при режиме слежения за кадровыми синхроимпульсами (FIG = 0) и поступлении импульса кадровой синхронизации ранее, чем закончится кадр согласно запрограммированному размеру кадра и слова. Наступление данного события приводит к прерыванию передачи текущего слова и началу передачи нового.

21.20 Переполнение приемника (RFULL)

Бит RFULL = 1 в регистре SPCR сигнализирует о том, что произошло (!) переполнение приемного регистра при приеме данных и данные были потеряны. Бит RFULL выставляется при условии, что RSR заполнен и нет текущего цикла переноса данных в FIFO и начался прием нового слова. RSR может быть заполнен по причине заполненности FIFO, которое в свою очередь может быть заполненным по причине отсутствия чтения DRR с момента последнего перемещения в него данных из FIFO, либо

отсутствием частоты на системной шине, по которой производится чтение регистра DRR и соответственно перенос в него данных из FIFO.

Данный флаг снимается автоматически при опустошении регистра RSR или при переносе данных из него в FIFO приемника RBR. Данное событие может произойти, если вычитан регистр DRR и соответственно в FIFO освободилось место для приема нового слова. Также данный флаг может быть снят сбросом приемника или всего McBSP.

Примечание – возникновение данного события показывает, что сложились все условия для возникновения ошибки и велика вероятность, что данные будут потеряны.

Несвоевременная кадровая синхронизация приемника (RSYNCERR)

На рисунке ниже (Рисунок 21-15) показано дерево решения по обработке всех кадровых синхроимпульсов приемника.

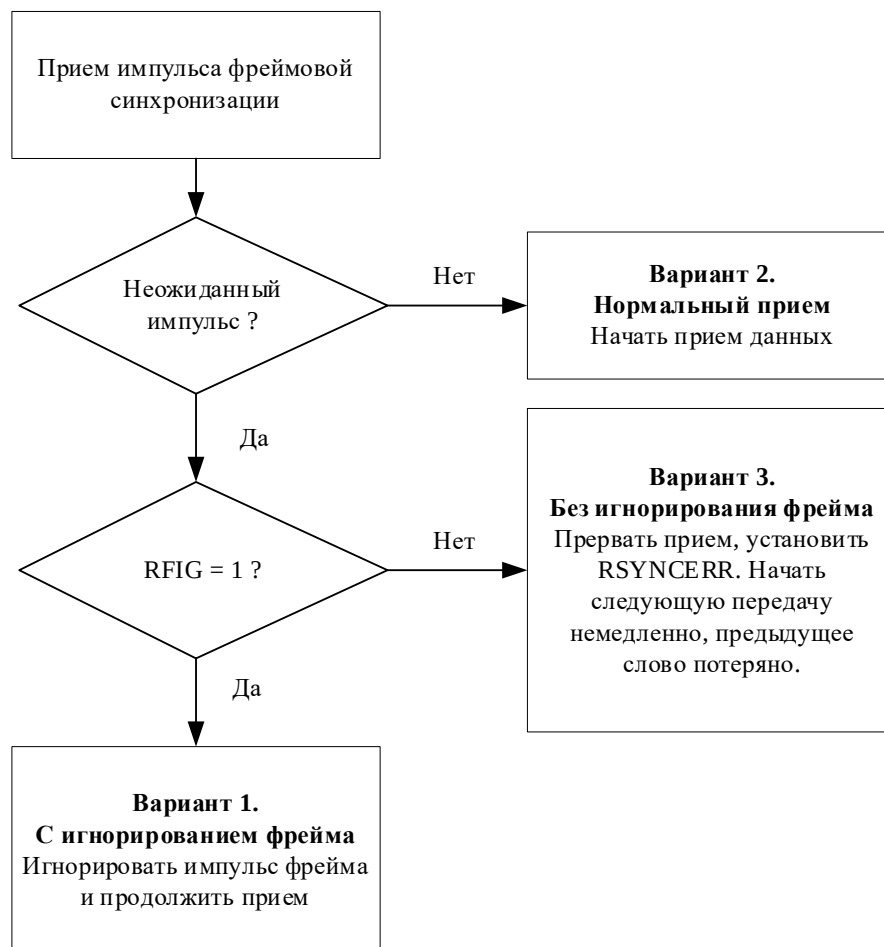


Рисунок 21-15 – Алгоритм обработки всех кадровых синхроимпульсов приемника

Диаграмма (Рисунок 21-16) подразумевает, что приемник находится в активном состоянии (RRST=0). Непредвиденные кадровые импульсы могут формироваться от внешнего или внутреннего источника кадровых синхроимпульсов. Непредвиденный кадровый синхроимпульс определен как синхроимпульс, пришедший на RCR.DELAY битовых интервалов ранее последнего передаваемого бита.

При этих условиях могут произойти следующие четыре случая:

1 Вариант 1: поступление непредвиденного внутреннего импульса FR при установленном FIG = 1. В данном случае кадровый синхроимпульс игнорируется и прием продолжается.

2 Вариант 2: Штатная работа приемника. Возможны следующие три случая:

а) SFSR первый синхроимпульс после сброса приемника (RSR = 1);

б) SFSR первый синхроимпульс после снятия флага ошибки переполнения приемника (RFULL = 1);

в) Приемник находится в состоянии обработки межпакетного интервала (данные в канале отсутствуют).

3 Вариант 3: Поступление непредвиденного кадрового синхроимпульса при установленном RCR.FIG = 0 (режим пропуска синхроимпульсов). Данный режим рассмотрен ранее. При наступлении данного случая происходит формирование сигнала ошибки кадровой синхронизации приемника (RSYNCERR), который может быть снят только записи «1» в соответствующий бит слова состояния или сбросом приемника.

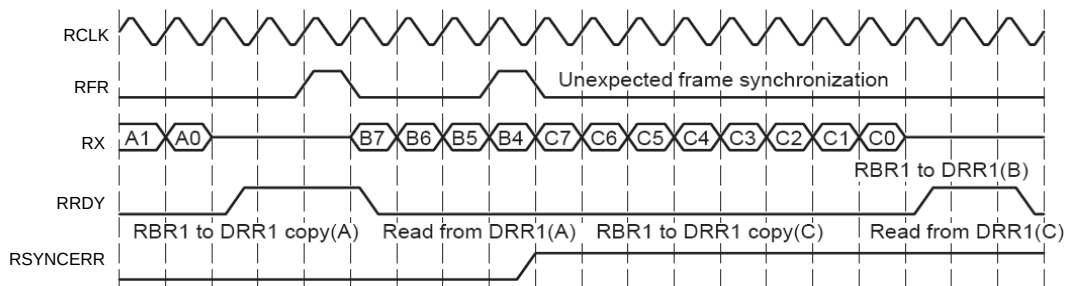


Рисунок 21-16 – Диаграмма обработки непредвиденных кадровых импульсов

21.21 Перезапись передаваемых данных.

Рисунок 21-17 показывает, что происходит, если данные в DXR были перезаписаны до того, как были отправлены. В примере изначально программист записывает данные С для передачи. После чего записывает данные D, перезаписывая данные С до того, как они были скопированы в FIFO передатчика XBR. Таким образом, данные С не будут переданы. ЦП или может избежать этого опрашивая бит готовности передатчика к приему новых данных для отправки (XRDY). Контроллер DMA может избежать этого, работая синхронизируя передачи по XRDY в одиночном режиме или по состоянию XBR в блочном режиме (состояния полуполного или пустого FIFO).

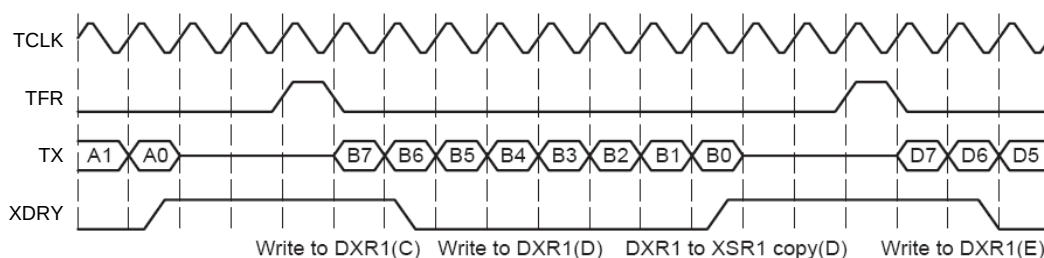


Рисунок 21-17 – Диаграмма работы модуля, в случае перезаписи неотправленных данных

Опустошение передатчика (XEMPTY)

Данное событие возникает когда сдвиговый регистр

Признак XEMPTY отражает состояние ошибки, когда начата передача слова.

Несвоевременная кадровая синхронизация передатчика (XSYNCERR)

Это событие аналогично событию отражаемому признаком RSYNCERR.

21.22 Выравнивание данных и расширение знака

В McBSP предусмотрено автоматическое выравнивание данных по границе байта, слова, двойного слова, а также расширение знака (при приеме) для предварительной обработки и тем самым снижения нагрузки на ЦП.

Данные возможности присутствуют как в приемнике для выравнивания, нормализации и получения знаковых чисел при приеме данных, так и в передатчике для отправки данных нормализованных по определенной границе.

Режим расширения знака доступен только в приемнике (в передатчике в нем нет необходимости).

При выравнивания по левой границе выравнивание осуществляется до ближайшей границы кратной указываемому в SPCR.JBOUND.

JBOUND описание:

- 00 выравнивание по границе байта (8 бит)
- 01 выравнивание по границе слова (16 бит)
- 1x выравнивание по границе двойного слова (32 бит)

Так при длине приеме слова 5 бит и значении SPCR.JBOUND = 00b слово будет выровнено по границе 8 бит (старший бит слова будет на 8-й позиции слова), а младшие $8 - 5 = 3$ бит будут дополнены нулями. При значении SPCR.JBOUND = 01b слово будет выровнено по границе 16 бит (старший бит слова будет на 16й позиции слова), а младшие $16 - 5 = 11$ бит будут дополнены нулями. При значении SPCR.JBOUND = 1xb слово будет выровнено по границе 32 бит (старший бит слова будет на 32-й позиции слова), а младшие $32 - 5 = 27$ бит будут дополнены нулями. Если же принимаемое слово будет, например, 17 разрядов, то границы слова на выходе приемника будут соответственно 24, 32 и 32.

Аналогично в передатчике и при передаче. Данный режим позволяет производить обрезание незначащих разрядов нормализованных величин тем самым сокращая число передаваемых битов и соответственно увеличивая пропускную способность последовательного порта, если имеется необходимость передавать строго начиная наименее значащего бита.

Расширение знака производится только в приемнике, поскольку при передаче в этом нет необходимости. При правом выравнивании расширение знака производится начиная с разряда, равного длине слова, а при левом выравнивании начиная с ближайшей границы выравнивания, установленной полем SPCR.JBOUND большей длины слова.

Граница выравнивания устанавливается одинаковая для приемника и передатчика в то время как режимы выравнивания независимы. По умолчанию граница выравнивания установлена на границу слова.

21.23 Кодирование/декодирование данных

В McBSP предусмотрены встроенное кодирование и декодирование данных, при передаче и приеме соответственно, для снижения нагрузки на ЦП. Режим кодирования/декодирования определяется полем CODEC. Всего доступны 8 режимов кодирования/декодирования. Штатно доступны четыре режима:

- кодирования отключено (CODEC = 000b);
- кодирование/декодирование кодом Манчестер-2 (CODEC = 001b);

- компадирование/декомпадирование по u-закону (CODEC = 010b);
- компадирование/декомпадирование по A-закону (CODEC = 011b).

Остальные режимы кодирования/декодирования зависят от конкретной реализации. Если в какой-либо из режимов не реализован, то поведение McBSP в этом случае аналогично отсутствию кодирования/декодирования (CODEC = 000b).

Путь кодирования/декодирования данных представлен на рисунке ниже.

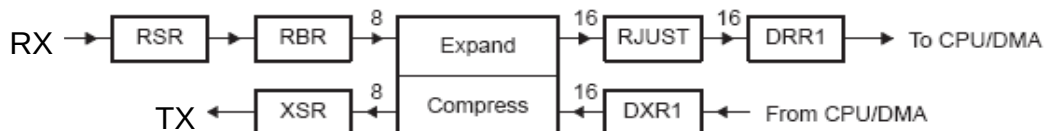


Рисунок 21-18 – Структурная схема модуля кодирования/декодирования данных

21.24 Компадирование/декомпадирование по u- и A-законам

При компадировании по u- и A-законам необходимо выравнивание данных согласно приведенному ниже (Рисунок 21-19 и Рисунок 21-20).

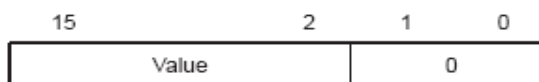


Рисунок 21-19 – Выравнивание данных для компадирования по u-закону в DXR

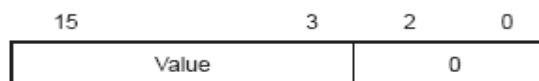


Рисунок 21-20 – Выравнивание данных для компадирования по A-закону в DXR

При декомпадировании данные декодируются как 16 битное слово. При этом декомпадированные данные в приемнике расположены также как и в регистре DXR при передаче (см выше). Далее эти данные могут быть выровнены по границе согласно установленному в SPCR.JBOUND и при необходимости расширен знак.

21.25 Кодирование/декодирование кодом Манчестер-2

При кодировании кодом Манчестер-2 размер передаваемого слова увеличивается вдвое и при настройках приемника/передатчика необходимо указывать удвоенную длину слова.

Так, если будет производиться передача/прием данных кодом манчестер-2 в кадрах с 1-ой фазой и размерами кадра 1 слово и слова 8 бит с задержкой передачи однобитовый интервал, то необходимо установить следующие настройки:

- MPHASE = 0b , однофазный кадр;
- CODEC = 001b (0x1), кодирование/декодирование кодом Манчестер-2;
- FLEN_P0 = 0000000b (0x00), кадр из одного слова;
- WLEN_P0 = 00111b (0x0F), слово из восьми бит;
- DELAY = 01b (0x1), задержка один битовый интервал.

21.26 Порядок передачи бит в словах

В контроллере McBSP предусмотрена реверсивная передача бит – передача, как с наименее, так и с наиболее значащих битов. Управление в передатчике и приемнике производится независимо битами MSB1st. Для передачи/приема данных начиная со старшего бита (MSB) необходимо установить бит MSB1st = 1, а для передачи начиная с наименее значащего (LSB) MSB1st = 0.

Управление направлением передачи доступно во всех режимах работы и диапазонах длин передаваемого/принимаемого слова.

21.27 Программируемые кадровая и битовая синхронизации

В контроллере McBSP есть несколько возможных вариантов подключения кадровой и битовой синхронизации для приемника и передатчика. И та и другая могут быть взяты от внешнего или от внутреннего источников. Приемник и передатчик имеют независимые настройки управления кадровой и битовой синхронизацией.

Схему подключений синхронизации демонстрирует Рисунок 21-21.

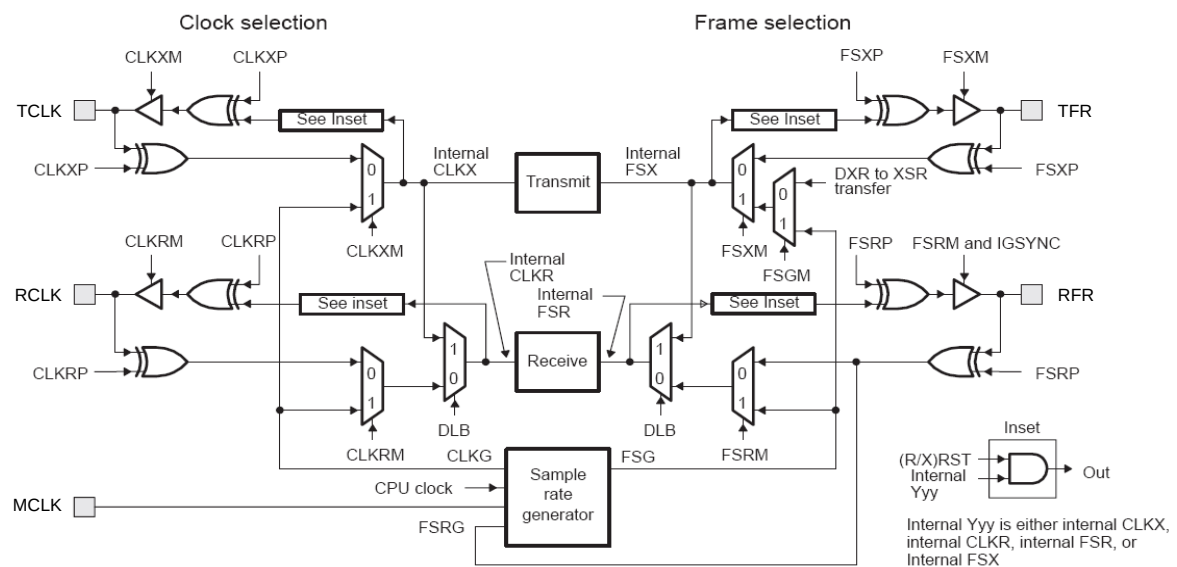


Рисунок 21-21 – Структурная схема модуля управления синхронизацией

Примечание – После смены источника частоты синхронизации для того или иного блока необходимо произвести программный сброс данного блока, в противном случае работа блока не гарантируется.

21.28 Внутренний генератора кадровой и битовой синхронизации

Внутренний генератор кадровой и битовой синхронизации структурно скомпонован из трех стадий деления опорной частоты для формирования заданных битового и кадрового синхроимпульсов. Рисунок 21-22 изображает схему генератора.

- Делитель опорного сигнала (задается CLKDIV). Формирует сигнал битовой синхронизации CLKG.
- Формирователь кадрового синхросигнала (задается FPER). Задаст межкадровый интервал.
- Формирователь кадрового импульса (задается FWID). Формирует кадровый синхроимпульс заданной длины и периода в соответствии с межкадровым интервалом.

Сигналы CLKG и FSG могут быть использованы для синхронизации битов и кадров приемника/передатчика. В качестве опорной частоты для генератора может быть использована внутренняя частота (частота работы параллельного интерфейса), либо частота опорная частота от внешнего источника.

Также генератор позволяет осуществлять синхронизацию от внешнего кадрового импульса.

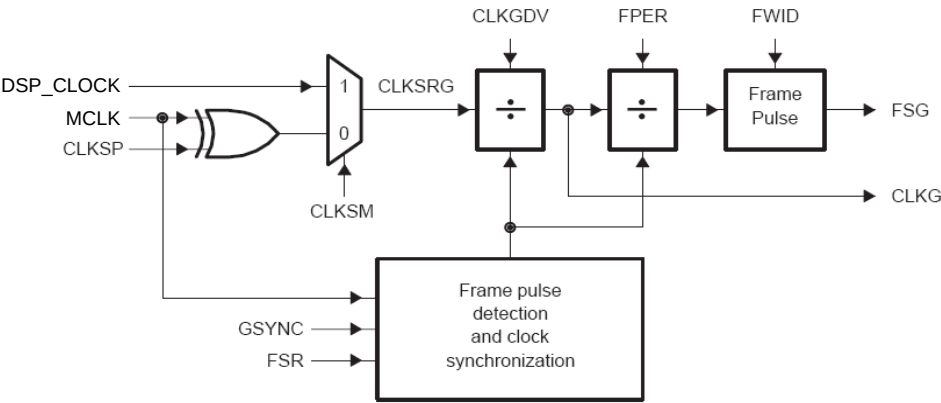


Рисунок 21-22 – Структурная схема генератора кадровой и битовой синхронизации

21.28.1 SRGRH

Таблица 21-16 – Регистр SRGRH

Номер	15	14	13	12	11...8	7...0
Доступ						
Сброс	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0
	GSYNC	FSGM	CLKSM	CLKSP	FPER[11:8]	FPER[7:0]

Таблица 21-17 – Описание бит регистра SRGRH

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15	GSYNC	Выбор режима синхронизации генератора 0 – режим FreeRun. Без синхронизации 1 – синхронизации от внешнего источника. Внешним источником является вход (!) кадровой синхронизации приемника (SFSR). * Период следования кадровых синхроимпульсов определяется SFSR ** Для корректной работы необходимо чтобы вывод кадровой синхронизации был сконфигурирован как вход PCR.FSRM = 0
14	FSGM	Выбор режима формирования кадрового сигнала 0 – кадровая синхронизация формируется генератором 1 – кадровый синхроимпульс формируется при перемещении данных из FIFO передатчика XBR в сдвиговый регистр передатчика XSR. * Режим может быть активен только при следующих условиях: SRSGR.GSYNC = 0; MPHASE = 0; MCM = 0.

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
13	CLKSM	Выбор источника опорного тактового сигнала: 0 – генератор тактируется от внешнего источника (MCLK); 1 – генератор тактируется от системного тактового сигнала
12	CLKSP	Полярность тактового сигнала: 0 – кадровый и битовый синхросигналы формируются по положительному фронту опорного тактового сигнала; 1 – кадровый и битовый синхросигналы формируются по отрицательному фронту опорного тактового сигнала
11...0	FPER	Период следования кадровых синхроимпульсов (-1). Диапазон от 1 до 4096 битовых интервалов

21.28.2 SRGRL

Таблица 21-18 – Регистр SRGRL

Номер	15...8	7...0
Доступ		
Сброс	R/W,+0	R/W,+0
	FWID[7:0]	CLKGDV[7:0]

Таблица 21-19 – Описание бит регистра SRGRL

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15...8	FWID[7:0]	Длительность кадровых синхроимпульсов (-1). Диапазон от 1 до 256 битовых интервалов
7...0	CLKGDV[7:0]	Коэффициент деления тактового сигнала для формирования битового синхроимпульса. Диапазон от 1 до 256 тактов опорной частоты

21.29 Сброс генератора кадровой и битовой синхронизации

Сброс и настройка генератора кадровой и битовой синхронизации производится следующим образом:

1 После аппаратного или программного сброса McBSP генератор находится в исходном состоянии. Программный сброс может быть осуществлен через внешний программный сброс или посредством бита регистра управления (GRST = 1). Сброс генератора кадровой синхронизации, кроме того, осуществляется посредством установки бита FRST = 1.

2 Если необходимо перевести приемник и передатчик в состояние сброса XRST и RRSST = 1.

3 Далее необходимо установить в SRGR настройки требуемых параметров работы генератора, таких как коэффициент деления тактовой частоты, частота и длительность кадровых синхроимпульсов и пр.

4 Обожждать несколько тактов опорной частоты работы генератора для установки соответствующих режимов.

- 5 Включить генератор битовой синхронизации, установив ($GRST = 0$)
- 6 Обождать несколько тактов опорной частоты работы генератора для его перехода в соответствующий режим.
- 7 Если необходимо вывести приемник и передатчик из состояния сбросом битов $XRST = 0$ и $RRST = 0$.
- 8 Для выхода генератора битовой синхронизации в режим необходимо не более $(1 + CLKGDV)$ тактов опорной частоты генератора.
- 9 После произведения всех предустановок и помещения данных на передачу в FIFO передатчика включить генератор кадровых синхроимпульсов установив ($FRST = 0$). Если генератор кадровых синхроимпульсов был включен одновременно с включением генератора битовой синхронизации, то его выход в режим будет одно временно с выходом в режим генератора битовой синхронизации. Для исключения отправки «пустого кадра» необходимо поместить данные на передачу в FIFO до вывода генератора кадровой синхронизации из состояния сброса.
- 10 Кадровой синхронизации из состояния сброса.

21.30 Генератор битовой синхронизации

При установке битов $FSXM = 1$ и $FSRXM = 1$ приемник и передатчик работают от внутреннего генератора кадровой синхронизации. Для передатчика и для приемника выбор источника битовой синхронизации осуществляется независимо. Доступные настройки:

- Тактовый сигнал для генератора может быть от внешнего источника или от тактового сигнала параллельного интерфейса (системный).
- Тактовый сигнал может быть поделен на целое число от 1 до 256. При этом в поле $CLKGDV$ задается коэффициент деления -1 (при $CLKGDV = 0$ коэффициент деления будет 1). Схема делителя всегда работает по положительному фронту опорного тактового сигнала. При делении опорной частоты скважность сигнала битовой синхронизации будет зависеть от четности и величины коэффициента деления:
 - для четных коэффициентов деления (нечетных $CLKGDV$): $Q = 50\%$;
 - для нечетных коэффициентов деления (четных $CLKGDV > 0$): $Q = (1 + 1/(CLKGDV + 1))/2 \cdot 100\%$.

21.31 Полярность тактового сигнала генератора

Генератор бытовой синхронизации может быть запрограммирован как для работы с прямым тактовым сигналом, так и с инверсным. Для работы с инверсным опорным тактовым сигналом необходимо установить бит $CLKSP = 1$.

21.32 Битовая и кадровая синхронизация

При выборе режима работы генератора от внешнего опорного тактового сигнала ($MCLK$) бит $GSYNC$ может применяться для управления синхронизацией фронта сигнала битовой синхронизации $CLKG$ с фронтом опорного тактового сигнала. При установленном бите $GSYNC = 1$ фазы сигнала битвой синхронизации и тактового сигнала синхронизирован, в противном случае по фазе они не согласованы. При установленном бите $GSYNC = 1$ синхронизация по фазе осуществляется по сигналу кадровой

синхронизации приемника, поэтому данный вывод должен быть запрограммирован как вход. Синхронизация осуществляется по первому активному уровню опорного тактового сигнала независимо от длительности кадрового синхроимпульса. Следует обратить внимание, что в этом режиме поле SRGR.FPER игнорируется, поскольку период следования кадров определяется только входной кадровой синхронизацией.

На рисунках (Рисунок 21-23 и Рисунок 21-24) приведены примеры диаграмм для разных настроек битовой и кадровой синхронизации.

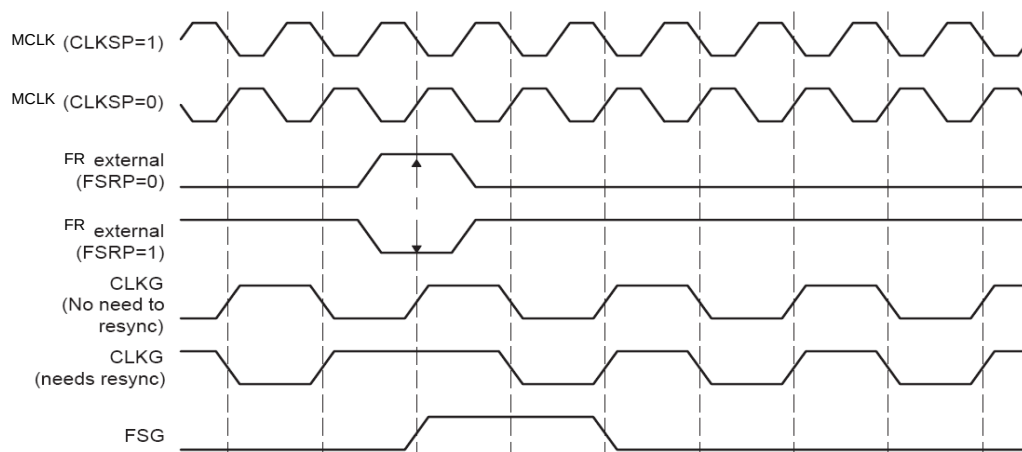


Рисунок 21-23

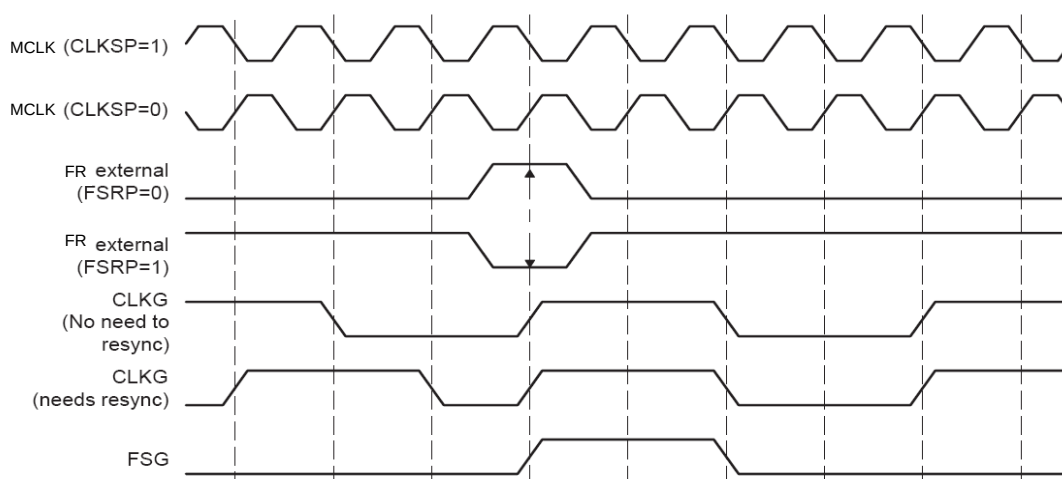


Рисунок 21-24

Также в данном режиме (GSYNC = 1) передатчик может работать синхронно с приемником. Для этого необходимы следующие настройки:

- 1 Вывод SFSX должен быть настроен для работы от внутреннего генератора (FSXM = 1 и CLKXM = 1);
- 2 Внешние цепи кадровой и битовой синхронизации приемника и передатчика (за исключением сигнала кадровой синхронизации приемника) должны быть отключены от внешних источников, поскольку в данном режиме эти выводы работают как выходы.

21.33 Режим КЗ

Для тестирования и прочих нужд (к примеру, кодирования или декодирования данных) в McBSP предусмотрен режим КЗ. В данный режим McBSP вводится установкой бита SPCR.DLB = 1. В этом режиме все сигналы передатчика подключаются в выводам

приемника. Данный режим штатно предназначен для тестирования McBSP программным путем ЦП без внешних инструментов. Также данный режим может быть использован для кодирования и декодирования массивов данных, а также для выравнивания по заданной границе и расширения знака без участия ЦП.

21.34 Генератор кадровой синхронизации

Подобно битовой синхронизации кадровая настраивается для приемника и для передатчика независимо. Сброс генератора кадровой синхронизации осуществляется битом $FRST = 1$.

- настройка периода следования (в битовых интервалах) и длительности кадрового синхроимпульса
- независимая настройка для приемника и передатчика с выбором работы от внешнего или внутреннего источника.

Примечание – генератор кадровой синхронизации работает от генератора битовой синхронизации и, таким образом, для его работы необходима работа генератора битовой синхронизации (!).

21.35 Период и длина кадрового синхроимпульса

В генераторе кадровой синхронизации возможна настройка частоты следования и длительности формируемого кадрового синхроимпульса. Данная настройка осуществляется в битовых интервалах задаваемых генератором битовой синхронизации и задается полями $SRGR.FPER$ и $SRGR.FWID$ соответственно. Аппаратно они представлены декрементирующими счетчиками разрядности 12 и 8 бит соответственно и, таким образом, могут быть запрограммированы период следования кадрового синхроимпульса от 1 до 4096 битовых интервалов и его длина от 1 до 256 битовых интервалов. При этом в соответствующем поле указывается значение за вычетом 1. То есть, к примеру длине синхроимпульса 1 битовый интервал соответствует значение 0 в поле $SRGR.FWID$.

Рекомендуется задавать длительность кадрового синхроимпульса менее длины передаваемого/принимаемого слова. Также рекомендуется задавать частоту следования кадровых синхроимпульсов более их длины, в противном случае McBSP будет формировать кадровые синхроимпульсы с заданной частотой следования и длиной активного уровня равной половине частоты следования.

Ниже приведен пример формирования кадровой синхронизации (Рисунок 21-25).

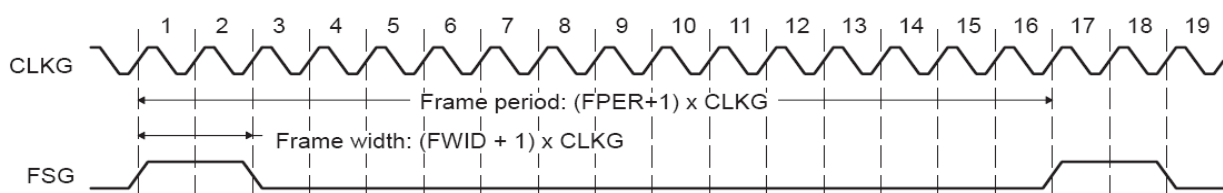


Рисунок 21-25 – Пример формирования кадровой синхронизации

21.36 Примеры битовой и кадровой синхронизации в различных форматах

В данном разделе приведены примеры настроек для некоторых форматов синхронных последовательных интерфейсов.

21.36.1 ST-BUS

Для работы с ST-BUS необходимы следующие настройки McBSP:

- PCR.FS[X/R]P = 1, кадровые синхроимпульсы имеют активный низкий уровень;
- PCR.CLK[X/R]M = 1, битовая синхронизация формируются внутренним генератором битовой синхронизации;
- SRGR.GSYNC = 1, генератор кадровой и битовой синхронизации синхронизируется внешним синхроимпульсом;
- SRGR.CLKSM = 1, генератор кадровой и битовой синхронизации тактируется внешним источником тактовой частоты;
- SRGR.CLKSP = 1, генератор кадровой и битовой синхронизации синхронизируются по отрицательному фронту тактовой частоты;
- SRGR.CLKGDV = 0, прием ведется на половине частоты опорного тактового сигнала;
- MPHASE = 0, кадр содержит одну фазу;
- FLEN_P0 = 11111b, в кадре 32 слова;
- WLEN_P0 = 111b, каждое слово кадра 8 бит;
- DELAY = 0 задержка данных в канале отсутствует.

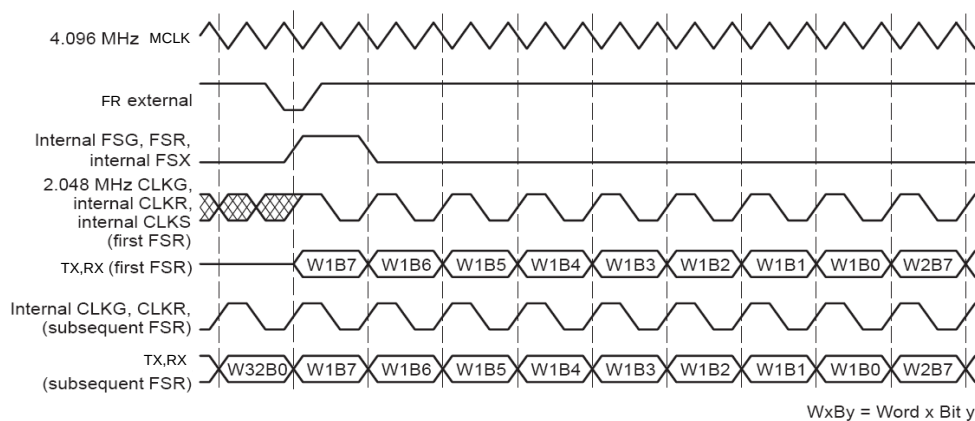


Рисунок 21-26 – Диаграмма работы в режиме ST-BUS

21.36.2 ST-BUS с удвоенной частотой

Для работы с Mitel ST-BUS необходимы следующие настройки McBSP:

- PCR.FS[X/R]P = 1, кадровые синхроимпульсы имеют активный низкий уровень;
- PCR.CLK[X/R]M = 1, битовая синхронизация формируются внутренним генератором битовой синхронизации;
- SRGR.GSYNC = 1, генератор кадровой и битовой синхронизации синхронизируется внешним синхроимпульсом;

- SRGR.CLKSM = 1, генератор кадровой и битовой синхронизации тактируется внешним источником тактовой частоты;
- SRGR.CLKSP = 1, генератор кадровой и битовой синхронизации синхронизируются по отрицательному фронту тактовой частоты;
- SRGR.CLKGDV = 1, прием ведется на половине частоты опорного тактового сигнала;
- [X/R]CR.MPHASE = 0, кадр содержит одну фазу;
- [X/R]CR.FLEN_P0 = 11111b, в кадре 32 слова;
- [X/R]CR.WLEN_P0 = 111b, каждое слово кадра 8 бит;
- [X/R]CR.DELAY = 0 задержка данных в канале отсутствует.

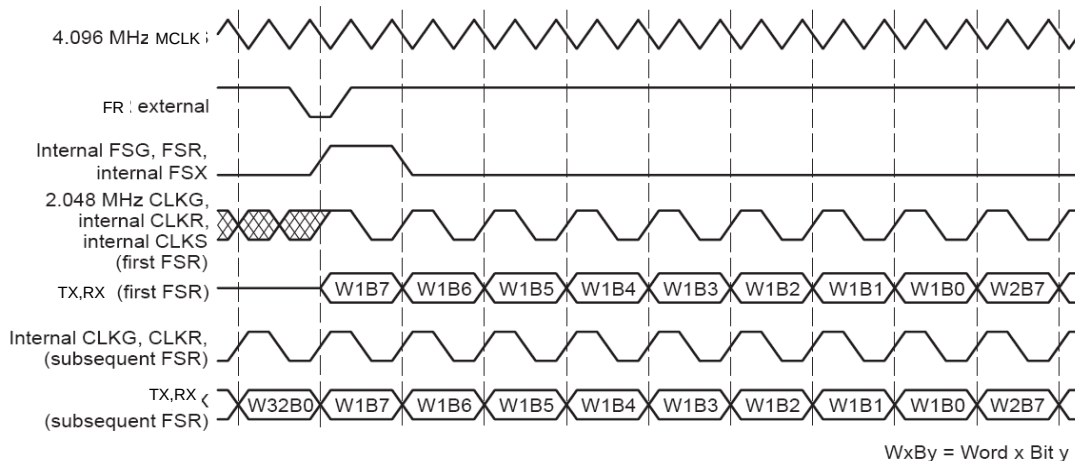


Рисунок 21-27 – Диаграмма работы в режиме ST-BUS с удвоенной частотой

21.37 Работа в многоканальном режиме с выбором каналов

В McBSP предусмотрена независимая настройка выборки определенных каналов из множества для приема/передачи данных. Каждый канал представлен потоком данных с временным разделением (ВРК). В данном режиме все каналы имеют длину WLEN_P0 и количество FLEN_P0. Работа бита MPHASE блокируется.

При использовании ВРК ЦП зачастую имеет необходимость работы только в некоторых из них. Для этого имеется возможность маскирования активных и неактивных каналов для снижения нагрузки на системную шину. Одновременно могут быть активны некоторые или все из 128 каналов. Весь набор каналов разбит на 8 блоков по 16 каналов. Все блоки включаются независимо. Маска каналов для четных блоков содержится в регистрах XCERL и RCERL, а для нечетных XCERN и RCERN. Для всех четных блоков маска одинакова. Аналогично и для нечетных блоков.

Если отключен канал на прием, то данные в сдвиговый регистр приемника RSR не поступают и данные в нем не изменяются.

В передатчике в отключенном канале вывод DX находится в высокоимпедансном состоянии. Сдвига данных также не происходит.

При отключенных каналах обработка передач между сдвиговыми регистрами и FIFO осуществляется и все статусные признаки изменяются, кроме связанных с собственно приемом/передачей данных.

21.37.1 MCRH, MCRL, XMCR, RMCR

Таблица 21-20 – Регистры MCRH, MCRL, XMCR, RMCR

Номер	15	14	13	12	11	10
Доступ						
Сброс	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0
	BLK7_EN	BLK6_EN	BLK5_EN	BLK4_EN	BLK3_EN	BLK2_EN

Номер	9	8	7...5	4...2	1, 0
Доступ					
Сброс	R/W,+0	R/W,+0	R,+0	R/W,+0	R/W,+0
	BLK1_EN	BLK0_EN	CUR_BLK		MCM

Таблица 21-21 – Описание бит регистров MCRH, MCRL, XMCR, RMCR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15...8	BLK_EN[7:0]	Разрешение активности каналов в Блоке: Бит 0 – каналы 0 .. 15; Бит 1 – каналы 16 .. 31; Бит 2 – каналы 32 .. 47; Бит 3 – каналы 48 .. 63; Бит 4 – каналы 64 .. 79; Бит 5 – каналы 80 .. 95; Бит 6 – каналы 96 .. 111; Бит 7 – каналы 112 .. 127. 0 – блок каналов неактивен; 1 – блок каналов активен
7...5	CUR_BLK[2:0]	Номер текущего блока в кадре
4...2	-	Зарезервировано
1...0	MCM[1:0]	Выбор многоканального режима: 00 – все каналы включены; 01 – все каналы выключены; 10 – включены каналы согласно маске в регистрах XCER и RCER; 11 – выключены каналы согласно маске в регистрах XCER и RCER

21.38 Включение многоканального режима

Многоканальный режим для приемника и для передатчика включается независимо битами RMCR.MCM[1] = 1 и XMCR.MCM[1] = 1 соответственно.

Включение и маскирование каналов

В многоканальном режиме в McBSP доступны до 128 каналов разделенных на восемь блоков по 16 каналов в каждом. Каждый блок может быть включен (BLK_EN[номер блока] = 1) или отключен (BLK_EN[номер блока] = 0) независимо от других. Включение блока разрешает или запрещает работу каналов посредством регистра

разрешения работы активности каналов XCER и RCER. При этом для четных блоков применяется маска из младшей части регистра разрешения активности каналов (XCERL и RCERL), а для нечетных – старшей (XCERH и RCERH).

В зависимости от режима работы MCM могут быть включены или отключены все 128 каналов, включены каналы соответствующие «1» в маске или наоборот – включены все каналы соответствующие «0» в маске.

Рисунок 21-28 демонстрирует распределение каналов и блоков в кадре.

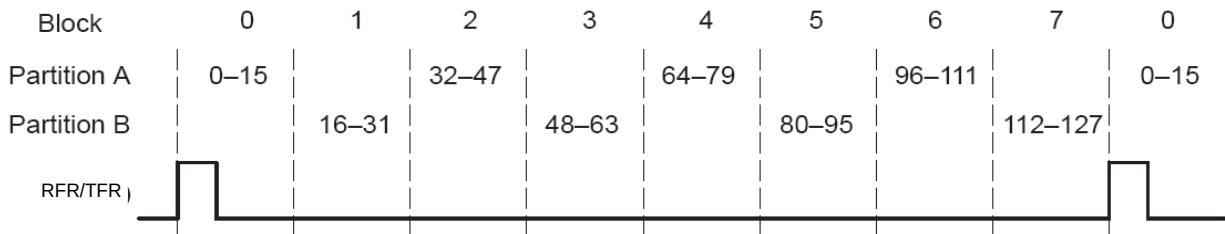


Рисунок 21-28 – Распределение каналов и блоков в кадре

21.39 Регистр управления активностью каналов

Регистр управления активностью позволяет управлять активностью того или иного канала в многоканальном режиме работы McBSP. Поскольку число разрешений активности каналов менее доступного количества каналов в кадре, то активность того или иного канала кроме также определяется активностью или неактивностью блока (см выше). Регистр управления активностью каналов разбит на младшую и старшую части, которые активны в четных и нечетных блоках соответственно.

Примечание – Настройку активности каналов и блоков не рекомендуется производить во время приема/передачи данных.

21.39.1 XCERH, RCERH

Таблица 21-22 – Регистры XCERH, RCERH

Номер	31[15]	30[14]	29[13]	28[12]	27[11]	26[10]	25[9]	24[8]
Доступ								
Сброс	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0
	CH31_EN	CH30_EN	CH29_EN	CH28_EN	CH27_EN	CH26_EN	CH25_EN	CH24_EN

Номер	23[7]	22[6]	21[5]	20[4]	19[3]	18[2]	17[1]	16[0]
Доступ								
Сброс	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0
	CH23_EN	CH22_EN	CH21_EN	CH20_EN	CH19_EN	CH18_EN	CH17_EN	CH16_EN

Таблица 21-23 – Описание бит регистров XCERH, RCERH

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15...0	CH_EN[15:0]	Разрешение активности канала N = номер бита + 16: 0 – канал неактивен; 1 – канал активен

21.39.2 XCERL, RCERL

Таблица 21-24 – Регистры XCERL, RCERL

Номер	15[15]	14[14]	13[13]	12[12]	11[11]	10[10]	9[9]	8[8]
Доступ								
Сброс	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0
	CH15_EN	CH14_EN	CH13_EN	CH12_EN	CH11_EN	CH10_EN	CH9_EN	CH8_EN

Номер	7[7]	6[6]	5[5]	4[4]	3[3]	2[2]	1[1]	0[0]
Доступ								
Сброс	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0
	CH7_EN	CH6_EN	CH5_EN	CH4_EN	CH3_EN	CH2_EN	CH1_EN	CH0_EN

Таблица 21-25 – Описание бит регистров XCERL, RCERL

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15...0	CH_EN[15:0]	Разрешение активности канала N = номер бита: 0 – канал неактивен; 1 – канал активен

21.39.3 Интерфейс A-bis

В режиме A-bis McBSP может принимать и передавать до 1024 бит тракта ИКМ. Приемник *может* извлечь все 1024 бита из кадра ИКМ в соответствии с заданной маской и выдать на ЦП прерывание при упаковке в слова до 16 бит.

22 Системный Таймер (DSP)

Таймер DSP является программируемым таймером, который управляется тремя регистрами и может быть использован для генерации прерываний или запросов DMA (прерывания для DSP и ядер RISC, запросы для DMA DSP и DMA RISC). Временное разрешение таймера определяется частотой синхросигналов DSP. Высокий динамический диапазон таймера достигнут 16-битовым счетчиком с четырехбитовым предварительным делителем частоты.

22.1 Регистры таймера (DSP)

Таймер состоит из трех регистров отображенных в памяти (TIM, PRD и TCR). Эти три регистра и их соответствующие адреса указаны в таблице ниже (Таблица 22-1).

Таблица 22-1 – Регистры таймера подсистемы DSP

Адрес RISC	Адрес DSP	Название	Описание
0x3000_0070	0038h	TIM	Регистр таймера
0x3000_0072	0039h	PRD	Регистр периода таймера
0x3000_0074	003Ah	TCR	Регистр управления таймера

22.1.1 Регистры таймера (TIM)

Регистр текущего значения таймера (TIM)

16-битный регистр таймера, адрес которого отображен в памяти, загружается значением регистром периода (PRD) и декрементируется.

Таблица 22-2 – Регистр TIM

Номер	15...0
Доступ	RW
Сброс	0
	Текущее значение таймера

Таблица 22-3 – Биты регистра управления таймером TIM

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15...0	TIM	Текущее значение таймера

22.1.2 Регистр периода таймера (PRD)

16-битный регистр периода таймера отображаемый в памяти (PRD), используется для того, чтобы перезагружать регистр таймера (TIM).

Таблица 22-4 – Регистр PRD

Номер	15...0
Доступ	RW
Сброс	0
	Начальное значение таймера

Таблица 22-5 – Биты регистра управления таймером PRD

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15...0	PRD	Начальное значение таймера

22.1.3 Управляющий регистр таймера (TCR)

16-битный регистр управления таймера, адрес которого отображен в памяти (TCR), содержит управляющие биты и биты состояния таймера. Ниже приведены битовые поля TCR (Таблица 22-6) и их описание (Таблица 22-7).

Таблица 22-6 – Регистр TCR

Номер	15...12	11	10	9...6	5	4	3...0
Доступ	NA	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0	0
	-	Soft	Free	PCS	TRB	TSS	TDDR

Таблица 22-7 – Биты регистра управления таймером TCR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15...12	-	Зарезервировано
11	Soft	Когда бит 10 сброшен, бит выбирает режим таймера: 0 – остановки таймера немедленные; 1 – остановки тогда, когда значение таймера декрементировано до 0
10	Free	Используется вместе с битом 11. Когда бит сброшен, бит 11 выбирает режим таймера: 0 – бит 11 выбирает режим таймера; 1 – таймер работает независимо от бита 11
9-6	PSC	Счетчик предварительного делителя частоты. Когда PSC декрементируется до нуля, значение регистра TIM декрементируется на 1, PSC снова загружается содержимым регистра TDDR
5	TRB	Перезагрузка таймера. Сбрасывает встроенный таймер. При установке TRB в 1, в регистр TIM загружается значение регистра PRD, а в регистр PSC загружается значение регистра TDDR. Всегда читается как 0
4	TSS	Остановка таймера: 0 – таймер работает; 1 – таймер остановлен
3...0	TDDR	Базовое значение предварительного счетчика

23 Работа таймера (DSP)

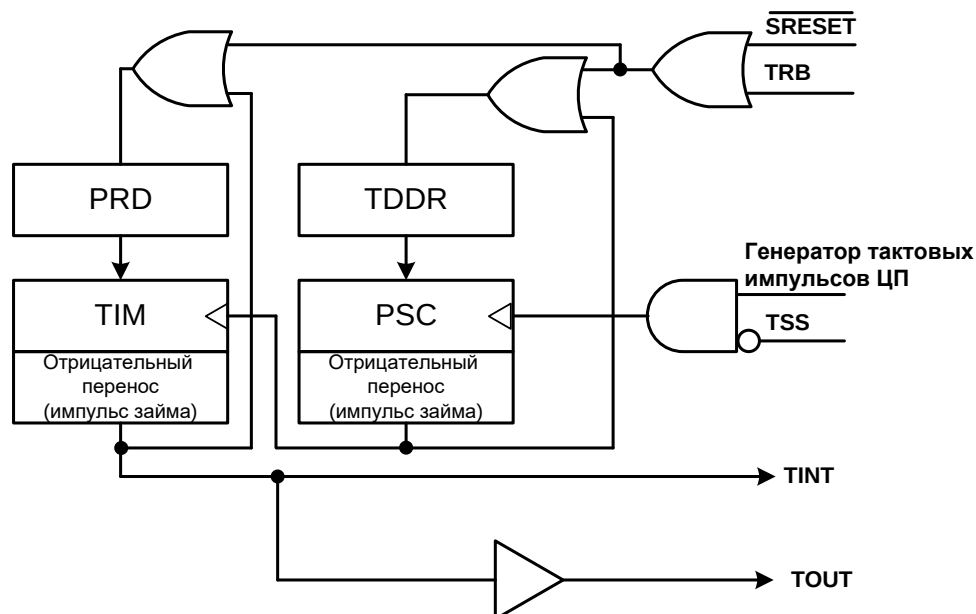


Рисунок 23-1 – Блок-схема таймера

Таймер управляется предварительным делителем частоты (PSC), который декрементируется на единицу каждый такт синхросигнала DSP. Всякий раз, когда счетчик PSC достигает нуля, он загружается значением поля TDDR, а регистр TIM декрементируется. Когда обнуляются оба счетчика (TIM и PSC) полный цикл счета считается выполненным, регистр TIM загружается значением регистра PRD, поле PSC значением поля TDDR, и цикл счета начинается снова. При завершении каждого цикла вырабатывается прерывание для DSP и RISC и запрос прерываний для DMA RISC и DMA DSP.

Периодичность прерывания от таймера (TINT) равняется частоте синхросигнала DSP поделенной на два независимых сомножителя:

$$TINT \text{ rate} = \frac{1}{t_{c(C)} \times u \times v} = \frac{1}{t_{c(C)} \times (TDDR + 1) \times (PRD + 1)},$$

где $t_{c(C)}$ – период синхрогенератора DSP;

u – содержание TDDR плюс 1;

v – содержание PRD плюс 1.

24 Контроллер DMA (DSP)

Контроллер прямого доступа в память DSP реализован для ускоренного переноса данных внутри подсистемы DSP. Данный контроллер имеет доступ только внутри подсистемы DSP (кроме регистров ядра DSP) (Таблица 24-1). При этом, в качестве адресов источника, назначения и управляющих структур используются система адресации RISC (т.е. общее адресное пространство и байтовые адреса).

24.1 Особенности контроллера DMA (DSP)

Контроллер DMA DSP аналогичен контроллеру DMA RISC, за исключением следующих особенностей:

- Количество активных каналов сокращено до 16. Аппаратные запросы реализованы только от периферийных модулей DSP части (Таблица 24-3). При этом, те каналы, для которых определены аппаратные запросы могут быть настроены либо на программные, либо на аппаратные запросы. Остальные каналы способны обрабатывать только программные запросы.
- Доступ к регистрам DMA контроллера возможен как со стороны DSP, так и со стороны RISC, соответственно, программные запросы и управление модулем может быть осуществлен обоими ядрами.
- Для обеспечения возможности доступа со стороны 16-разрядного DSP регистры контроллера разбиты на младшую и старшую часть каждый (Таблица 24-2). При этом, со стороны RISC младшая и старшие части регистров могут быть записаны одновременно (по 32 бита) по адресу младшей части.

Для получения подробной информации по функционированию контроллера DMA RISC см. раздел 17 «Контроллер прямого доступа в память MDR_DMA».

24.2 Возможности доступа DMA (DSP)

Контроллер DMA DSP части имеет доступ только внутри подсистемы DSP. Ему доступно все адресное пространство памяти программ и памяти данных, за исключением. В качестве адресов источника и назначения необходимо указывать адреса также как в подсистеме RISC (32-битный байтовый адрес). Контроллер DMA DSP может быть использован для модификации памяти программ внутри подсистемы DSP.

Таблица 24-1 – Таблица доступа к секторам памяти со стороны DMA DSP

Адрес DMA DSP	Сектор	Тип со стороны DMA (ВЫЗ)
0x0000_0000	BOOT ROM	NA
0x0800_0000	EEPROM	NA
0x1000_0000	EXTERNAL BUS	NA
0x2000_0000	SYSTEM RAM	NA
0x2200_0000	SYSTEM RAM Bit Band Region	NA
0x3000_0000	Регистры ядра DSP	NA
0x3000_0040	Регистры периферийных модулей DSP	WR
0x3000_0100	Память данных DSP	WR
0x3002_0000	Память программ DSP	WR

Адрес DMA DSP	Сектор	Тип со стороны DMA (ВЫЗ)
0x3000_0040	Регистры периферийных модулей RISC	NA
0x5000_0000	EXTERNAL BUS	NA
Обозначения: NA – нет доступа; RO – только чтение; WR – полный доступ (чтение/запись)		

24.3 Регистры контроллера DMA(DSP)

Набор регистров и их функции контроллера DMA DSP такие же, как и контроллера DMA RISC. Регистры могут быть доступны как со стороны RISC так и со стороны DSP (Таблица 24-2). Т.к. количество каналов контроллера сокращено, во всех регистрах доступны биты управления каналами с 1 по 16.

Таблица 24-2 – Описание регистров контроллера DMA DSP

Адрес RISC	Адрес DSP	Регистр	Описание
0x3000_00C0	0060h	DMA_STATUSL	Статусный регистр ПДП
0x3000_00C2	0061h	DMA_STATUSH	
0x3000_00C4	0062h	DMA_CONFIGL	Регистр конфигурации ПДП
0x3000_00C6	0063h	DMA_CONFIGH	
0x3000_00C8	0064h	CTRL_BASE_PTRL	Регистр базового адреса управляющих данных каналов
0x3000_00CA	0065h	CTRL_BASE_PTRH	
0x3000_00CC	0066h	ALT_CTRL_BASE_PTRL	Регистр базового адреса альтернативных управляющих данных каналов
0x3000_00CE	0067h	ALT_CTRL_BASE_PTRH	
0x3000_00D0	0068h	DMA_WAITONREG_STATUSL	Регистр статуса ожидания запроса на обработку каналов
0x3000_00D2	0069h	DMA_WAITONREG_STATUSH	
0x3000_00D4	006Ah	CHNL_SW_REQUESTL	Регистр программного запроса на обработку каналов
0x3000_00D6	006Bh	CHNL_SW_REQUESTH	
0x3000_00D8	006Ch	CHNL_USEBURST_SETL	Регистр установки пакетного обмена каналов
0x3000_00DA	006Dh	CHNL_USEBURST_SETH	
0x3000_00DC	006Eh	CHNL_USEBURST_CLRL	Регистр сброса пакетного обмена каналов
0x3000_00DE	006Fh	CHNL_USEBURST_CLRH	
0x3000_00E0	0070h	CHNL_REQ_MASK_SETL	Регистр маскирования запросов на обслуживание каналов
0x3000_00E2	0071h	CHNL_REQ_MASK_SETH	

Адрес RISC	Адрес DSP	Регистр	Описание
0x3000_00E4	0072h	CHNL_REQ_MASK_CLRL	Регистр очистки маскирования запросов на обслуживание каналов
0x3000_00E6	0073h	CHNL_REQ_MASK_CLRH	
0x3000_00E8	0074h	CHNL_ENABLE_SETL	Регистр установки разрешения каналов
0x3000_00EA	0075h	CHNL_ENABLE_SETH	
0x3000_00EC	0076h	CHNL_ENABLE_CLRL	Регистр сброса разрешения каналов
0x3000_00EE	0077h	CHNL_ENABLE_CLRH	
0x3000_00F0	0078h	CHNL_PRI_ALT_SETL	Регистр установки первичной/альтернативной структуры управляющих данных каналов
0x3000_00F2	0079h	CHNL_PRI_ALT_SETH	
0x3000_00F4	007Ah	CHNL_PRI_ALT_CLRL	Регистр сброса первичной/альтернативной структуры управляющих данных каналов
0x3000_00F6	007Bh	CHNL_PRI_ALT_CLRH	
0x3000_00F8	007Ch	CHNL_PRIORITY_SETL	Регистр установки приоритета каналов
0x3000_00FA	007Dh	CHNL_PRIORITY_SETH	
0x3000_00FC	007Eh	CHNL_PRIORITY_CLRL	Регистр сброса приоритета каналов
0x3000_00FE	007Fh	CHNL_PRIORITY_CLRH	
-	0086h	ERR_CLRL	Регистр сброса флага ошибки
-	0087h	ERR_CLRH	Регистр сброса флага ошибки

24.4 Распределение аппаратных запросов по каналам DMA(DSP)

Количество каналов контроллера сокращено до 16. Из них аппаратные прерывания заведены на 10 каналов. Запрос на эти каналы может быть осуществлен как программным, так и аппаратным способом. Транзакции по остальным каналам могут быть активированы программно.

Таблица 24-3 – Распределение аппаратных запросов DMA DSP по каналам

Номер канала	Источник req	Источник sreq	Описание
0	mcBSP1 TX	mcBSP1 TX	Запрос DMA от mcBSP1 по передаче
1	mcBSP1 RX	mcBSP1 RX	Запрос DMA от mcBSP1 по приему
2	mcBSP2 TX	mcBSP2 TX	Запрос DMA от mcBSP2 по передаче
3	mcBSP2 RX	mcBSP2 RX	Запрос DMA от mcBSP2 по приему
4	mcBSP3 TX	mcBSP3 TX	Запрос DMA от mcBSP3 по передаче
5	mcBSP3 RX	mcBSP3 RX	Запрос DMA от mcBSP3 по приему
6	AUC_in	AUC_in	Запрос DMA от Аудиокодека по приему (АЦП)
7	AUC_out	AUC_out	Запрос DMA от Аудиокодека по передаче (ЦАП)
8	Timer	Timer	Запрос DMA от Таймера
9	Crypto	Crypto	Запрос DMA от Криптомодуля
10...15	-	-	Только программные запросы

25 Контроллер AudioCodec (DSP)

Характеристики контроллера:

- Одноканальный $\Delta\Sigma$ АЦП с разрядность выходных отчетов 16 бит;
- Одноканальный $\Delta\Sigma$ ЦАП с разрядность входных отчетов 16 бит;
- Дифференциальный и недифференциальный аналоговый вход/выход;
- Возможность микширования данных тракта АЦП и ЦАП;
- Встроенный аналоговый antialiasing фильтр;
- Ручная регулировка усиления в тракте АЦП;
- Блок имеет независимые каналы DMA для тракта АЦП и ЦАП;
- Блок формирует сигнал прерывания по внутренним событиям;
- Имеет встроенный формирователь синхросигналов;
- ЦАП имеет защиту от короткого замыкания;

Примечание – Все измерения проводились для частоты отсчетов – 8 кГц.

Таблица 25-1 – Параметры каналов ЦАП/АЦП

Параметр	Значение
Уровень «0» для ЦАП	1,5 В
Размах сигнала для ЦАП, р-р	2 В (max)
Уровень «0» для АЦП	1,5 В
Размах входного сигнала для АЦП	2 В (max)

25.1 Задание рабочей частоты аудиокodeка

Рабочая частота аудиокodeка задается при помощи регистра ADC_MCO_CLOCK (см. раздел 12 «Сигналы тактовой частоты»). Запись данных в регистры осуществляется на частоте PCLK. Для пересинхронизации данных перед их обработкой цифровой частью аудиокodeка реализован два отдельных интерфейса FIFO (для ЦАП и АЦП) по 16 16-битных отсчетов каждый.

25.2 Архитектура модуля

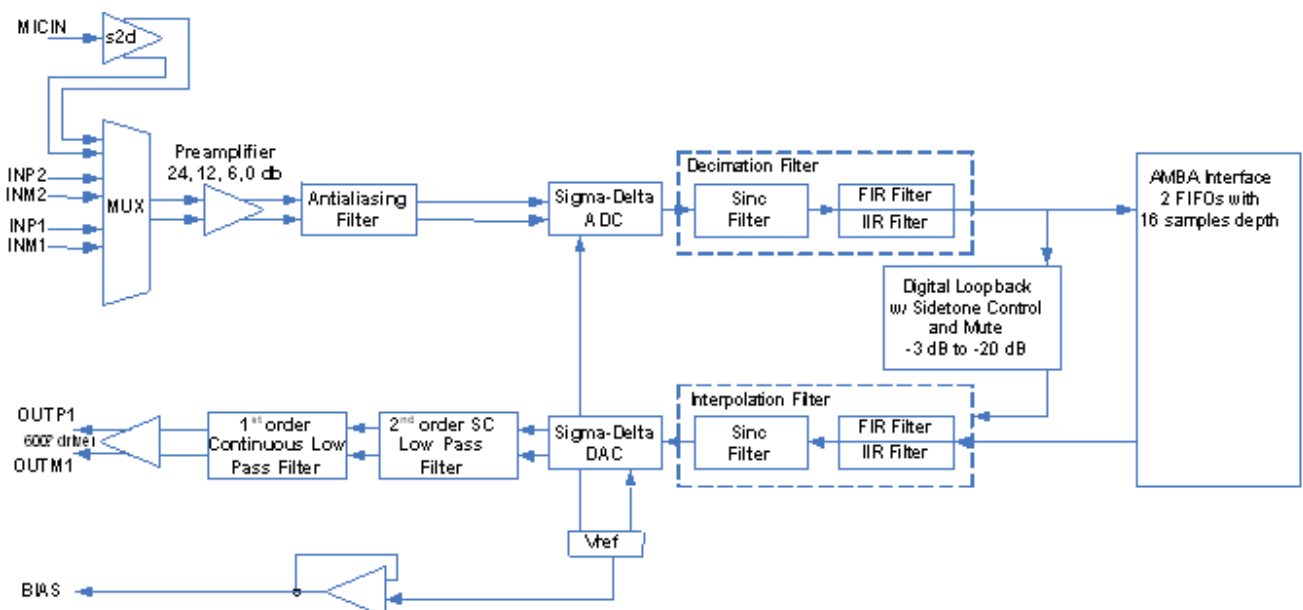


Рисунок 25-1 – Структура IP звукового codeка

Аналоговый фильтр нижних частот

Встроенный аналоговый фильтр нижних частот – фильтр с двумя полюсами, у которого есть ослабление на 20 дБ в 1 МГц.

Сигма-Дельта АЦП

Аналого-цифровой преобразователь (АЦП) – модулятор дельты сигмы с фиксированной частотой дискретизации 256 отсчетов рабочей частоты аудиокодека.

Изменение частоты дискретизации достигается изменением рабочей частоты аудиокодека.

Фильтр децимации

Фильтры децимации – это фильтр БИХ или КИХ, в зависимости от состояния бита IREN регистра общего управления кодеком. Фильтр КИХ обеспечивает выход линейной фазы $32/fs$ групповой задержки, тогда как фильтр БИХ генерирует нелинейную фазу с незначительной групповой задержкой. Фильтры децимации уменьшают цифровую скорость передачи данных до частоты сэмплирования. Это достигается коэффициентом передискретизации 128. Выход фильтра децимации 16-битные данные в дополнительном коде на частоте сэмплирования. BW фильтра (0.45 FS) и изменяется линейно в зависимости от частоты сэмплирования.

Интерполирующий фильтр

Фильтры интерполяции – это БИХ фильтр. Фильтр БИХ имеет нелинейный выход фазы с незначительной задержкой группы. Фильтр интерполяции передискретизирует цифровые данные в 64 раза поступающей частоты, в зависимости от заданной частоты дискретизации ЦАП. Выход фильтра используется в сигма-дельта ЦАП.

Цифровая обратная петля

Данные с выходов АЦП ослабляются и смешиваются со входами ЦАП. Уровень петли задается битами SIDETON регистра DACCTL.

Программируемый усилитель АЦП

В модуле имеется встроенный программируемый усилитель для управления уровнями сигнала на выходе АЦП. Программируемый усилитель может быть установлен с использованием поля ADGAIN регистра ADCCTL. Диапазон усилителя канала АЦП составляет от 20 до -42 дБ с шагом 1 дБ. Чтобы избежать внезапных скачков на уровнях сигнала с изменениями поля диапазона усиления, переход к новому значению ADGAIN происходит при пересечении нуля.

Программируемый усилитель ЦАП

В модуле имеется встроенный программируемый усилитель для управления уровнями сигнала на выходе ЦАП. Программируемый усилитель ЦАП может быть установлен с использованием поля DAGAIN регистра DACCTL. Диапазон усилителя канала ЦАП составляет от 20 до -42 дБ с шагом 1 дБ. Чтобы избежать внезапных скачков на уровнях сигнала с изменениями поля диапазона усиления ЦАП, переход к новому значению DAGAIN происходит при пересечении нуля.

25.3 Аналоговые площадки ввода-вывода

Для работы модуля реализованы три программируемых аналоговых входа и один программируемый аналоговый выход (Таблица 25-2). При помощи поля AINSEL регистра ADCCTL можно выбрать источник аналогового входа со входов MICIN, INP1/M1 или INP2/M2. Все аналоговые ввод/вывод могут быть как обычными, так и дифференциальными. Все аналоговые входы системы имеют внутренне смещение к 1,5 В. Управление аналоговым выходом осуществляется при помощи регистра DACCTL.

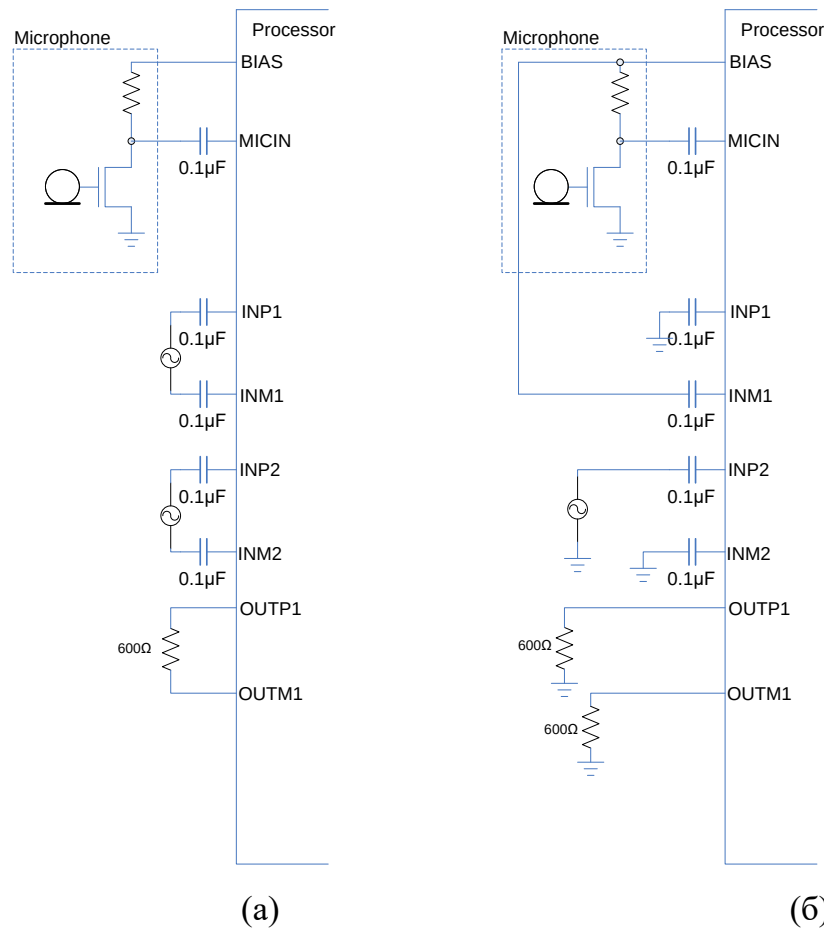


Рисунок 25-2 – Дифференциальное (а) и псевдо дифференциальное (б) подключение модуля

Таблица 25-2 – Выводы блока

Название вывода	Направление	Описание
MICIN	I	Вход микрофона
INP2	I	Положительный вход канала 2
INM2	I	Отрицательный вход канала 2
INP1	I	Положительный вход канала 1
INM1	I	Отрицательный вход канала 1
OUTP1	O	Положительный выход
OUTM1	O	Отрицательный выход
BIAS	O	Выход смещения 1,5 В

25.3.1 Вход микрофона

Модуль поддерживает один не дифференциальный вход микрофона. Микрофонный вход выбирается установкой поля AINSEL в состояние 01 или 10.

В случае установки поля AINSEL в состояние 01 выбирается режим с внутренним формированием смещения в 1,5 В. Для уменьшения шума в этом режиме необходимо подключить микрофонный вход, как показано на рисунке (Рисунок 25-3 а).

Запись значение 10 в поле AINSEL переводит аудиокодек в псевдо-дифференциальный режим. В этом режиме недифференциальный вход объединяется с микрофонным входом и выходом напряжения смещения BIAS (Рисунок 25-3 б). Для уменьшения помех выводы MICIN и INM1 должны иметь разводку одинаковой длины.

Недифференциальный вход преобразуется внутри в дифференциальный перед преобразованием для лучшей защиты от шумов. Для улучшения динамических характеристик модуль поддерживает настраиваемый предусилитель. Управление предусилителем осуществляется при помощи поля INBG регистра ADCCTL.

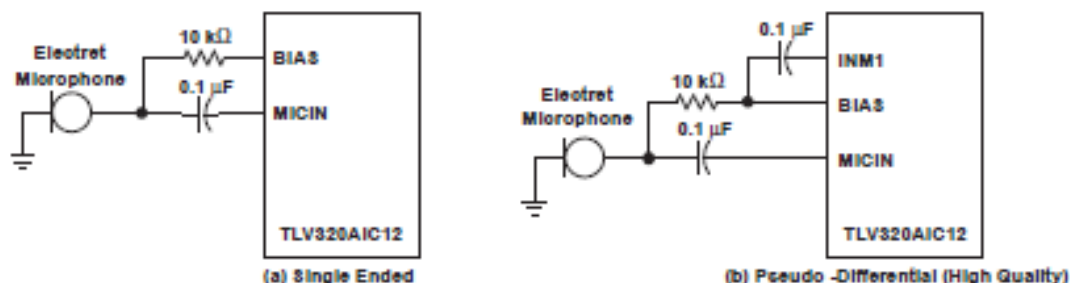


Рисунок 25-3 – Подключение MICIN

25.3.2 Входы INP и INM

Для уменьшения влияния шумов, в модуле реализована возможность дифференциального входа аналогового сигнала. Для этого используются выводы INM1/2 и INP1/2. Источник сигнала, приходящего на аналоговые выводы (INP1/2 и INM1/2), должен иметь низкий импеданс для наилучшего подавления шумов. Для получения максимального динамического диапазона, сигнал должен быть подключен ко входному терминалу как показано на рисунке ниже.

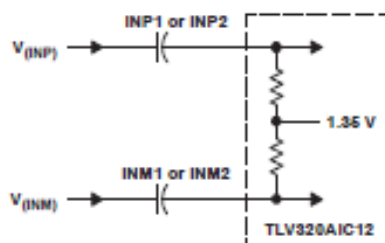


Рисунок 25-4 – Внутренняя схема смещения INP и INM

Недифференциальный вход

Каждый из дифференциальных входов (INP1/2 и INM1/2) может быть подключен как недифференциальный. Для этого вывод INP соответствующего входа необходимо подключить к источнику сигнала, а вывод INM к земле (Рисунок 25-5).

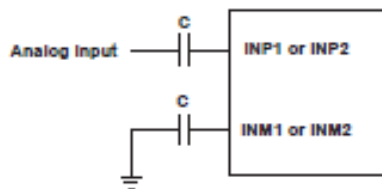


Рисунок 25-5 – Недифференциальный вход

25.3.3 Аналоговые выходы

Выходы OUTP и OUTM дифференциальные выходы канала ЦАП. Выходы OUTP и OUTM могут обеспечить прямую нагрузку 600 Ом и могут быть использованы как не дифференциальные выходы (Рисунок 25-6).

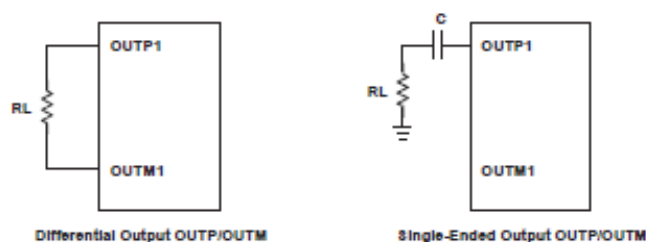


Рисунок 25-6 – Подключение выходов OUTP1/OUTM1

25.4 Работа с модулем

25.4.1 Воспроизведение звука (ЦАП)

Для воспроизведения потока данных необходимо:

- 1 Подать синхросигнал на подсистему DSP (см. раздел 12 «Сигналы тактовой частоты»).
- 2 Снять сброс с периферии DSP (см. раздел 18 «Организация управления подсистемой DSP»).
- 3 Подать синхросигнал на модуль Аудиокодек (см. раздел 12 «Сигналы тактовой частоты»).
- 4 Записать в FIFO DAC 16 отсчетов (регистр DACREG). Каждый отсчет представляет собой 16-битные данные в дополнительном коде (от -32768 до +32767). В случае, если на момент начала работы модуля оставить FIFO пустым, возможны сбои при дальнейшей работе.
- 5 Настроить DAC при помощи регистра DACCTL. Обязательным является включение аналоговой части ЦАП (ODDAC), выходов OUTP1/OUTM1 (ODAMP), всей схемы ЦАП (DACRES).
- 6 Включить DAC и выбрать фильтр (регистр POWCTL). При этом цифровая и аналоговая обратные связи должны быть обнулены.
- 7 По запросу прерывания или по флагу регистра IRQFLAG записывать в FIFO DAC (регистр DACREG) новые отсчеты. В том случае, если новые отсчеты не записаны в регистр, выход DAC стабилизируется в состоянии, соответствующем последнему записанному отсчету, и бит DACVF регистра IRQFLAG будет взведен.

25.4.2 Оцифровка входного потока (АЦП)

Для оцифровки входного потока данных необходимо:

- 1 Подать синхросигнал на подсистему DSP (см. раздел 12 «Сигналы тактовой частоты»).
- 2 Снять сброс с периферии DSP (см. раздел 18 «Организация управления подсистемой DSP»).
- 3 Подать синхросигнал на модуль Аудиокодек (см. раздел 12 «Сигналы тактовой частоты»).
- 4 Настроить ADC при помощи регистра ADCCTL. В регистре задаются: бит включения АЦП, выбор источника для АЦП (см. подраздел 25.3 «Аналоговые площадки ввода-вывода»), уровень записи и уровень входного предусилителя.
- 5 Включить ADC и выбрать фильтр (регистр POWCTL). При этом цифровая и аналоговая обратные связи должны быть обнулены.
- 6 По запросу прерывания или по флагу регистра IRQFLAG считывать из FIFO ADC (регистр ADCREG) новые отсчеты. Отсчеты представляют собой 16-битные данные в дополнительном коде. Полной шкале сигнала для АЦП $U_i = 0$ дБ будет соответствовать появление отчетов с кодами от -22 000 до +22 000 для диф. входов INP1/INM1 и INP2/INM2 и от -25 000 до +25 000 для входа MICIN. В том случае, если новые отсчеты не считаны, возможна потеря отсчетов и бит ADCVF регистра IRQFLAG будет взведен.

25.4.3 Запросы DMA

Запросы DMA от модуля выведены на 22 канал (ЦАП) и на 23 канал (АЦП) DMA контроллера RISC, на 7 канал DMA (ЦАП) и на 6 канал DMA (АЦП) контроллера DSP (см. раздел 24 «Контроллер DMA (DSP)» и раздел 17 «Контроллер прямого доступа в память MDR_DMA»).

Запросы DMA ЦАП возникает в случае опустошения в FIFO ЦАП как минимум четырех отсчетов.

Запросы DMA АЦП возникают, если в FIFO АЦП имеется четыре или более отсчетов.

25.4.4 Прерывания модуля

Прерывания от модуля аудиокодек могут быть обработаны как при помощи процессора RISC, так и при помощи процессора DSP (см. раздел 36 «Прерывания и исключения RISC» и подраздел 19.7 «Прерывания DSP»). Вектора прерываний ЦАП и АЦП различны для обоих ядер.

Прерывание ЦАП возводится по любому из нижеперечисленных событий:

- Короткое замыкания выхода ЦАП (В том случае, если прерывание незамаскировано в регистре MASKCTL и разрешена схема детектирования короткого замыкания в регистре DACCTL).
- Случилось опустошение FIFO DAC (В том случае, если прерывание незамаскировано в регистре MASKCTL).
- Есть хотябы один свободный отсчет FIFO DAC (В том случае, если прерывание незамаскировано в регистре MASKCTL).

Прерывание АЦП возводится по любому из нижеперечисленных событий:

- Случилось переполнение FIFO ADC (В том случае, если прерывание незамаскировано в регистре MASKCTL).
- Есть хотя бы одно оцифрованное значение FIFO ADC (В том случае, если прерывание незамаскировано в регистре MASKCTL).

25.5 Регистры управления

Таблица 25-3 – Регистры управления

Адрес RISC	Адрес DSP	Название	Начальное значение	Описание
0x3000_00A0	0x0050	POWCTL	0x000001C0	Общее управление кодеком
0x3000_00A4	0x0052	ADCCTL	0x0000002A	Управление АЦП
0x3000_00A8	0x0054	DACCTL	0x0000002A	Управление ЦАП
0x3000_00AC	0x0056	MASKCTL	0x0000000F	Маска вектора прерываний
0x3000_00B0	0x0058	IRQFLAG	0x00000000	Флаги прерываний
0x3000_00B4	0x005A	ADCREG	0x00000000	АЦП FIFO регистр
0x3000_00B8	0x005C	DACREG	0x00000000	ЦАП FIFO регистр

25.5.1 Регистр общего управления кодеком

Таблица 25-4 – Описание бит регистра общего управления кодеком

Бит	Поле	Нач. значение	Описание
0	ADCEN	0	Включение канала АЦП: 0 – канал выключен; 1 – канал включен
1	DACEN	0	Включение канала ЦАП: 0 – канал выключен; 1 – канал включен
2	IIREN	0	Включение фильтра БИХ: 0 – фильтр выключен; 1 – фильтр включен

25.5.2 Регистр управления АЦП

Таблица 25-5 – Описание бит регистра управления АЦП

Бит	Поле	Нач. значение	Описание
5:0	ADGAIN	101010	Уровень записи (шаг – 1 дБ): 111111 – Mute; 111110 – +20 дБ; 111101 – +19 дБ; ... 101010 – 0 дБ; 101001 – -1 дБ; ... 000000 – -42 дБ

Бит	Поле	Нач. значение	Описание
7:6	INBG	00	Входной предусилитель: 00 – 0 дБ; 01 – 12 дБ; 10 – 6 дБ; 11 – 24 дБ
9:8	AINSEL	00	Выбор источника для аналогового входа: 00 – сигнал на INP1/INM1; 01 – сигнал на MICIN с внутренним формированием смещения 1,5 В; 10 – сигнал на MICIN с внешним формированием смещения; 11 – сигнал на INP2/INM2
11	-	-	Зарезервировано
10	ICONT	0	Управление аналоговым АЦП и фильтром: 0 – выключен; 1 – включен

25.5.3 Регистр управления ЦАП

Таблица 25-6 – Описание бит регистра управления ЦАП

Бит	Поле	Нач. значение	Описание
5:0	DAGAIN	101010	Уровень воспроизведения (шаг – 1 дБ): 111111 – Mute; 111110 – +20 дБ; 111101 – +19 дБ; ... 101010 – 0 дБ; 101001 – -1 дБ; ... 000000 – -42 дБ
6	MUTE1	0	Mute на выходах OUTP1/OUTM1: 0 – выходы OUTP1/OUTM1 работает в нормальном режиме; 1 – выходы OUTP1/OUTM1 обнулены
7	ODAMP	0	Управление аналоговым выходным усилителем OUTP1/OUTM1: 0 – OUTP1/OUTM1 – OFF; 1 – OUTP1/OUTM1 – ON
8	ODDAC	0	Управление схемой смещения BIAS: 0 – Схема смещения BIAS выключена; 1 – Схема смещения BIAS включена
9	ODDAC	0	Управление аналоговым DAC и фильтрами: 0 – аналоговая часть DAC выключена; 1 – аналоговая часть DAC включена

Бит	Поле	Нач. значение	Описание
10	OVECBA	0	Разрешение схемы детектирования короткого замыкания выходного буфера: 1 – разрешено; 0 – запрещено
11	OVECBS	0	Разрешение схемы детектирования короткого замыкания схемы смещения: 1 – разрешено; 0 – запрещено
14:12	SIDETONE	111	Цифровая обратная петля: 111 – Mute; 110 – -21 дБ; 110 – -18 дБ; 110 – -15 дБ; 110 – -12 дБ; 110 – -9 дБ; 110 – -6 дБ; 000 – -3 дБ
15	DACRES	1	Сброс ЦАП (аналоговая часть): 0 – ЦАП находится в сбросе; 1 – ЦАП находится в рабочем режиме

25.5.3.1 Маска вектора прерываний

Таблица 25-7 – Описание бит маски вектора прерываний

Бит	Поле	Нач. значение	Описание
0	DAOVFM	1	Маска прерывания по событию чтения со стороны аудиокодека пустого значения из FIFO DAC (FIFO пусто и произошло чтение): 0 – запретить прерывание; 1 – разрешить прерывание
1	ADCVFM	1	Маска прерывания по событию записи в переполненный FIFO ADC со стороны ЦПУ: 0 – запретить прерывание; 1 – разрешить прерывание
2	ADCNSM	1	Маска прерывания ADC FIFO: 0 – запретить прерывание; 1 – разрешить прерывание
3	DACNSM	1	Маска прерывания DAC FIFO: 0 – запретить прерывание; 1 – разрешить прерывание
4	OVERCURM	1	Маска прерывания короткого замыкания выхода ЦАП: 0 – запретить прерывание; 1 – разрешить прерывание

25.5.4 Флаги прерываний

Таблица 25-8 – Описание бит флагов прерываний

Бит	Поле	Нач. значение	Описание
0	DAOVF	0	Прерывания по событию чтения со стороны аудиокодека пустого значения из FIFO DAC (FIFO пусто и произошло чтение): 0 – DAC не переполнен; 1 – Произошло переполнение DAC. Запись «1» сбрасывает прерывание
1	ADCVF	0	Прерывания по событию записи в переполненный FIFO ADC со стороны ЦПУ: 0 – ADC не переполнен; 1 – Произошло переполнение ADC. Запись «1» сбрасывает прерывание
2	ADCNS	0	Прерывание ADC FIFO: 0 – ADC FIFO пусто; 1 – в ADC FIFO имеется хотя бы один отсчет для чтения
3	DACNS	0	Прерывание DAC FIFO: 0 – DAC FIFO пусто; 1 – в DAC FIFO имеется хотя бы одно свободное место для записи отсчета
4	OVERCUR	0	Прерывания короткого замыкания выхода ЦАП: 0 – нет прерывания КЗ ЦАП; 1 – есть прерывание КЗ ЦАП

25.5.4.1 АЦП FIFO регистр

Таблица 25-9 – Описание бит регистра АЦП FIFO

Бит	Поле	Нач. значение	Описание
15:0	ADCSMP	00	Новый отчет тракта АЦП

25.5.4.2 ЦАП FIFO регистр

Таблица 25-10 – Описание бит регистра ЦАП FIFO

Бит	Поле	Нач. значение	Описание
15:0	DACSMP	00	Новый отчет тракта ЦАП

26 Контроллер блока шифрования ГОСТ 28147-89 (DSP)

Блок шифрования предназначен для аппаратной поддержки программ криптографической защиты информации в соответствии с ГОСТ 28147-89.

Со стороны ядра микросхемы блок шифрования представляет собой набор программно доступных 16-разрядных регистров (Таблица 26-1).

26.1 Описание регистров

Таблица 26-1 – Описание регистров блока шифрования

Адрес RISC	Адрес DSP	Название регистра	Описание
0x3000_0080	0x0040	CRPT_CWR	Регистр управления
0x3000_0084	0x0042	CRPT_SR	Регистр состояния
0x3000_0088	0x0044	CRPT_DATA	Регистр данных
0x3000_008C	0x0046	CRPT_KR	Регистр ключа
0x3000_0090	0x0048	CRPT_CR	Регистр констант замены
0x3000_0094	0x004A	CRPT_SYNR	Регистр синхропосылки
0x3000_0098	0x004C	CRPT_IMIT	Регистр имитовставки
0x3000_009C	0x004E	CRPT_ITER	Регистр числа итераций

Регистры данных, ключа, синхропосылки и констант замены являются 16-разрядными портами, через который осуществляется доступ к соответствующим массивам данных (8 байт), ключа (32 байта), синхропосылки (8 байт), констант замены (64 байт) и имитовставки (8 байт).

Блок шифрования содержит два практически одинаковых модуля, один из которых реализует один из трех основных режимов работы устройства, а другой предназначен для аппаратной поддержки выработки имитовставки. Это необходимо для того, чтобы данные проходили через устройство лишь однажды, без повторного прохождения через модуль для выработки имитовставки.

Работа блока шифрования возможна в режиме опроса или в режиме прерывания. В режиме опроса необходимо постоянно опрашивать состояние бита готовности в регистре состояния. Если бит установлен, данные можно считывать. В режиме прерывания процедура обработки прерывания, которая активизируется по сигналу INT устройства, может сразу считывать данные, не опрашивая регистр состояния. Все сказанное относится ко всем режимам работы устройства.

Флаг запроса прерывания CRPTIF (выход INT устройства) устанавливается после окончания преобразования данных при условии, что биты IE_CRPT (регистра CRPT_CWR) и CRPTIE (регистра PIE1) установлены. Запрос прерывания снимается либо при сбросе устройства (программном или аппаратном), либо при старте (установке бита «START» регистра управления), либо по факту чтения данных.

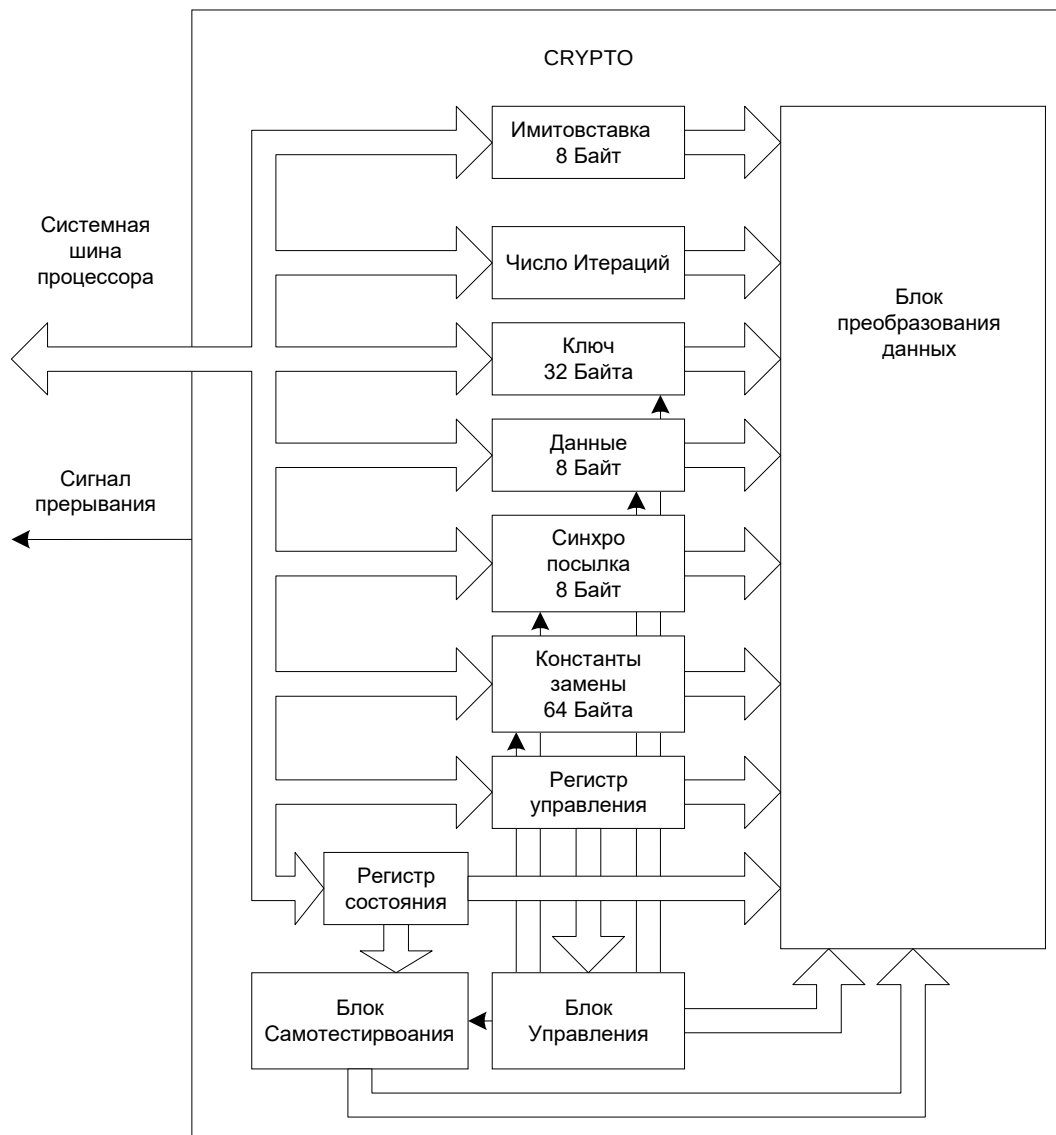


Рисунок 26-1 – Блок-схема блока аппаратной поддержки алгоритма шифрования по ГОСТ 28147-89

Выработка имитовставки возможна в любом из описанных выше режимов работы. Для корректной работы в режимах с выработкой имитовставки необходимо перед загрузкой ключа, данных и т.д. сбросить устройство (установить бит «RST» в регистре управления) для синхронизации с началом пакета обрабатываемых данных. Таким образом, процесс программирования в режимах с выработкой имитовставки дополняется еще одним, самым первым действием – сбросом устройства. В остальном отличий нет. После окончания преобразования данных имитовставка считывается из порта имитовставки, из нее программным путем вырезается отрезок нужной длины и посылается вслед пакету преобразованных данных.

В режимах дешифрации данных необходимо после обсчета последнего пакета данных дать еще один старт преобразования для завершения расчета имитовставки. При этом требуется установить бит IM в регистре CRPT_CWR, а новые данные посылать не требуется.

26.1.1 Регистр управления CRPT_CWR

Таблица 26-2 – Регистр управления CRPT_CWR

Номер	7	6	5	4	3	2	1	0
Доступ	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	BIST	RST	IE_CRPT	DIR	START	IM	MODE1_CRPT	MODE0_CRPT
Обозначения здесь и далее по тексту: R – бит для чтения; W – бит с возможностью записи; U – бит не реализован, читается как 0; -п – значение бита после сброса по включению питания: 1 – установлен; 0 – сброшен; x – значение не известно								

Таблица 26-3 – Описание бит регистра управления CRPT_CWR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
7	BIST	Тестовый режим работы
6	RST	Сброс – предназначен для приведения устройства в исходное состояние. Примечание – Бит «RST» сбрасывается автоматически в следующем цикле системной тактовой частоты
5	IE_CRPT	Разрешение прерывания. Используется для разрешения («1») или запрета («0») выдачи сигнала прерывания
4	DIR	Направление шифрации: 0 – шифрация; 1 – дешифрация
3	START	Старт. Предназначен для запуска процесса шифрации/дешифрации данных. Примечание – Бит «START» сбрасывается автоматически после начала преобразования
2	IM	Выработка имитовставки в режимах дешифрации данных. Бит «IM» устанавливается после окончания обработки данных для завершения выработки имитовставки. После установки необходимо выполнить еще один цикл обработки (не посылая данных), после чего имитовставка может быть прочитана из устройства
1, 0	MODE1_CRPT – MODE0_CRPT	Режим работы блока: 00 – работа в режиме простой замены; 01 – работа в режиме гаммирования; 10 – работа в режиме гаммирования с обратной связью; 11 – инициализация синхропосылки. Примечание – В режимах «01» и «10» перед началом обработки данных следует записать синхропосылку, установить режим «11» и послать команду «старт». Далее необходимо установить нужный режим и обработать данные

26.1.2 Регистр состояния CRPT_SR

Таблица 26-4 – Регистр состояния CRPT_SR

Номер	7	6	5	4	3	2	1	0
Доступ	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
	DNC_BIST	KW*	SW*	DR*	DW*	KF*	ERROR_CRPT	READY_CRPT

Таблица 26-5 – Описание бит регистра состояния CRPT_SR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
7	DNC_BIST	Результат динамического контроля/самотестирования
6	KW*	Состояние указателя чтения памяти имитоданных
5	SW*	Состояние указателя записи памяти синхроросылки
4	DR*	Состояние указателя чтения памяти данных
3	DW*	Состояние указателя записи памяти данных
2	KF*	Состояние указателя записи памяти ключа
1	ERROR_CRPT	Ошибка. Бит определяет наличие или отсутствие ошибки в работе устройства. Состояние ошибки возникает в случае попытки запустить процесс преобразования данных при не полностью записанных данных или ключе
0	READY_CRPT	Работа/готовность. Бит определяет готовность данных после выполненного преобразования (шифрации или дешифрации). После начала преобразования данных бит автоматически сбрасывается и вновь устанавливается после завершения преобразования
* Биты 6...2 зарезервированы для системных целей. По их состоянию программист при необходимости может определить положение указателей записи и чтения (состояние битов и интерпретация будут определены в дальнейшем, так как для работы они не являются необходимыми)		

26.1.3 Регистр данных шифрации CRPT_DATA

Таблица 26-6 – Регистр данных шифрации CRPT_DATA

Номер	15...0
Доступ	R/W-0
	DATA[15:0]

Таблица 26-7 – Описание бит регистра данных шифрации CRPT_DATA

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15...0	DATA[15:0]	16-разрядный порт данных, предназначенный для ввода исходных или вывода преобразованных данных. При этом адрес для всех байтов данных в программе указывается один и тот же. Запись всех байтов данных в модуль осуществляется последовательной записью 16-разрядных слов по адресу

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
		регистра данных, начиная с младшего байта (или первого байта текста, если данные рассматривать как текст). Чтение осуществляется аналогично. Реальная адресация во внутренней памяти модуля осуществляется внутренними указателями, автоматически инкрементирующимися или декрементирующимися в зависимости от направления передачи данных. Отметим, что указатели записи и чтения сбрасываются при программном сбросе устройства, то есть при установке бита «RST»

26.1.4 Регистр ключа шифрации CRPT_KR

Таблица 26-8 – Регистр ключа шифрации CRPT_KR

Номер	15...0
Доступ	R/W-0
	KEY[15:0]

Таблица 26-9 – Описание бит регистра ключа шифрации CRPT_KR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15...0	KEY[15:0]	16-разрядный порт ключа, предназначенный для ввода исходного ключа. При этом адрес для всех байтов ключа в программе указывается одни и тот же. Запись всех байтов ключа в модуль осуществляется последовательной записью 16-разрядных слов по адресу регистра ключа, начиная с младшего байта

26.1.5 Регистр синхропосылки CRPT_SYNR

Таблица 26-10 – Регистр синхропосылки CRPT_SYNR

Номер	15...0
Доступ	R/W-0
	SYNCH[15:0]

Таблица 26-11 – Описание бит регистра синхропосылки CRPT_SYNR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15...0	SYNCH[15:0]	16-разрядный порт синхропосылки, предназначенный для ввода или вывода синхропосылки. При этом адрес для всех байтов синхропосылки в программе указывается один и тот же. Запись всех байтов данных в модуль осуществляется последовательной записью 16-разрядных слов по адресу регистра синхропосылки, начиная с младшего байта. Чтение осуществляется аналогично

26.1.6 Регистр констант замены CRPT_CR

Таблица 26-12 – Регистр констант замены CRPT_CR

Номер	15...0
Доступ	R/W-0
	CONST[15:0]

Таблица 26-13 – Описание бит регистра констант замены CRPT_CR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15...0	CONST[15:0]	16-ти разрядный порт констант замены, предназначенный для ввода констант замены. При этом адрес для всех байтов констант в программе указывается один и тот же. Запись всех байтов данных в модуль осуществляется последовательной записью байтов по адресу регистра данных, начиная с младшего байта. Ввод констант замены осуществляется по 16 разрядов, те есть вводятся сразу четыре константы. Заполнение узлов замены осуществляется, начиная с первого узла и заканчивая восьмым узлом

26.1.7 Регистр имитовставки CRPT_IMIT

Таблица 26-14 – Регистр имитовставки CRPT_IMIT

Номер	15...0
Доступ	R/W-0
	IMIT[15:0]

Таблица 26-15 – Описание бит регистра имитовставки CRPT_IMIT

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15...0	IMIT[15:0]	16-разрядный порт имитовставки, предназначенный для вывода имитовставки после процесса шифрации и дешифрации. При этом адрес для всех байтов имитовставки в программе указывается один и тот же. Чтение всех байтов данных из модуля осуществляется последовательно по адресу регистра имитовставки, начиная с младшего байта. Выработка имитовставки осуществляется автоматически параллельно с шифрацией или дешифрацией данных. После окончания шифрования имитовставка считывается из регистра и далее обрабатывается программно в соответствие с ГОСТ 28147-89

26.1.8 Регистр числа итераций CRPT_ITER

Таблица 26-16 – Регистр числа итераций CRPT_ITER

Номер	7	6	5	4	3	2	1	0
Доступ	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	-	EN_CRPT	ITER5	ITER4	ITER3	ITER2	ITER1	ITER0

Таблица 26-17 – Описание бит регистра числа итераций CRPT_ITER

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
7	-	Зарезервировано.
6	EN_CRPT	Бит разрешения работы блока: 0 – блок аппаратной поддержки алгоритма кодирования выключен; 1 – блок включен
5...0	ITER[5:0]	Регистр счетчика итераций циклов шифрации/дешифрации
Примечание – Бит EN_CRPT меняется только в том случае, если бит DNC_BIST регистра CRPT_CWR сброшен		

26.2 Работа блока по шифрованию и дешифрованию данных

26.2.1 Шифрование данных. Режим простой замены

Порядок выполняемых действий:

- 1 Выполнить программный сброс блока для синхронизации имитовставки (установить бит «RST» в регистре управления);
- 2 Установить режим «00» в регистре управления и бит направления шифрования;
- 3 Ввести ключ;
- 4 Ввести константы замены;
- 5 Ввести очередной блок данных;
- 6 Установить бит «START» в регистре управления;
- 7 Подождать, пока преобразование закончится (либо по прерыванию, либо по опросу бита готовности);
- 8 Прочитать преобразованные данные и, если не все пакеты обработаны перейти к п.5, или, в противном случае, перейти к п.9.;
- 9 Прочитать имитовставку из порта имитовставки.

Примечание – Пункты 1 и 2 (программный сброс и установка режимов) можно выполнять одновременно (одной командой).

26.2.2 Дешифрование данных. Режим простой замены

Порядок выполняемых действий:

- 1 Выполнить программный сброс устройства для синхронизации имитовставки (установить бит «RST» в регистре управления);
- 2 Ввести ключ;
- 3 Ввести константы замены;
- 4 Установить режим «00» в регистре управления и бит направления шифрования;
- 5 Ввести очередной блок данных;
- 6 Установить бит «START» в регистре управления;
- 7 Подождать, пока преобразование закончится (либо по прерыванию, либо по опросу бита готовности);
- 8 Прочитать преобразованные данные и, если не все пакеты обработаны перейти к п.5, или, в противном случае, перейти к п.9;
- 9 Одновременно установить биты «IM» и «START» в регистре управления для завершения выработки имитовставки (их запись в регистр должна проводиться одной командой);
- 10 Подождать, пока преобразование закончится (либо по прерыванию, либо по опросу бита готовности);
- 11 Прочитать имитовставку из порта имитовставки.

26.2.3 Шифрование данных. Режим гаммирования

Порядок выполняемых действий:

- 1 Выполнить программный сброс устройства для синхронизации имитовставки (установить бит «RST» в регистре управления);
- 2 Ввести ключ;
- 3 Ввести константы замены;
- 4 Установить режим «11» в регистре управления для инициализации синхропосылки и бит направления шифрования;
- 5 Ввести синхропосылку;
- 6 Установить бит «START» в регистре управления;
- 7 Подождать, пока преобразование закончится (либо по прерыванию, либо по опросу бита готовности);
- 8 Установить режим гаммирования «01»;
- 9 Ввести очередной блок данных;
- 10 Установить бит «START» в регистре управления;
- 11 Подождать, пока преобразование закончится (либо по прерыванию, либо по опросу бита готовности);
- 12 Прочитать преобразованные данные и, если не все пакеты обработаны перейти к п.9, или, в противном случае перейти к п.13;
- 13 Прочитать имитовставку из порта имитовставки.

26.2.4 Дешифрование данных. Режим гаммирования

Порядок выполняемых действий:

- 1 Выполнить программный сброс устройства для синхронизации имитовставки (установить бит «RST» в регистре управления);
- 2 Ввести ключ;
- 3 Ввести константы замены;
- 4 Установить режим «11» в регистре управления для инициализации синхропосылки и бит направления шифрования;
- 5 Ввести синхропосылку;
- 6 Установить бит «START» в регистре управления;
- 7 Подождать, пока преобразование закончится (либо по прерыванию, либо по опросу бита готовности);
- 8 Установить режим гаммирования «01»;
- 9 Ввести очередной блок данных;
- 10 Установить бит «START» в регистре управления;
- 11 Подождать, пока преобразование закончится (либо по прерыванию, либо по опросу бита готовности);
- 12 Прочитать преобразованные данные и, если не все пакеты обработаны перейти к п.9, или, в противном случае перейти к п.13;
- 13 Одновременно установить биты «START» и «IM» в регистре управления для завершения выработки имитовставки;
- 14 Подождать, пока преобразование закончится (либо по прерыванию, либо по опросу бита готовности);
- 15 Прочитать имитовставку из порта имитовставки.

26.2.5 Шифрование данных. Режим гаммирования с обратной связью

Порядок выполняемых действий:

- 1 Выполнить программный сброс устройства для синхронизации имитовставки (установить бит «RST» в регистре управления);
- 2 Ввести ключ;
- 3 Ввести константы замены;
- 4 Установить режим гаммирования с обратной связью «10» в регистре управления и бит направления шифрования;
- 5 Ввести синхропосылку;
- 6 Ввести очередной блок данных;
- 7 Установить бит «START» в регистре управления;
- 8 Подождать, пока преобразование закончится (либо по прерыванию, либо по опросу бита готовности);
- 9 Прочитать преобразованные данные и, если не все пакеты обработаны перейти к п.6, или, в противном случае перейти к п.10;
- 10 Прочитать имитовставку из порта имитовставки.

26.2.6 Дешифрование данных. Режим гаммирования с обратной связью

Порядок выполняемых действий :

- 1 Выполнить программный сброс устройства для синхронизации имитовставки (установить бит «RST» в регистре управления);
- 2 Ввести ключ;
- 3 Ввести константы замены;
- 4 Установить режим гаммирования с обратной связью «10» в регистре управления и бит направления шифрования;
- 5 Ввести синхропосылку;
- 6 Ввести очередной блок данных;
- 7 Установить бит «START» в регистре управления;
- 8 Подождать, пока преобразование закончится (либо по прерыванию, либо по опросу бита готовности);
- 9 Прочитать преобразованные данные и, если не все пакеты обработаны перейти к п.6, или, в противном случае перейти к п.10;
- 10 Одновременно установить биты «START» и «IM» в регистре управления для завершения выработки имитовставки;
- 11 Подождать, пока преобразование закончится (либо по прерыванию, либо по опросу бита готовности);
- 12 Прочитать имитовставку из порта имитовставки.

26.2.7 Режим самопроверки

Самопроверка устройства может осуществляться между преобразованиями пакетов данных. Для ее осуществления надо записать в регистр управления управляющее слово «11100000», если завершение самотестирования определяется по прерыванию, или «11000000», если завершение определяется по опросу бита готовности. Результат самотестирования определяется состоянием бита «DNC_BIST» регистра контроля устройства. Если он установлен в «1», то устройство исправно, в противном случае аппаратура работает с ошибками.

Динамический контроль данных в процессе вычислений осуществляется автоматически. После окончания процесса шифрования бит «DNC_BIST» содержит результат динамического контроля процесса шифрования. Если бит установлен в «1», то результат динамического контроля положительный, в противном случае во время работы схемы произошла ошибка.

26.2.8 Порядок занесения констант замены и ключа в соответствии с ГОСТ Р 34.11-94

В приложении А ГОСТ Р 34.11-94 приведена таблица констант замены. Последовательность занесения данных из этой таблицы в регистр CRPT_CR содержит Таблица 26-19.

В приложении А ГОСТ Р 34.11-94 на странице 7 приведен ключ замены:

K1 = 733D2C20 65686573 74746769 79676120
626E7373 20657369 326C6568 33206D54

Порядок занесения этого ключа в регистр CRPT_KR содержит Таблица 26-20.

Таблица 26-18 – Константы замены

	8	7	6	5	4	3	2	1
0	1	D	4	6	7	5	E	4
1	F	B	B	C	D	6	B	A
2	D	4	A	7	A	1	4	9
3	0	1	0	1	1	D	C	2
4	5	3	7	5	0	A	6	D
5	7	F	2	F	8	3	D	8
6	A	5	1	D	9	4	F	0
7	4	9	D	8	F	2	A	E
8	9	0	3	4	E	E	2	6
9	2	A	6	A	4	F	3	B
10	3	E	8	9	6	C	8	1
11	E	7	5	E	C	7	1	C
12	6	6	9	0	B	6	0	7
13	B	8	C	3	2	0	7	F
14	8	2	F	B	5	9	5	5
15	C	C	E	2	3	B	9	3

Таблица 26-19 – Последовательность занесения данных в регистр CRPT_CR

Номер	Константа	Номер	Константа	Номер	Константа	Номер	Константа
1	0xA4	17	0x85	33	0xC6	49	0xBD
2	0x29	18	0xD1	34	0x17	50	0x14
3	0x8D	19	0x3A	35	0xF5	51	0xF3
4	0xE0	20	0x24	36	0x8D	52	0x95
5	0xB6	21	0xFE	37	0xA4	53	0xA0
6	0xC1	22	0x7C	38	0xE9	54	0x7E
7	0xF7	23	0x06	39	0x30	55	0x86
8	0x35	24	0xB9	40	0x2B	56	0xC2
9	0xBE	25	0xD7	41	0xB4	57	0xF1
10	0xC4	26	0x1A	42	0x0A	58	0x0D
11	0xD6	27	0x80	43	0x27	59	0x75
12	0xAF	28	0xF9	44	0xD1	60	0x4A
13	0x32	29	0x4E	45	0x63	61	0x29
14	0x18	30	0xC6	46	0x58	62	0xE3
15	0x70	31	0x2B	47	0xC9	63	0xB6
16	0x95	32	0x35	48	0xEF	64	0xC8

Таблица 26-20 – Порядок занесения ключа в регистр CRPT_KR

Номер	Константа	Номер	Константа	Номер	Константа	Номер	Константа
1	0x54	9	0x69	17	0x20	25	0x73
2	0x6D	10	0x73	18	0x61	26	0x65
3	0x20	11	0x65	19	0x67	27	0x68
4	0x33	12	0x20	20	0x79	28	0x65
5	0x68	13	0x73	21	0x69	29	0x20
6	0x65	14	0x73	22	0x67	30	0x2C
7	0x6C	15	0x6E	23	0x74	31	0x3D
8	0x32	16	0x62	24	0x74	32	0x73

27 Контроллер интерфейса MDR_USB

Контролер USB реализует функции контроллера функционального устройства (Device) и управляющего устройства (Host) в соответствии со спецификацией USB 1.0.

Контроллер USB поддерживает следующие возможности:

- режимы работы Full Speed (12 Мбит/с) и Low Speed (1,5 Мбит/с);
- контроль ошибок с помощью циклического избыточного кода (CRC);
- NRZI код приема/передачи;
- управляющая (Control);
- сплошная (Bulk);
- изохронная (Isochronous) передачи и передача по прерываниям (Interrupt);
- конфигурирование USB Device от одной до четырех оконечных точек. Каждая

оконечная точка USB Device имеет собственную память FIFO размером 64 байта. USB Host поддерживает до 16 оконечных точек. Возможности USB Host: FIFO размером 64 байта; автоматическая отправка SOF пакетов; вычисление оставшегося во фрейме времени.

27.1 Инициализация контроллера при включении

При включении питания в первую очередь должны быть заданы параметры тактового сигнала блока USB. Источником тактового сигнала для блока USB может быть внешний генератор HSE. Блок USB функционирует на частоте 48 МГц. Требуемая частота может быть получена умножением частоты генератора до требуемого значения. Умножение выполняется встроенным блоком PLL_USB.

Блок умножения позволяет провести умножение входной тактовой частоты на коэффициент от 2 до 16, задаваемый в поле PLLUSBMUL регистра PLL_CONTROL. Входная частота блока умножителя должна быть в диапазоне 6...16 МГц, выходная должна составлять 48 МГц. При выходе блока умножителя тактовой частоты в расчетный режим вырабатывается сигнал PLLRDY. Блок включается с помощью сигнала PLLUSBON. Выходная частота используется как основная частота протокольной части интерфейса USB.

Для задания тактовой частоты блока необходимо соблюдать следующий порядок работы. Установить бит разрешения тактирования блока (бит 3 регистра PER_CLOCK). В регистре USB_CLOCK установить бит USBCLKEN, задать источник тактового сигнала в полях USBC1SEL и USBC2SEL. Установить бит PLLUSBON и задать коэффициент умножения в поле PLLUSBMUL регистра PLL_CONTROL, если используется USBPLL.

После подачи тактового сигнала на блок USB необходимо выполнить сброс контроллера. Сброс выполняется установкой бита RESET_CORE в регистре USB_HSCR. Сигнал сброса необходимо удерживать как минимум 10 циклов тактовой частоты. После этого могут быть заданы параметры шины USB (скорость, полярность, наличие подтяжек).

27.2 Задание параметров шины USB и события подключения/отключения

Контроллер USB может быть сконфигурирован как USB Host или как USB Device. Конфигурация задается битом CORE_MODE в регистре HSCR (0 – режим Device, 1 – режим Host). Прием/передача через физический интерфейс USB разрешаются установкой бит EN_RX и EN_TX в этом же регистре. В режиме приема имеется возможность отключить передатчик в целях экономии потребления (EN_TX=0). Отключение всего блока в целом осуществляется при EN_RX=0.

В режиме Device параметры шины задаются в регистре SC. Скорость задается битом SCFSR (0 – 1,5 Мбит/с, 1 – 12 Мбит/с), полярность битом SCFSP (0 – Low speed, 1 – Full speed) этого регистра.

В режиме Host параметры шины задаются в регистре HTXLC. Скорость задается битом FSLR (0 – 1,5 Мбит/с, 1 – 12 Мбит/с), полярность битом FSPL (0 – Low speed, 1 – Full speed) этого регистра.

В режиме Host контроллер автоматически определяет подключение или отключение устройства к шине. Бит CONEV регистра USB_HSI устанавливается в 1 при возникновении одного из событий.

В режиме Host необходимо прямое подключение (без USB HUB) Low speed устройства.

27.3 Задание адреса и инициализация оконечных точек

Функциональный адрес устройства USB задается в регистре SA.

Для инициализации конечной точки в первую очередь необходимо установить бит глобального разрешения всех оконечных точек (SCGEN = 1 в регистре SC). Биты EPEN в регистрах SEP[x].CTRL должны быть установлены, чтобы разрешить соответствующую оконечную точку. Если предполагается использовать изохронный тип передачи оконечной точки, то необходимо установить бит EPISEN в соответствующем регистре SEP[x].CTRL.

27.4 Транзакция IN (USB Device)

Если на шине появляется IN пакет, и адрес совпадает с заданным в регистре SA, то бит SCTDONE регистра SIS устанавливается в 1.

Если оконечная точка не готова (бит EPRDY = 0 в регистре SEP[x].CTRL), то контроллер отправляет NAK пакет (Рисунок 27-1 а). Бит NAKSENT регистра SEP[x].STS устанавливается в 1.

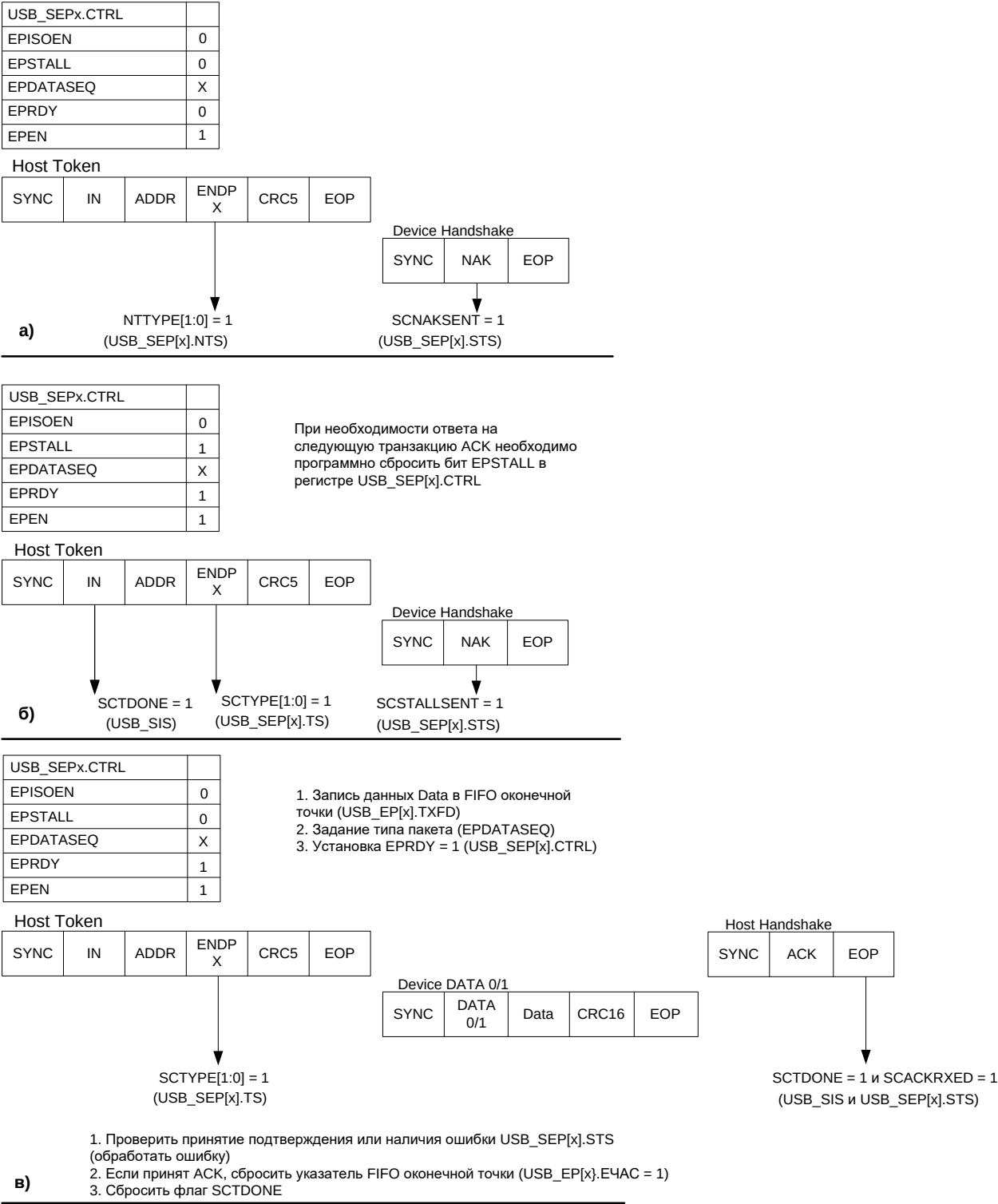


Рисунок 27-1 (а, б, в) – Транзакция IN (USB Device)

а – оконечная точка не готова;
б – установлен бит EPSTALL;
в – оконечная точка готова

Если оконечная точка готова и установлен бит EPSTALL в регистре SEP[x].CTRL, то контроллер отправляет STALL пакет (Рисунок 27-1 б). Бит SCSTALLSENT регистра SEP[x].STS устанавливается в 1.

Если оконечная точка готова (Рисунок 27-1 в), биты SCTTYPE[1:0] в регистре SEP[x].TS устанавливаются в «1» для конечной точки с номером, содержащимся в поле пакета. Контроллер может передавать пакет данных. Пакет данных формируется

записью в регистр EP[x].TXFD побайтно в FIFO оконечной точки. Запись «1» в EP[x].TXFDC сбрасывает указатель FIFO передачи в 0. Максимальный размер передаваемого пакета составляет 64 байт. Попытка записи более 64 байт подряд приведет к переполнению FIFO. Перед началом формирования очередного пакета необходимо выполнять сброс указателя FIFO.

Если в ответ на переданные данные хост отправляет ACK пакет, то бит SCACKRXED в регистре SEP[x].STS устанавливается в «1». Для отправки следующего пакета необходимо инвертировать бит EPDATASEQ в регистре SEP[x].CTRL, чтобы соблюдалась очередность отправки пакетов DATA0, DATA1.

После окончания транзакции бит SCTDONE регистра SIS должен быть очищен записью 1.

27.5 Транзакция SETUP/OUT (USB Device)

Если на шине появляется SETUP/OUT пакет, и адрес совпадает с заданным в регистре USB_SA и оконечная точка готова (бит EP_READY = 1 в регистре ENDPOINTx_CONTROL), то бит SCTDONE регистра SIS устанавливается в «1».

Если оконечная точка не готова (бит EPRDY = 0 в регистре SEP[x].CTRL), то контроллер отправляет NAK пакет (Рисунок 27-2 а). Бит NAKSENT регистра SEP[x].ST устанавливается в «1».

Если оконечная точка готова и установлен бит EPSSTALL в регистре SEP[x].CTRL, то контроллер отправляет STALL пакет (Рисунок 27-2 б). Бит SCSTALLSENT регистра SEP[x].STS устанавливается в «1».

Если оконечная точка готова (Рисунок 27-2 в) и на шине был пакет SETUP, то биты SCTTYPE[1:0] в регистре SEP[x].TS устанавливаются в значение 00 для конечной точки с номером, содержащимся в поле пакета. Если пакет OUT, то значение SCTTYPE[1:0] = 2.

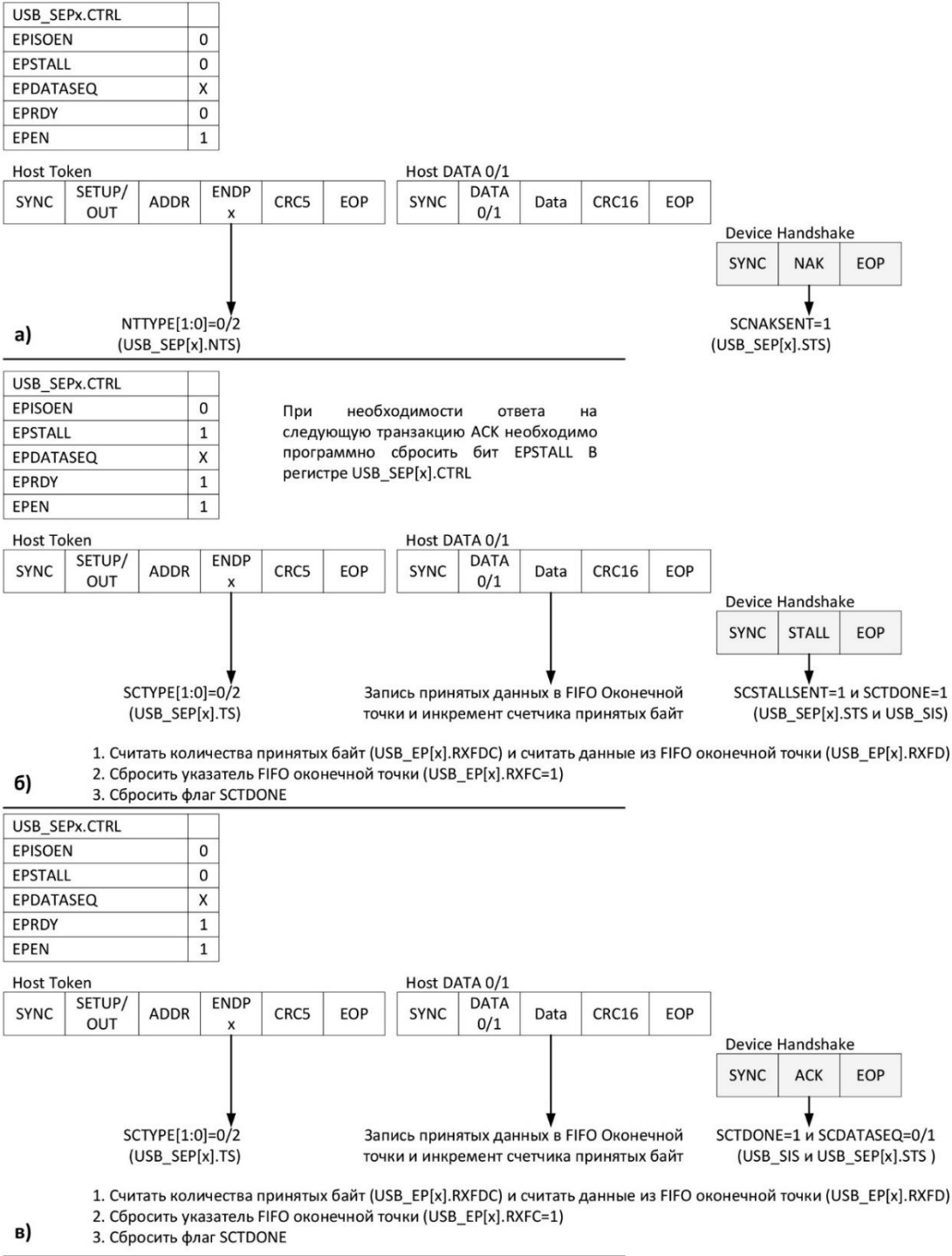


Рисунок 27-2 (а, б, в) – Транзакция SETUP/OUT (USB Device)

- а – оконечная точка не готова;
- б – установлен бит EPSTALL;
- в – оконечная точка готова

Когда на шине появляется DATA0/DATA1 пакет, данные начинают записываться побайтно в FIFO приема соответствующей оконечной точки. После записи каждого байта увеличивается на единицу счетчик принятых байтов. Принятые байты считываются через регистр EP[х].RXFD. Количество принятых байтов содержится в регистре EP[х].RXFDC. После приема очередного пакета необходимо выполнять сброс указателя FIFO приема записью «1» в регистр EP[х].RXFC.

После окончания транзакции бит SCTDONE регистра SIS должен быть очищен записью «1».

27.6 Транзакция SETUP/OUT (USB Host)

Для начала транзакции должны быть заданы адрес устройства (регистр НТХА), оконечная точка (регистр НТХЕ) и тип token пакета (регистр НТХТ). Данные записываются побайтно в регистр НТХFD. Максимальный размер передаваемого пакета составляет 64 байтов. Попытка записи более 64 байтов подряд приведет к переполнению FIFO. Запись «1» в НТХFDC сбрасывает указатель FIFO передачи в 0. Перед началом формирования очередного пакета необходимо выполнять сброс указателя FIFO. Транзакция запускается при установке бита TREQ регистра НТХС. Host отправляет пакет Setup/Out и пакет данных.

После окончания транзакции бит TDONE = 1 (регистр HSI). Этот бит перед началом каждой транзакции должен быть очищен записью «1». PID принятого пакета записывается в регистре HRXP.

Если в ответ получен пакет NAK (Рисунок 27-3 а), то бит NAKRXED = 1 (регистр HRXS).

Если в ответ получен пакет STALL (Рисунок 27-3 б), то бит STALLRXED = 1 (регистр HRXS).

Если в ответ получен пакет ACK (Рисунок 27-3 в), то бит ACKRXED = 1 (регистр HRXS).

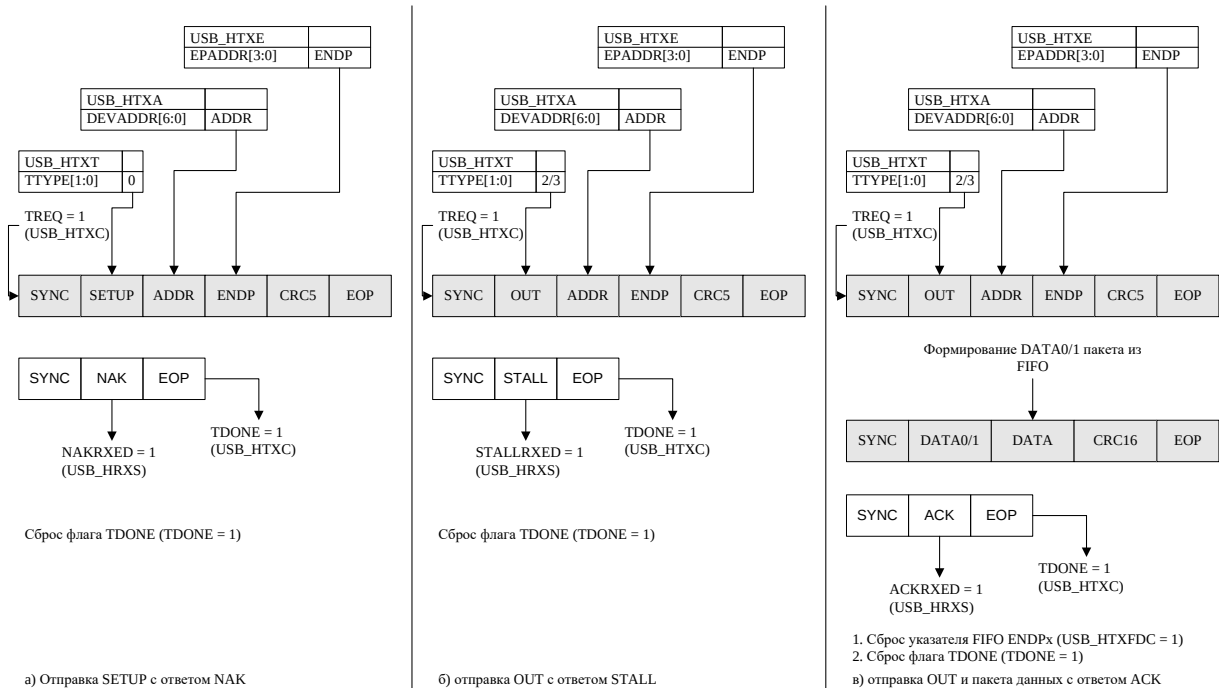


Рисунок 27-3 (а, б, в) – Транзакция SETUP/OUT (USB Host)

27.7 Транзакция IN (USB Host)

Для начала транзакции должны быть заданы адрес устройства (регистр НТХА), оконечная точка (регистр НТХЕ) и тип token пакета (регистр НТХТ). Транзакция запускается при установке бита TREQ регистра НТХС. Host отправляет IN пакет.

После окончания транзакции бит TDONE = 1 (регистр HSI). Этот бит перед началом каждой транзакцией должен быть очищен записью «1». PID принятого пакета записывается в регистре HRXP.

Если в ответ получен пакет NAK (Рисунок 27-4 а), то бит NAKRXED = 1 (регистр HRXS).

Если в ответ получен пакет STALL (Рисунок 27-4 б), то бит STALLRXED = 1 (регистр HRXS).

Если приходит DATA0/DATA1 пакет (Рисунок 27-4 в), то данные начинают записываться побайтно в FIFO приема. После записи каждого байта увеличивается на единицу счетчик принятых байтов. Принятые байты считываются через регистр HRXFD. Количество принятых байтов содержится в регистре HRXFDC. После приема очередного пакета необходимо выполнять сброс указателя FIFO приема записью «1» в регистр HRXFC. Бит DATASEQ регистра HRXS отображает тип принятого пакета данных (0 – DATA0, 1 – DATA1).

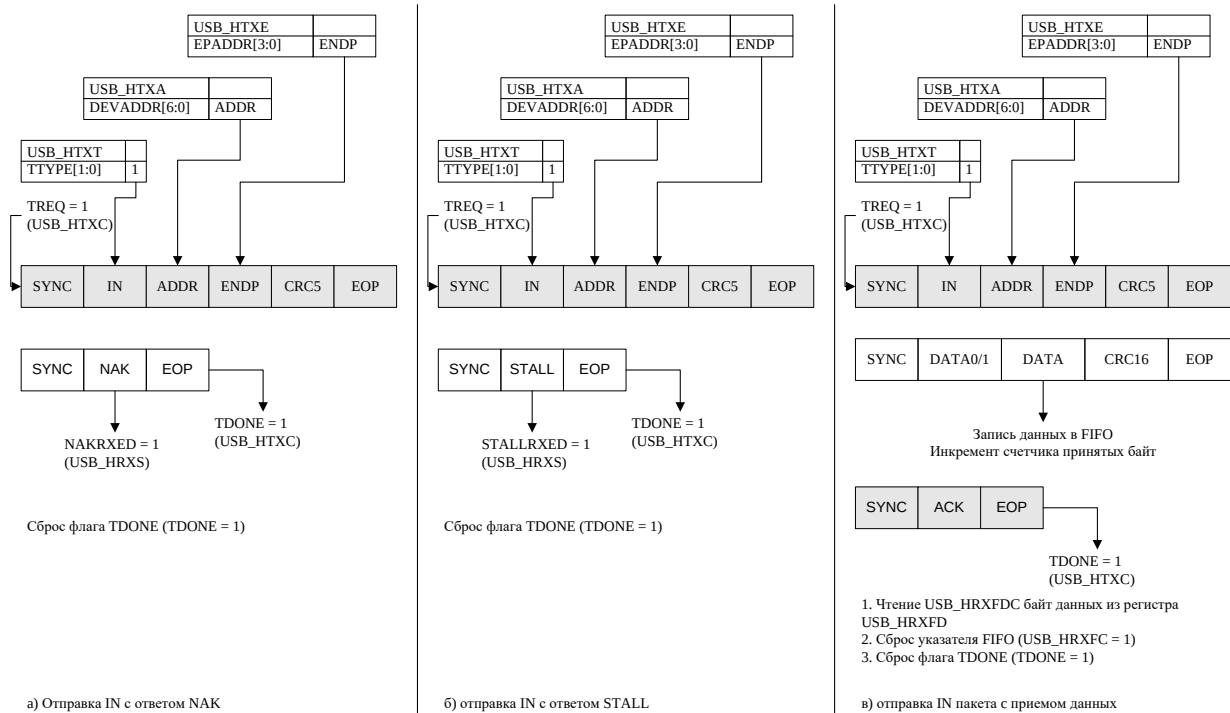


Рисунок 27-4 (а, б, в) – Транзакция IN (USB Host)

27.8 Отправка SOF пакетов и отсчет времени (USB Host)

Для того, чтобы контроллер автоматически отправлял SOF пакеты на Full speed, необходимо установить SOFEN в регистре HTXSE. Если FSPL = 1 (регистр TXLC), то SOF будет автоматически отправляться каждые 1 мс. Если FSPL = 0, то автоматически будет отправляться EOP каждые 1 мс.

После отправки SOF пакета бит SOFS = 1 (регистр HSI). Этот бит должен быть очищен записью «1».

Контроллер ведет счет времени во фрейме таймером. Таймер увеличивается на частоте 48 МГц и имеет 48000 тактов в 1 мс фрейме. Старший байт таймера содержится в регистре HSTM. Этот регистр может быть использован для вычисления оставшегося во фрейме времени.

27.9 Описание регистров управления контроллером интерфейса USB

Таблица 27-1 – Описание регистров управления контроллером интерфейса USB

Базовый адрес	Название	Описание
0x4001_0000	MDR_USB	Контроллер интерфейса USB
Смещение		
0x380	MDR_USB->HSCR	Общее управление для контроллера интерфейса USB
0x384	MDR_USB->HSVR	Версия аппаратного контроллера интерфейса USB
Контроллер HOST		
0x00	MDR_USB->HTXC	Регистр управления передачей пакетов со стороны хоста
0x04	MDR_USB->HTXT	Регистр задания типа передаваемых пакетов со стороны хоста
0x08	MDR_USB->HTXLC	Регистр управления линиями шины USB
0x0C	MDR_USB->HTXSE	Регистр управление автоматической отправки SOF
0x10	MDR_USB->HTXA	Регистр задания адреса устройства для отправки пакета
0x14	MDR_USB->HTXE	Регистр задания номера конечной точки для отправки пакета
0x18 0x1C	MDR_USB->HFN_L MDR_USB->HFN_H	Регистр задания номера фрейма для отправки SOF
0x20	MDR_USB->HSI	Регистр флагов событий контроллера хост.
0x24	MDR_USB->HIM	Регистра флагов разрешения прерываний по событиям контроллера хоста
0x28	MDR_USB->HRXS	Регистр состояния очереди приема данных хоста
0x2C	MDR_USB->HRXP	Регистр отображения PID принятого пакета
0x30	MDR_USB->HRXA	Регистр отображения адреса устройства от которого принят пакет.
0x34	MDR_USB->HRXE	Регистр отображения номер конечной точки от которой принят пакет.
0x38	MDR_USB->HRXCS	Регистр отображения состояния подсоединения устройства
0x3C	MDR_USB->HSTM	Регистр расчета времени фрейма
0x80	MDR_USB->HRXFD	Данные очереди приема
0x88	MDR_USB->HRXFDC	Число принятых данных в очереди
0x90	MDR_USB->HRXFC	Управление очередью приема
0xC0	MDR_USB->HTXFD	Данные для передачи
0xD0	MDR_USB->HTXFC	Управление очередью передачи

Базовый адрес	Название	Описание
Контроллер SLAVE		
0x100 0x110 0x120 0x130	MDR_USB->SEP[x].CTRL	Управление очередью нулевой оконечной точки
0x104 0x114 0x124 0x134	MDR_USB->SEP[x].STS	Состояние оконечной точки
0x108 0x118 0x128 0x138	MDR_USB->SEP[x].TS	Состояние типа передачи оконечной точки
0x10C 0x11C 0x12C 0x13C	MDR_USB->SEP[x].NTS	Состояние передачи NAK оконечной точки
0x140	MDR_USB->SC	Управление контроллеров SLAVE
0x144	MDR_USB->SLS	Отображение состояния линий USB шины
0x148	MDR_USB->SIS	Флаги событий контроллера SLAVE
0x14C	MDR_USB->SIM	Флаги разрешения прерываний от контроллера SLAVE
0x150	MDR_USB->SA	Функциональный адрес контроллера
0x154 0x158	MDR_USB->SFN_L MDR_USB->SFN_H	Номер фрейма
0x180 0x200 0x280 0x300	MDR_USB->SEP[x].RXFD	Принятые данные оконечной точки
0x188 0x208 0x288 0x308	MDR_USB->SEP[x].RXFDC	Число данных в оконечной точке
0x190 0x210 0x290 0x310	MDR_USB->SEP[x].RXFC	Управление очередью приема оконечной точки
0x1C0 0x240 0x2C0 0x340	MDR_USB->SEP[x].TXFD	Данные для передачи через оконечную точку
0x1D0 0x250 0x2D0 0x350	MDR_USB->SEP[x].TXFDC	Управление очередью передачи оконечной точки

27.9.1 MDR_USB->HSCR

Таблица 27-2 – Регистр HSCR

Номер	31...8	7	6	5	4	3	2	1	0
Доступ	U	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0	0	0	0
	-	D- PULL DOWN	D- PULL UP	D+PULL DOWN	D+PULL UP	EN RX	EN TX	RESET CORE	HOST MODE

Таблица 27-3 – Описание бит регистра HSCR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...8	-	Зарезервировано
7	D- PULLDOWN	Управление встроенной подтяжкой линии D-: 0 – нет подтяжки вниз; 1 – есть подтяжка вниз
6	D- PULLUP	Управление встроенной подтяжкой линии D-: 0 – нет подтяжки вверх; 1 – есть подтяжка вверх
5	D+ PULLDOWN	Управление встроенной подтяжкой линии D+: 0 – нет подтяжки вниз; 1 – есть подтяжка вниз
4	D+ PULLUP	Управление встроенной подтяжкой линии D+: 0 – нет подтяжки вверх; 1 – есть подтяжка вверх
3	EN_RX	Разрешение работы приемника USB: 0 – запрещен; 1 – разрешен. Может использоваться в энергосберегающих целях
2	EN_TX	Разрешение работы передатчика USB: 0 – запрещен; 1 – разрешен. Может использоваться в энергосберегающих целях
1	RESET_CORE	Программный сброс контроллера: 1 – сброс контроллера (удерживать минимум 10 циклов USBCLK); 0 – рабочий режим
0	HOST_MODE	Режим работы контроллера: 1 – режим HOST; 0 – режим Device

27.9.2 MDR_USB->HSVR

Таблица 27-4 – Регистр HSVR

Номер	31...8	7...4	3...0
Доступ	U	RO	RO
Сброс	0	0	0
	-	REVISION	VERSION

Таблица 27-5 – Описание бит регистра HSVR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...8	-	Зарезервировано
7...4	REVISION	Номер Ревизии
3...0	VERSION	Номер Версии

Регистры HOST режима

27.9.3 MDR_USB->HTXC

Таблица 27-6 – Регистр HTXC

Номер	31...4	3	2	1	0
Доступ	U	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0
	-	ISOEN	PREEN	SOFS	TREQ

Таблица 27-7 – Описание бит регистра HTXC

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений.
31...4	-	Зарезервировано
3	ISOEN	Флаг разрешения изохронного режима: 1 – разрешение изохронного режима, ACK не посылается и не принимается. Необходимо, что бы TRANS_TYPE_REG был установлен в IN_TRANS или OUTDATA0_TRANS. Isoхронный режим не применим ни с какими другими типами передачи; 0 – запрещение изохронного режима
2	PREEN	Рекомендуется оставлять 0
1	SOFS	Флаг задания синхронизации передачи с SOF: 1 – синхронизировать передачу с окончанием SOF. Передача будет запущена сразу за передачей SOF; 0 – передача не синхронизирована
0	TREQ	Флаг запроса передачи данных: 1 – запрос разрешения передачи данных, автоматически сбрасывается после передачи; 0 – запрещена передача

27.9.4 MDR_USB->HTXT

Таблица 27-8 – Регистр HTXT

Номер	31...2	1	0
Доступ	U	R/W	R/W
Сброс	0	0	0
	-	TTYPE	

Таблица 27-9 – Описание бит регистра НТХТ

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...2	-	Зарезервировано
1...0	TTYPE	Тип передачи: 00 – setup_trans; 01 – in_trans; 10 – outdata0_trans; 01 – outdata1_trans

27.9.5 MDR_USB->HTXLC

Таблица 27-10 – Регистр HTXLC

Номер	31...5	4	3	2	1...0
Доступ	U	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0
	-	FSLR	FSLP	DC	TXLS[1:0]

Таблица 27-11 – Описание бит регистра HTXLC

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...5	-	Зарезервировано
4	FSLR	1 – 12 Мбит в сек; 0 – 1,5 Мбит в сек
3	FSPL	1 – FULL SPEED полярность шины USB; 0 – LOW SPEED полярность шины USB. Если host работает с full speed устройством, full speed полярность должна быть установлена. Если работа ведется с low speed устройством на прямую, то должна быть установлена low speed полярность. Работа с low speed через hub не поддерживается
2	DC	Режим управления линиями шины USB: 1 – разрешение прямого управления состоянием линий USB шины; 0 – нормальный режим работы
1...0	TXLC[1:0]	Если установлен бит DIRECT_CONTROL_BIT, то отображается состояние шины USB: TXLC[0] = D- ; TXLC[1] = D+

27.9.6 MDR_USB->HTXSE

Таблица 27-12 – Регистр HTXSE

Номер	31...1	0
Доступ	U	R/W
Сброс	0	0
	-	SOFEN

Таблица 27-13 – Описание бит регистра HTXSE

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...1	-	Зарезервировано
0	SOFEN	1 – Если FSPL установлен, то SOF будет автоматически отправляться каждые 1 мс. SOF отправляется на full speed независимо от состояния FSPL. Если FSPL не установлен, то автоматически будет отправляться EOP каждые 1 мс. Это необходимо при работе с low speed устройством напрямую (не через хаб); 0 – запрет автоматической отправки SOF/EOP и позволяет подсоединенным устройствам перейти в suspend режим

27.9.7 MDR_USB->HTXA

Таблица 27-14 – Регистр HTXA

Номер	31...7	6...0
Доступ	U	R/W
Сброс	0	0
	-	DEVADDR[6:0]

Таблица 27-15 – Описание бит регистра HTXA

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...7	-	Зарезервировано
6...0	DEVADDR[6:0]	USB Device address. Адрес устройства для обращения

27.9.8 MDR_USB->HTXE

Таблица 27-16 – Регистр HTXE

Номер	31...4	3...0
Доступ	U	R/W
Сброс	0	0
	-	EPADDR[3:0]

Таблица 27-17 – Описание бит регистра HTXE

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...4	-	Зарезервировано
3...0	EPADDR[3:0]	Endpoint address. Номер конечной точки устройства для обращения

27.9.9 MDR_USB->HFN

Таблица 27-18 – Регистр HFN

Номер	31...11	10...0
Доступ	U	R/W
Сброс	0	0
	-	FNUM[10:0]

Таблица 27-19 – Описание бит регистра HFN

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...11	-	Зарезервировано
10...0	FNUM[10:0]	Номер фрейма

27.9.10 MDR_USB->HSI

Таблица 27-20 – Регистр HSI

Номер	31...4	3	2	1	0
Доступ	U	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0
	-	SOFS	CONEV	RESUME	TDONE

Таблица 27-21 – Описание бит регистра HSI

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...4	-	Зарезервировано
3	SOFS	1 – автоматически устанавливается, когда SOF был отправлен. Должен быть очищен записью 1. 0 – не было SOF
2	CONEV	1 – автоматически устанавливается, когда подключение или отсоединение происходит. Должно быть очищено записью 1. 0 – события не было
1	RESUME	1 – автоматически устанавливается, когда возникает состояние повтора. Должен быть очищен записью 1. 0 – не было повтора
	TDONE	1 – автоматически устанавливается, когда передача закончена. Должен быть очищен записью 1. 0 – передача не закончена или ее нет

27.9.11 MDR_USB->HIM

Таблица 27-22 – Регистр HIM

Номер	31...4	3	2	1	0
Доступ	U	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0
	-	SOFSIE	CONEVIE	RESUMEIE	TDONEIE

Таблица 27-23 – Описание бит регистра HIM

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...4	-	Зарезервировано
3	SOFIE	1 – разрешение выработки прерывания при передаче SOF; 0 – запрещение выработки прерывания
2	CONEVIE	1 – разрешение выработки прерывания при подсоединении или отсоединении; 0 – запрещение выработки прерывания
1	RESUMEIE	1 – разрешение выработки прерывания при повторе передачи; 0 – запрещение выработки прерывания
0	TDONEIE	1 – разрешение выработки прерывания при окончании передачи; 0 – запрещение выработки прерывания

27.9.12 MDR_USB->HRXS

Таблица 27-24 – Регистр HRXS

Номер	31...8	7	6	5	4	3	2	1	0
Доступ	U	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0	0	0	0
	-	DATASEQ	ACK RXED	STALL RXED	NAK RXED	RX TO	RXOF	BSERR	CRCER

Таблица 27-25 – Описание бит регистра HRXS

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...8	-	Зарезервировано
7	DATASEQ	Если последняя транзакция была типа IN_TRANS, этот бит указывает номер последнего принятого пакета. DATA0 = 0, DATA1 = 1
6	ACK RXED	1 – получен ACK; 0 – не получен ACK
5	STALL RXED	1 – получен STALL; 0 – не получен STALL
4	NAK RXED	1 – получен NAK от устройства; 0 – не получен NAK
3	RXTO	1 – переполнения времени ожидания ответа от устройства; 0 – нет переполнения времени
2	RXOF	1 – обнаружена ошибка переполнения FIFO при приеме пакета; 0 – не было переполнения
1	BSERR	1 – обнаружена ошибка stuff при последней передаче; 0 – ошибки stuff не было
0	CRCERR	1 – обнаружена ошибка CRC при последней передаче; 0 – ошибки CRC не было

27.9.13 MDR_USB->HRXP

Таблица 27-26 – Регистр HRXP

Номер	31...4	3...0
Доступ	U	R/W
Сброс	0	0
	-	RPID[3:0]

Таблица 27-27 – Описание бит регистра HRXP

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...4	-	Зарезервировано
3...0	RPID[3:0]	Packet identifier от последнего принятого пакета

27.9.14 MDR_USB->HRXA

Таблица 27-28 – Регистр HRXA

Номер	31...7	6...0
Доступ	U	R/W
Сброс	0	0
	-	RADDR[6:0]

Таблица 27-29 – Описание бит регистра HRXA

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...7	-	Зарезервировано
6...0	RADDR[6:0]	Адрес последнего принятого пакета, который был послан

27.9.15 MDR_USB->HRXE

Таблица 27-30 – Регистр HRXE

Номер	31...4	3...0
Доступ	U	R/W
Сброс	0	0
	-	RXENDP[3:0]

Таблица 27-31 – Описание бит регистра HRXE

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений.
31...4	-	Зарезервировано
3...0	RXENDP[3:0]	Номер окончной точки в последнем принятом пакете, который был послан

27.9.16 MDR_USB->HRXCS

Таблица 27-32 – Регистр HRXCS

Номер	31...2	1, 0
Доступ	U	R/W
Сброс	0	0
	-	RXLS[1:0]

Таблица 27-33 – Описание бит регистра HRXCS

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...2	-	Зарезервировано
1, 0	RXLS[1:0]	Состояние линий шины USB: DISCONNECT = 0; LOW_SPEED_CONNECT = 1; FULL_SPEED_CONNECT = 2

27.9.17 MDR_USB->HSTM

Таблица 27-34 – Регистр HSTM

Номер	31...8	7...0
Доступ	U	R/W
Сброс	0	0
	-	HSTM[7:0]

Таблица 27-35 – Описание бит регистра HSTM

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...8	-	Зарезервировано
7...0	HSTM[7:0]	Старший байт SOF таймера, используемого для передачи SOF. Таймер увеличивается на частоте 48 МГц, и имеет 48000 тактов в 1 мс фрейме. Этот регистр может быть использован для вычисления оставшегося во фрейме времени

27.9.18 MDR_USB->HRXFD

Таблица 27-36 – Регистр HRXFD

Номер	31...8	7...0
Доступ	U	R/W
Сброс	0	0
	-	RX FIFO DATA[7:0]

Таблица 27-37 – Описание бит регистра HRXFD

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...8	-	Зарезервировано
7...0	RX FIFO DATA[7:0]	Если последняя транзакция была IN_TRANS, то порт содержит принятые данные, и они могут быть считаны

27.9.19 MDR_USB->HRXFDC

Таблица 27-38 – Регистр HRXFDC

Номер	31...16	15...0
Доступ	U	R/W
Сброс	0	0
	-	FIFO DATA COUNT[15:0]

Таблица 27-39 – Описание бит регистра HRXFDC

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...16	-	Зарезервировано
15...0	FIFO DATA COUNT[15:0]	Счетчик принятых байт в очереди

27.9.20 MDR_USB->HRXFC

Таблица 27-40 – Регистр HRXFC

Номер	31...1	0
Доступ	U	R/W
Сброс	0	0
	-	FIFO FORCE EMPTY

Таблица 27-41 – Описание бит регистра HRXFC

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...1	-	Зарезервировано
0	FIFO FORCE EMPTY	Запись 1 принудительно сбрасывает очередь

27.9.21 MDR_USB->HTXFD

Таблица 27-42 – Регистр HTXFD

Номер	31...8	7...0
Доступ	U	R/W
Сброс	0	0
	-	TX FIFO DATA[7:0]

Таблица 27-43 – Описание бит регистра HTXFD

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...8	-	Зарезервировано
7...0	TX FIFO DATA[7:0]	При запросах передачи OUTDATA0_TRANS или OUTDATA1_TRANS, через данный порт должны быть загружены данные для отправки

27.9.22 MDR_USB->HTXFC

Таблица 27-44 – Регистр HTXFC

Номер	31...1	0
Доступ	U	R/W
Сброс	0	0
	-	FIFO FORCE EMPTY

Таблица 27-45 – Описание бит регистра HTXFC

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...1	-	Зарезервировано
0	FIFO FORCE EMPTY	Запись 1 принудительно сбрасывает очередь

USB Slave (Device)

27.9.23 MDR_USB->SEP[x].CTRL

Таблица 27-46 – Регистр SEP[x].CTRL

Номер	31...5	4	3	2	1	0
Доступ	U	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0
	-	EPISOEN	EPSSTALL	EPDATASEQ	EPRDY	EPEN

Таблица 27-47 – Описание бит регистра USB_SEPx.CTRL

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...5	-	Зарезервировано
4	EPISOEN	0 – не изохронный режим передачи; 1 – разрешить изохронные передачи. В изохронном режиме не отсылаются какие-либо подтверждения передачи
3	EPSSTALL	0 – не отвечать STALL на запрос; 1 – если точка разрешена, готова, и не в изохронном режиме, то на запрос хоста будет отвечать STALL
2	EPDATASEQ	0 – отвечать на IN запрос от хоста с DATA0; 1 – отвечать на IN запрос от хоста с DATA1
1	EPRDY	0 – оконечная точка не готова или закончила передачу; 1 – оконечная точка готова. Если точка разрешена и готова, то она может ответить на инициализированную хостом передачу. Автоматически сбрасывается в 0 после успешного окончания передачи
0	EPEN	0 – оконечная точка запрещена; 1 – оконечная точка разрешена. Если точка запрещена она не отвечает на транзакции, если точка разрешена, но не готова и не находится в изохронном режиме, то отвечает NAK

27.9.24 MDR_USB->SEP[x].STS

Таблица 27-48 – Регистр SEP[x].STS

Номер	31...8	7	6	5	4	3	2	1	0
Доступ	U	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0	0	0	0
	-	SC DATA SEQ	SC ACK RXED	SC STALL SENT	NAK SENT	SC RXTO	SC RXOF	SC BS ERR	SC CRC ERR

Таблица 27-49 – Описание бит регистра USB_SEPx.STS

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...8	-	Зарезервировано
7	SC DATA SEQ	Если предыдущий тип передачи был OUT_TRANS, то этот бит определяет тип принятого пакета DATA0 = 0, DATA1= 1
6	SC ACK RXED	0 – нет подтверждения; 1 – получено подтверждение ACK от хоста на переданные данные.
5	SC STALL SENT	0 – не было STALL; 1 – признак отправки STALL
4	NAK SENT	1 – признак отправки NAK ответа. 0 – не было NAK
3	SC RXTO	1 – признак возникновения ошибки времени ожидания ответа от хоста. 0 – нет ошибки
2	SC RXOF	0 – нет переполнения; 1 – признак возникновения переполнения очереди при последней передаче
1	SC BS ERR	0 – нет ошибки; 1 – признак возникновения STUFF ошибки в последней передаче
0	SC CRC ERR	0 – нет ошибки; 1 – признак возникновения CRC ошибки в последней передаче

27.9.25 MDR_USB->SEP[x].TS

Таблица 27-50 – Регистр SEP[x].TS

Номер	31...2	1, 0
Доступ	U	R/W
Сброс	0	0
	-	SCTTYPE[1:0]

Таблица 27-51 – Описание бит регистра SEP[x].TS

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...2	-	Зарезервировано
1...0	SCTTYPE[1:0]	Отображает тип последней передачи перед тем как ENDPOINT_READY_BIT был изменен с 1 на 0. SC_SETUP_TRANS = 0; SC_IN_TRANS = 1; SC_OUTDATA_TRANS = 2

27.9.26 MDR_USB->SEP[x].NTS

Таблица 27-52 – Регистр SEP[x].NTS

Номер	31...2	1, 0
Доступ	U	R/W
Сброс	0	0
	-	NTTYPE[1:0]

Таблица 27-53 – Описание бит регистра USB_SEPx.NTS

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...2	-	Зарезервировано
1...0	NTTYPE[1:0]	Тип последней передачи, в результате которой на хост был послан NAK. SC_SETUP_TRANS = 0; SC_IN_TRANS = 1; SC_OUTDATA_TRANS = 2

27.9.27 MDR_USB->SC

Таблица 27-54 – Регистр SC

Номер	31...6	5	4	3	2, 1	0
Доступ	U	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0
	-	SCFSR	SCFSP	SCDC	SCTXLS[1:0]	SCGEN

Таблица 27-55 – Описание бит регистра USB_SC

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...6	-	Зарезервировано
5	SCFSR	Флаг управления скоростью работы: 1 – 12 Мбит/с; 0 – 1,5 Мбит/с
4	SCFSP	Флаг выбора полярности линий шины USB: 1 – FULL SPEED; 0 – LOW SPEED
3	SCDC	Флаг прямого управления линиями шины USB: 1 – разрешено прямое управление; 0 – запрещено прямое управление

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
2, 1	SCTXL[1:0]	Если установлен бит SC_DIRECT_CONTROL_BIT, то через SC_TX_LINE_STATE осуществляется прямое управление состоянием линий USB шины: SC_TX_LINE_STATE [2] = D+ ; SC_TX_LINE_STATE [1] = D-
0	SCGEN	1 – разрешение для работы с разрешенных конечных точек; 0 – все конечные точки запрещены

27.9.28 MDR_USB->SLS

Таблица 27-56 – Регистр SLS

Номер	31...2	1	0
Доступ	U	R/W	R/W
Сброс	0	0	0
	-	SCRXLS[1:0]	

Таблица 27-57 – Описание бит регистра SLS

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...2	-	Зарезервировано
1...0	SCRXLS[1:0]	Отображает состояние подсоединения на шине USB: RESET = 0; LOW_SPEED_CONNECT = 1; FULL_SPEED_CONNECT = 2

27.9.29 MDR_USB->SIS

Таблица 27-58 – Регистр SIS

Номер	31...6	5	4	3	2	1	0
Доступ	U	U	R/W	R/W	R/W	R/W	R/W
Сброс	0	1	0	0	0	0	0
	-	-	SC NAK SENT	SC SOF REC	SC RESET EV	SC RESUME	SC TDONE

Таблица 27-59 – Описание бит регистра USB_SIS

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...5	-	Зарезервировано
4	SC NAK SENT	При ответе NAK на запрос от хоста автоматически устанавливается в 1. Очищается записью 1
3	SC SOF REC	При принятии пакета SOF от хоста автоматически устанавливается в 1. Очищается записью 1

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
2	SC RESET EV	Автоматически устанавливается в 1 при наличии состояния сброса на шине USB. Очищается записью 1
1	SC RESUME	Автоматически устанавливается в 1 при обнаружении состояние повтора. Очищается записью 1
0	SC TDONE	Автоматически устанавливается в 1 после успешного выполнения передачи. Очищается записью 1
Примечание – Для организации надежного сброса статусных флагов прерываний необходимо учитывать требования, изложенные в подразделе 36.3		

27.9.30 MDR_USB->SIM

Таблица 27-60 – Регистр SIM

Номер	31...5	4	3	2	1	0
Доступ	U	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0
	-	SC NAK SENT IE	SC SOF RECIE	SC RESET EVIE	SC RESUME IE	SC TDONE IE

Таблица 27-61 – Описание бит регистра B_SIM

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...6	-	Зарезервировано
4	SC NAK SENT IE	Флаг управления разрешением прерывания при отправке NAK: 1 – разрешено прерывание; 0 – запрещено прерывание
3	SC SOF RECIE	Флаг управления разрешением прерывания при приеме SOF: 1 – разрешено прерывание; 0 – запрещено прерывание
2	SC RESET EVIE	Флаг управления разрешением прерывания при состоянии сброса на шине: 1 – разрешено прерывание; 0 – запрещено прерывание
1	SC RESUME IE	Флаг управления разрешением прерывания при состоянии повтора: 1 – разрешено прерывание; 0 – запрещено прерывание
0	SC TDONE IE	Флаг управления разрешением прерывания при окончании передачи: 1 – разрешено прерывание; 0 – запрещено прерывание

27.9.31 MDR_USB->SA

Таблица 27-62 – Регистр SA

Номер	31...7	6...0
Доступ	U	R/W
Сброс	0	0
	-	SDEVADDR[6:0]

Таблица 27-63 – Описание бит регистра SA

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...7	-	Зарезервировано
6...0	SDEVADDR[6:0]	Функциональный адрес устройства USB

27.9.32 MDR_USB->SFN

Таблица 27-64 – Регистр SFN

Номер	31...11	10...0
Доступ	U	R/W
Сброс	0	0
	-	FRAME NUM[10:0]

Таблица 27-65 – Описание бит регистра SFN

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...11	-	Зарезервировано
10...0	FRAME NUM[10:0]	Номер фрейма, принятый в последнем SOF

27.9.33 MDR_USB->SEP[x].RXFD

Таблица 27-66 – Регистр SEP[x].RXFD

Номер	31...8	7...0
Доступ	U	R/W
Сброс	0	0
	-	RX FIFO DATA[7:0]

Таблица 27-67 – Описание бит регистра SEP[x].RXFD

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...8	-	Зарезервировано
7...0	RX FIFO DATA[7:0]	После приема OUTDATA_TRANS или SETUP_TRANS пакета, принятые данные читаются из регистра RX_FIFO_DATA

27.9.34 MDR_USB->SEP[x].RXFDC

Таблица 27-68 – Регистр SEP[x].RXFDC

Номер	31...16	15...0
Доступ	U	R/W
Сброс	0	0
	-	FIFO DATA COUNT[15:0]

Таблица 27-69 – Описание бит регистра SEP[x].RXFDC

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...16	-	Зарезервировано
15...0	FIFO DATA COUNT[15:0]	Отображает число принятых байт в очереди

27.9.35 MDR_USB->SEP[x].RXFC

Таблица 27-70 – Регистр SEP[x].RXFC

Номер	31...1	0
Доступ	U	R/W
Сброс	0	0
	-	FIFO FORCE EMPTY

Таблица 27-71 – Описание бит регистра USB_SEPx.RXFC

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...1	-	Зарезервировано
0	FIFO FORCE EMPTY	Запись 1 очищает всю очередь

27.9.36 MDR_USB->SEP[x].TXFD

Таблица 27-72 – Регистр SEP[x].TXFD

Номер	31...8	7...0
Доступ	U	R/W
Сброс	0	0
	-	TX FIFO DATA[7:0]

Таблица 27-73 – Описание бит регистра SEP[x].TXFD

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...8	-	Зарезервировано
7...0	TX FIFO DATA [7:0]	Перед приемом IN_TRANS в очередь записываются данные для отправки

27.9.37 MDR_USB->SEP[x].TXFDC

Таблица 27-74 – Регистр SEP[x].TXFDC

Номер	31...1	0
Доступ	U	R/W
Сброс	0	0
	-	FIFO FORCE EMPTY

Таблица 27-75 – Описание бит регистра SEP[x].TXFDC

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...1	-	Зарезервировано
0	FIFO FORCE EMPTY	Запись 1 очищает всю очередь

28 Таймеры общего назначения MDR_TIMERx

В микросхеме реализовано три блока таймеров общего назначения, каждый из которых может быть использован для широкого спектра применений, включая:

- подсчет циклов частоты TIM_CLK или каких-либо внешних событий;
- формирование прерываний и запросов DMA по заданным событиям;
- захват входных сигналов, в том числе измерение длительности импульсов входных сигналов;
- генерацию различных форм выходных сигналов.

Основу таймеров составляет 16-битный перезагружаемый счетчик. Счет может быть прямой, обратный или двунаправленный. В качестве источника синхросигнала может выступать внутренняя тактовая частота TIM_CLK, внешние сигналы или другие таймеры.

В каждый блок таймера входит четыре канала, которые имеют в своем составе схему захвата и блок ШИМ с функциями формирования «мертвой зоны» и аппаратной блокировки.

Каждый из таймеров позволяет генерировать прерывания и запросы DMA.

28.1 Основные характеристики

Основные характеристики блока таймера:

- 16-битный счетчик: счет прямой, обратный или двунаправленный;
- 16-разрядный предварительный делитель частоты TIM_CLK;
- схема выбора источника тактирования основного счетчика от внешних сигналов или от других таймеров;
- четыре независимых канала, каждый канал может работать в одном из режимов:
 - режим захвата: позволяет захватить (сохранить) текущее значение счетчика при изменении некоторого входного сигнала;
 - режим ШИМ: позволяет осуществлять непрерывное сравнение заданных значений со значением счетчика для формирования выходных сигналов;
 - формирование выходных сигналов в режиме ШИМ:
 - сброс в НИЗКИЙ уровень при совпадении;
 - установка в ВЫСОКИЙ уровень при совпадении;
 - переключение (инвертирование) при совпадении;
 - переключение при некотором условии;
 - формирование прерываний и запросов DMA по событиям:
 - обновление счетчика;
 - захват;
 - сравнение;
 - внешние события по входам ETR и BRK.

28.1.1 Структурная схема

Структурная схема блока «Таймер» представлена на рисунке ниже (Рисунок 28-1).

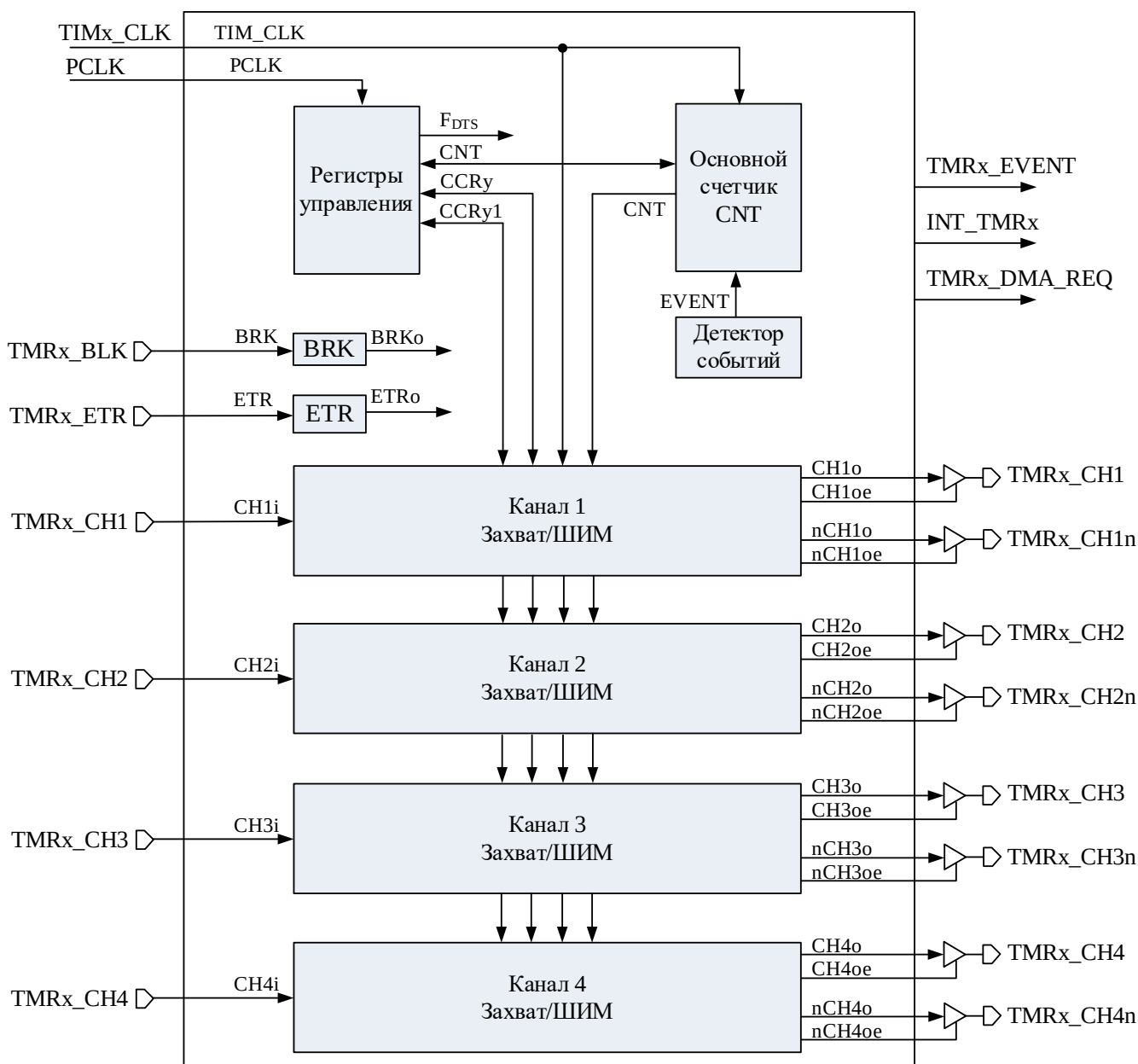


Рисунок 28-1 – Структурная схема блока «Таймер»

Таймер содержит основной 16-битный счетчик CNT, блок регистров управления и четыре канала схем захвата/ШИМ.

Таймер позволяет работать в режимах:

- таймер;
- расширенный таймер, с объединением нескольких таймеров;
- схема захвата;
- схема ШИМ.

28.2 Базовый таймер счета

Таймер построен на базе 16-битного счетчика. Базовый блок таймера включает в себя:

- основной счетчик таймера (CNT);
- основание счета (максимальное значение) основного счетчика (ARR);
- делитель частоты TIM_CLK (PSG), используемый для тактирования основного счетчик;
- регистр управления основным счетчиком (CNTRL).

Сигналом для изменения основного счетчика CNT может служить как внутренняя частота TIM_CLK, так и события в других счетчиках, либо внешние входные сигналы (см. подраздел 28.4 «Источники событий для счета»).

28.2.1 Инициализация тактирования таймера

Перед началом работы с таймером в первую очередь должны быть включены тактовые сигналы. Параметры задаются в блоке «Сигналы тактовой частоты».

Таймер общего назначения тактируется частотами PCLK и TIM_CLK. Частота PCLK используется для записи/чтения регистров блока по шине APB. Работа блоков таймера осуществляется на частоте TIM_CLK.

Для разрешения тактовой частоты PCLK необходимо установить бит тактирования блока в регистре PER_CLOCK: бит 14 для таймера 1, бит 15 для таймера 2, бит 16 для таймера 3. Задание тактовой частоты TIM_CLK осуществляется в регистре TIM_CLOCK: в поле TIMxBRG[7:0] устанавливается коэффициент деления тактовой частоты HCLK для формирования частоты TIM_CLK, разрешение подачи частоты TIM_CLK на блок таймера управляется битом TIMxCLKEN.

После подачи тактовых сигналов можно приступить к работе с таймером.

28.2.2 Инициализация основного счетчика таймера

Чтобы запустить работу основного счетчика необходимо задать:

- начальное значение основного счетчика таймера в регистре CNT;
- значение основания счета для основного счетчика в регистре ARR;
- режим работы счетчика в регистре CNTRL:
 - выбрать источник события переключения счетчика EVENT_SEL[3:0];
 - режим счета основного счетчика CNT_MODE[1:0]:
 - значения 00 и 01 при тактировании внутренней частотой;
 - значения 10 при тактировании внешними сигналами;
 - направление счета основного счетчика DIR;
- при тактировании внутренней частотой установить значение предварительного делителя в регистре PSG, основной счетчик при этом будет считать на частоте $TIM\_CLK_d = TIM\_CLK / (PSG + 1)$;
- разрешить работу счетчика CNT_EN.

Значения регистров CNT, PSG и ARR можно изменять даже во время работы счетчика. Значения регистров CNT и PSG вступают в силу мгновенно после их записи. Значение регистра основания счета (ARR) может вступить в силу сразу после записи, если в регистре CNTRL бит ARRB_EN = 0.

При установленном бите ARRB_EN = 1 записанное значение ARR применяется при CNT == ARR. Необходимо учитывать, что если установлен прямой счет таймера, то новое значение ARR будет использоваться в следующем периоде счета. Если установлен обратный счет таймера, то новое значение ARR будет использовано через один период счета.

Поле CNT_MODE[1:0] в регистре CNTRL определяет режим работы основного счетчика:

- CNT_MODE[1:0] = 00 или 10 – направление счета определяется битом DIR:
 - DIR = 0 – счет прямой;
 - DIR = 1 – счет обратный;
- CNT_MODE[1:0] = 01 – счет двунаправленный с автоматическим изменением

DIR.

28.3 Режимы счета

28.3.1 Счет прямой: CNT_MODE[1:0] = 00, DIR = 0

MDR_TIMERx->CNTRL = 0x00000000; //Режим инициализации таймера

//Настраиваем работу основного счетчика

MDR_TIMERx->CNT = 0x00000004; //Начальное значение счетчика

MDR_TIMERx->PSG = 0x00000000; //Предделитель частоты TIM_CLK

MDR_TIMERx->ARR = 0x00000013; //Основание счета

//Разрешение работы таймера

MDR_TIMERx->CNTRL = 0x00000001; //Счет прямой по TIM_CLK

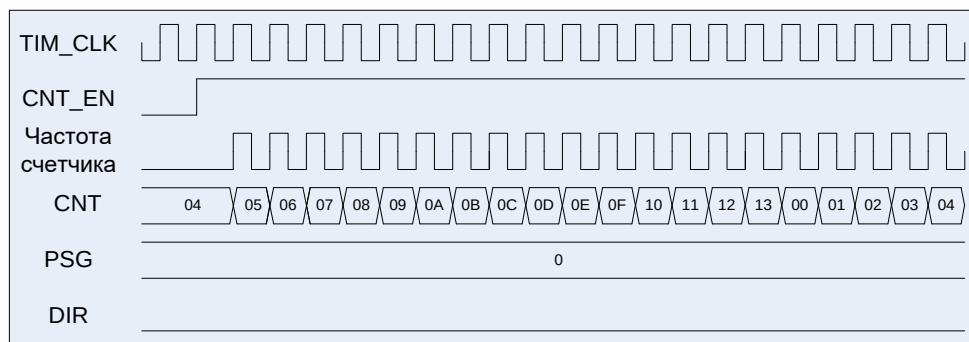


Рисунок 28-2 – Диаграммы работы таймера, счет прямой от 0 до 0x13, стартовое значение 0x04

28.3.2 Счет обратный: CNT_MODE[1:0] = 00, DIR = 1

MDR_TIMERx->CNTRL = 0x00000000; //Режим инициализации таймера

//Настраиваем работу основного счетчика

MDR_TIMERx->CNT = 0x00000004; //Начальное значение счетчика

MDR_TIMERx->PSG = 0x00000000; //Предделитель частоты TIM_CLK

MDR_TIMERx->ARR = 0x00000013; //Основание счета

//Разрешение работы таймера.

MDR_TIMERx->CNTRL = 0x00000009; //Счет обратный по TIM_CLK

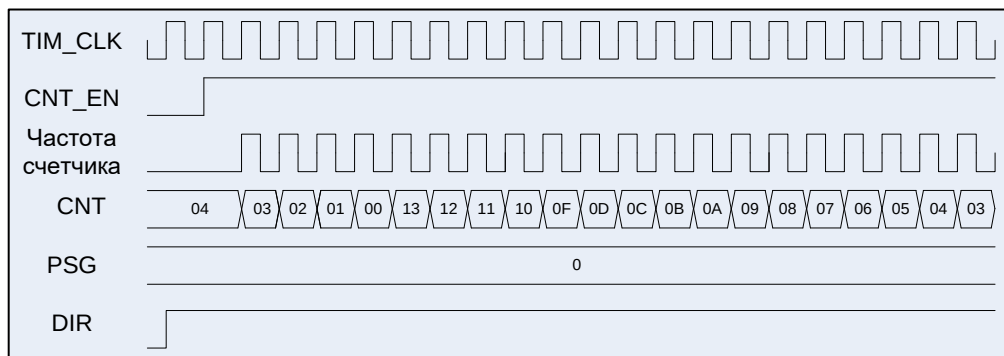


Рисунок 28-3 – Диаграммы работы таймера, счет обратный от 0x13 до 0, стартовое значение 0x04

28.3.3 Счет двунаправленный: CNT_MODE = 01, DIR = 0

MDR_TIMERx->CNTRL = 0x00000000; //Режим инициализации таймера

//Настраиваем работу основного счетчика

MDR_TIMERx->CNT = 0x00000004; //Начальное значение счетчика

MDR_TIMERx->PSG = 0x00000000; //Предделитель частоты TIM_CLK

MDR_TIMERx->ARR = 0x00000013; //Основание счета

//Разрешение работы таймера.

MDR_TIMERx->CNTRL = 0x00000041; //Счет двунаправленный по TIM_CLK

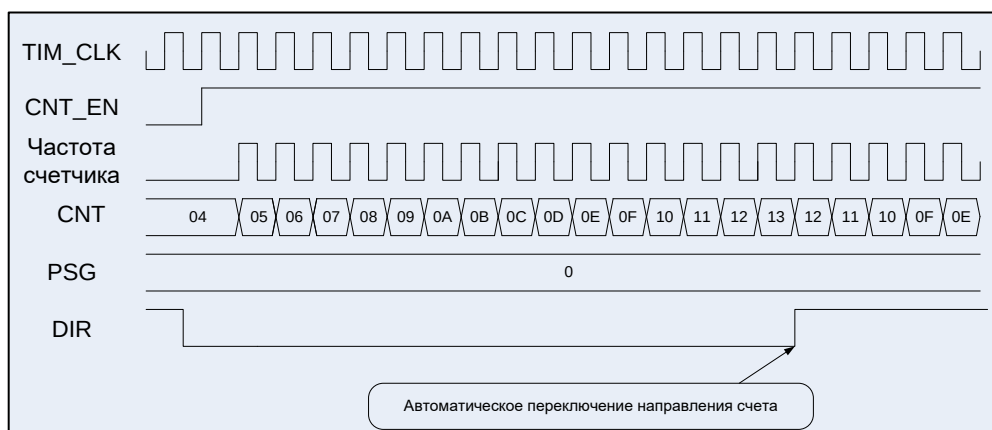


Рисунок 28-4 – Диаграммы работы таймера, счет двунаправленный, сначала прямой

28.3.4 Счет двунаправленный: CNT_MODE = 01, DIR = 1

MDR_TIMERx->CNTRL = 0x00000000; //Режим инициализации таймера

//Настраиваем работу основного счетчика

MDR_TIMERx->CNT = 0x00000004; //Начальное значение счетчика

MDR_TIMERx->PSG = 0x00000000; //Предделитель частоты TIM_CLK

MDR_TIMERx->ARR = 0x00000013; //Основание счета

//Разрешение работы таймера.

MDR_TIMERx->CNTRL = 0x00000049; //Счет двунаправленный по TIM_CLK

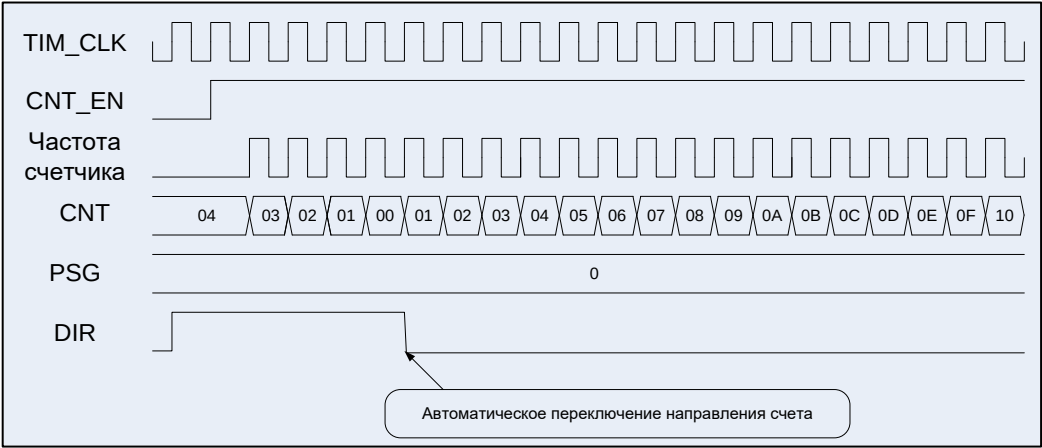


Рисунок 28-5 – Диаграммы работы таймера, счет двунаправленный, сначала обратный

28.3.5 Тактовая частота F_{DTS}

В блоке таймера предусмотрено формирование дополнительной тактовой частоты F_{DTS}, которая может использоваться для работы генератора «мертвой зоны» и цифровых фильтров на входах ETR и CНуi.

Тактовая частота F_{DTS} формируется из частоты TIM_CLK путем прореживания на заданный коэффициент (1, 2, 3 или 4). Настройка частоты F_{DTS} осуществляется в поле F_{DTS}[1:0] регистра CNTRL.

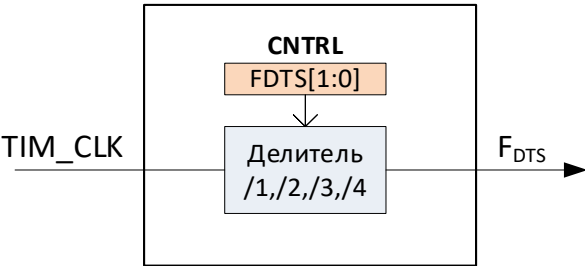


Рисунок 28-6 – Схема формирования тактовой частоты F_{DTS}

Рисунок 28-7 демонстрирует диаграмму возможных частот F_{DTS}.

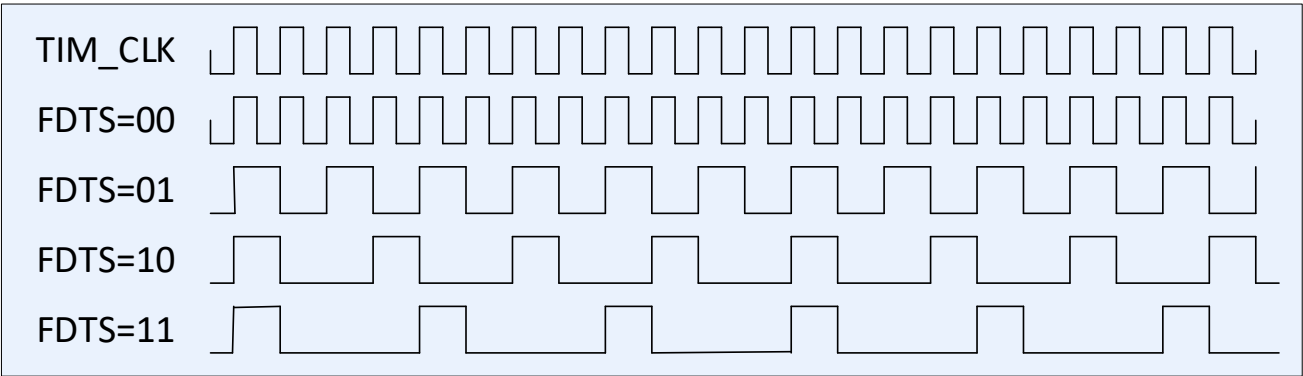


Рисунок 28-7 – Диаграмма тактовой частоты F_{DTS} в зависимости от значения F_{DTS}[1:0] в регистре CNTRL

28.4 Источники событий для счета

Тактирование основного счетчика таймера может осуществляться от следующих источников:

- внутренний тактовый сигнал (TIM_CLKd);
- событие в другом таймере (CNT==ARR);
- внешний тактовый сигнал, «Режим 1»: событие переднего фронта на входе канала CH_Ui;
- внешний тактовый сигнал, «Режим 2»: событие переднего или заднего фронта на входе ETR.

Выбор источника тактирования основного счетчика осуществляется в регистре CNTRL, поле EVENT_SEL[3:0]. При выборе любого источника, кроме внутреннего тактового сигнала (EVENT_SEL[3:0] = 0000), необходимо также установить CNT_MODE[1:0] = 10 в регистре CNTRL.

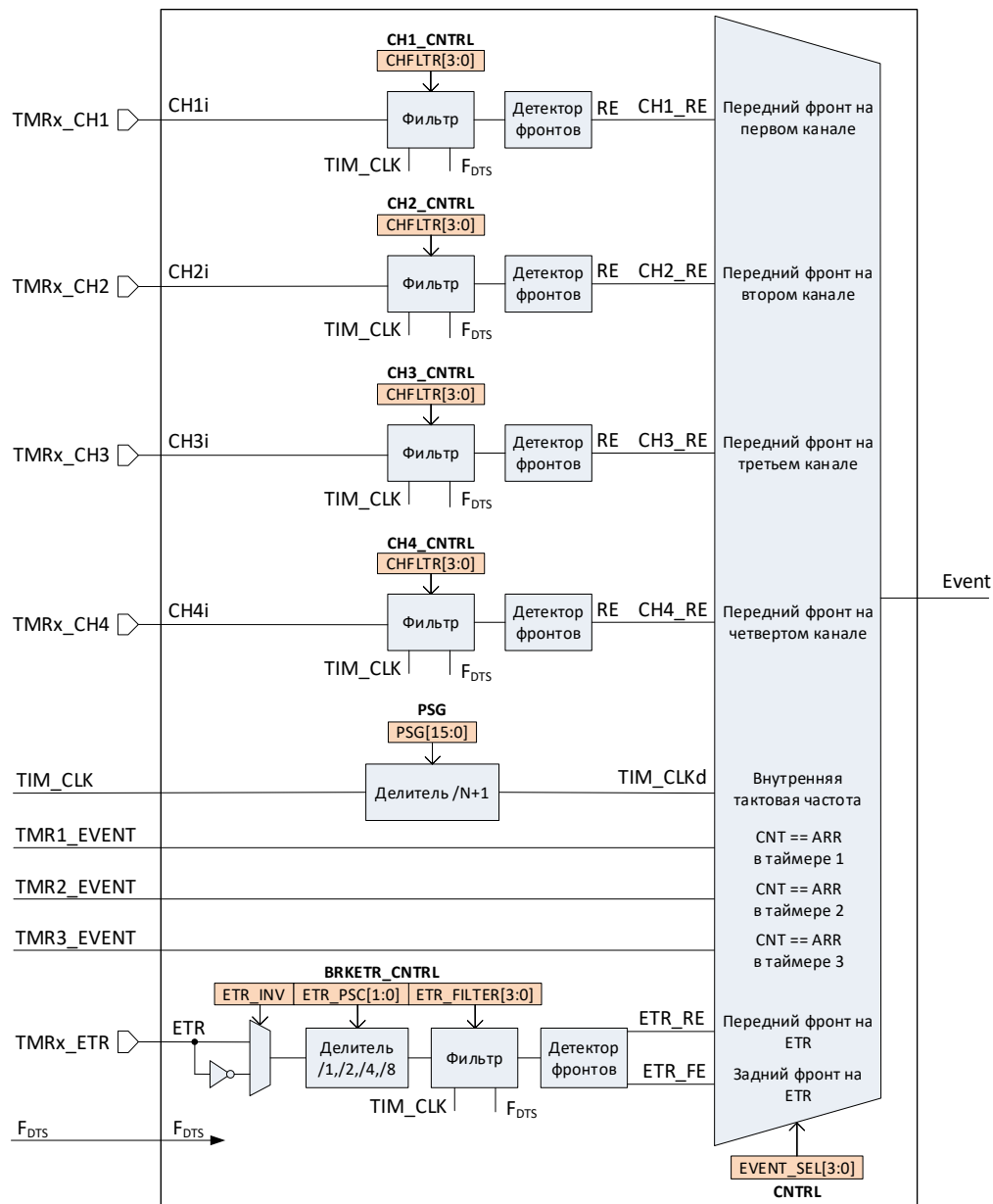


Рисунок 28-8 – Структурная схема формирования события для счета

28.4.1 Внутренний тактовый сигнал (TIM_CLKd)

Данный режим выбирается, когда EVENT_SEL[3:0] = 0000 и CNT_MODE[1:0] = 0x в регистре CNTRL. Основной счетчик таймера тактируется от внутренней частоты TIM_CLKd, которая формируется путем деления частоты TIM_CLK в соответствии с коэффициентом деления, записанным в регистре PSG.

Если значение предварительного делителя основного счетчика (PSG) неравно нулю, то счетный регистр делителя будет инкрементироваться по каждому импульсу сигнала TIM_CLK до тех пор, пока не достигнет значения, находящегося в регистре делителя. Далее счетный регистр делителя сбрасывается в ноль, содержимое основного счетчика таймера изменяется на 1 и счет начинается заново. Таким образом выходная частота предварительного делителя составляет

$TIM\_CLKd = \frac{TIM\_CLK}{PSG + 1}$	(1)

Значение регистра PSG можно изменять даже во время работы счетчика, новое значение предделителя вступит в силу сразу после записи. Рисунок 28-9 и Рисунок 28-10 демонстрируют диаграммы работы счетчика при обновлении значения PSG.

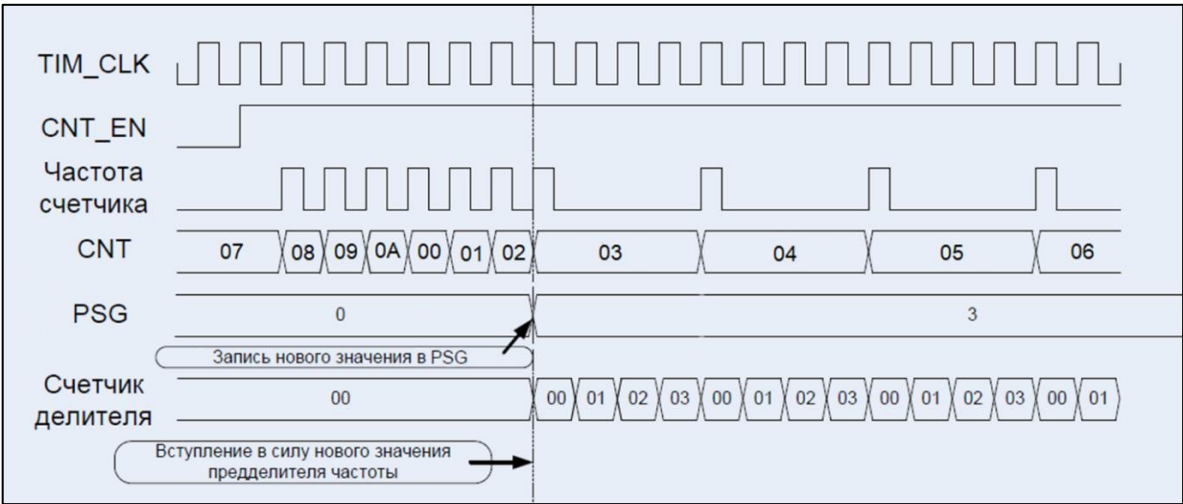


Рисунок 28-9 – Диаграмма работы счетчика: счет прямой (CNT_MODE[1:0] = 00, EVENT_SEL[3:0] = 0000, DIR = 0)

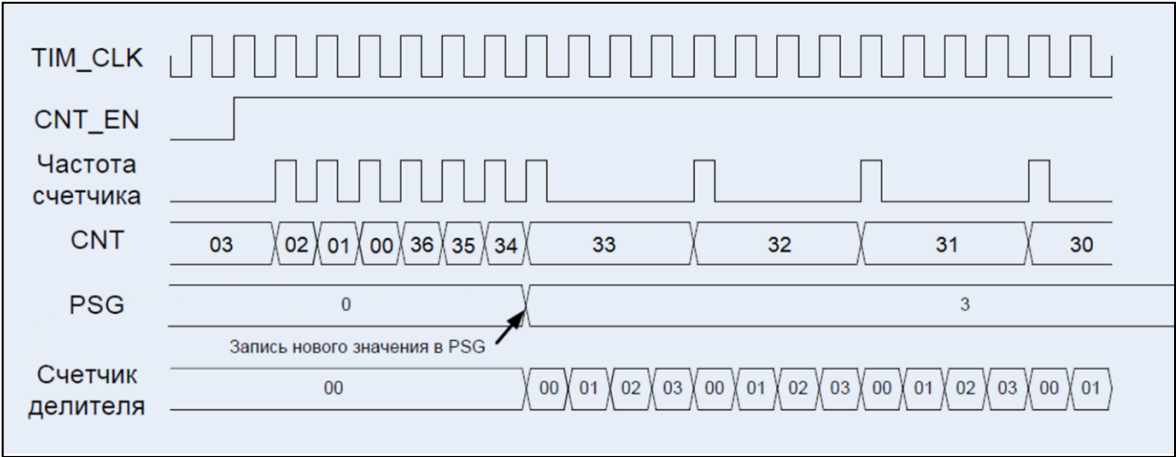


Рисунок 28-10 – Диаграмма работы счетчика: счет обратный (CNT_MODE[1:0] = 00, EVENT_SEL[3:0] = 0000, DIR = 1)

28.4.2 Событие в другом таймере (CNT==ARR)

Все таймеры полностью независимы друг от друга, но при этом у них предусмотрена возможность синхронизированной работы. Это позволяет создавать более сложные массивы таймеров, которые работают полностью автономно и не требуют написания какого-либо кода программы для выполнения сложных временных функций.

У каждого таймера имеется выход запуска TMRx_EVENT, который соединен с входами других таймеров. Тактирование от другого таймера выбирается, когда EVENT_SEL[3:0] = 0001 – 0011 и CNT_MODE[1:0] = 10 в регистре CNTRL. Основной счетчик таймера тактируется от другого таймера по сигналу TMRx_EVENT, который устанавливается при CNT == ARR. Пересинхронизация сигнала TMRx_EVENT (CNT == ARR) с одного таймера на другой происходит с задержкой один такт частоты TIM_CLK.

Синхронизация таймеров возможна в различных режимах. Рисунок 28-11 демонстрирует пример каскадного соединения таймеров, а Рисунок 28-12 – диаграммы работы данных таймеров.

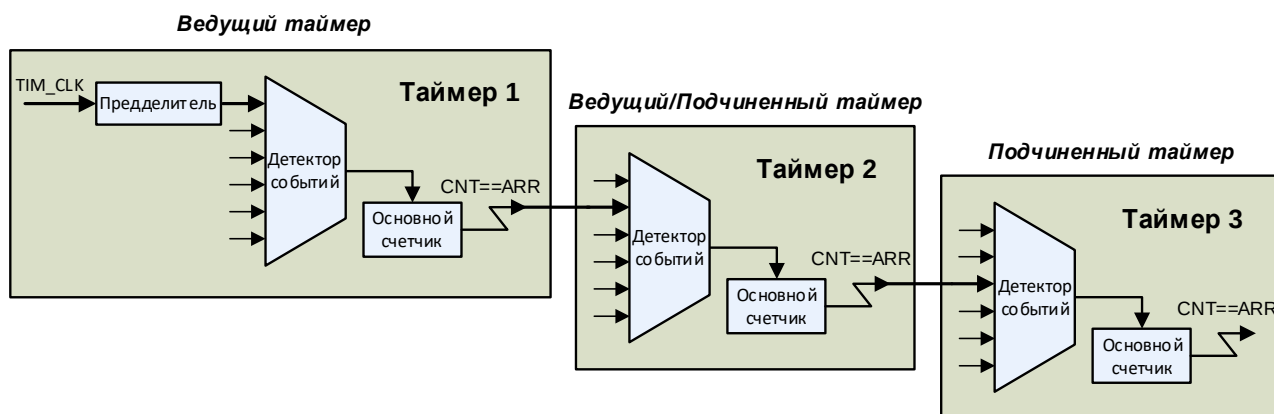


Рисунок 28-11 – Пример каскадного соединения таймеров

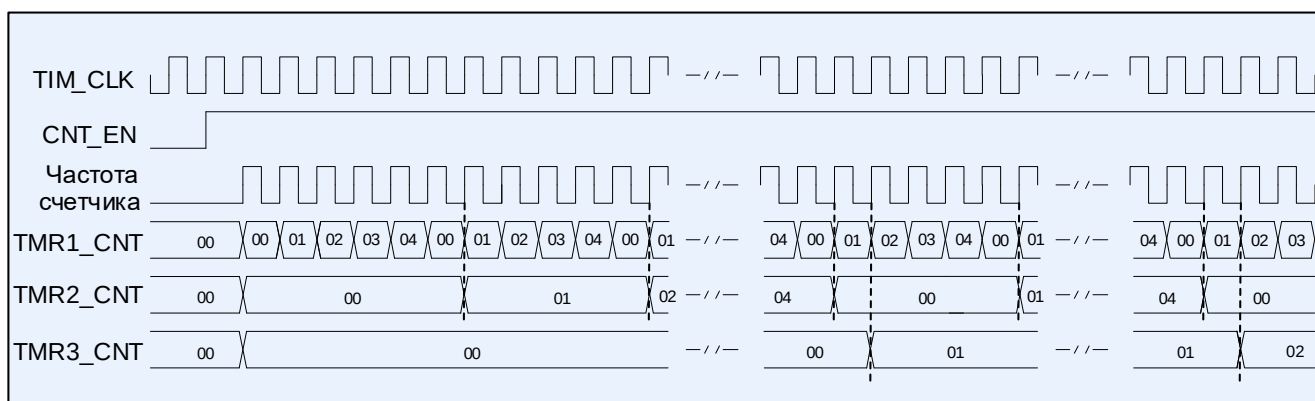


Рисунок 28-12 – Диаграммы работы трех таймеров в каскаде

MDR_TIMER1: DIR = 0, EVENT_SEL[3:0] = 0000, CNT_MODE[1:0] = 00;
MDR_TIMER2: DIR = 0, EVENT_SEL[3:0] = 0001, CNT_MODE[1:0] = 10;
MDR_TIMER3: DIR = 0, EVENT_SEL[3:0] = 0010, CNT_MODE[1:0] = 10

28.4.3 Внешний тактовый сигнал, «Режим 1»: событие переднего фронта на входе канала CHyі

Данный режим выбирается, когда $EVENT\_SEL[3:0] = 01xx$ и $CNT\_MODE[1:0] = 10$. Основной счетчик таймера считает по переднему фронту внешнего сигнала, поступающего на вход канала CHyі. Биты CHSEL[1:0] регистра CHy_CNTRL не оказывают влияния, так как они применяются для работы канала таймера только в режиме захвата.

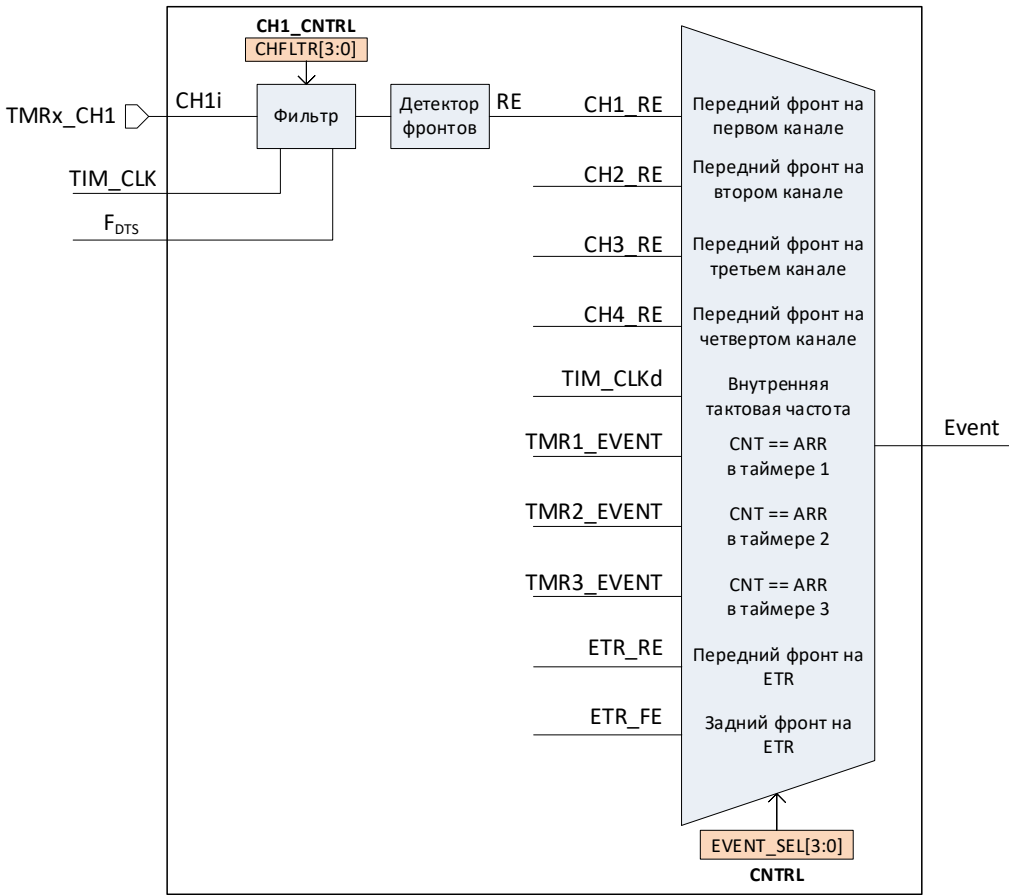


Рисунок 28-13 – Схема тактирования сигналом со входа первого канала

Со входа CHyі внешний тактовый сигнал поступает в блок цифрового фильтра. Данный блок позволяет отфильтровать входной сигнал с целью устранения импульсов, длительность которых меньше заданного порога (см. подраздел 28.7 «Блок цифрового фильтра»). Настройки фильтра задаются в поле CHFLTR[3:0] регистра CHy_CNTRL.

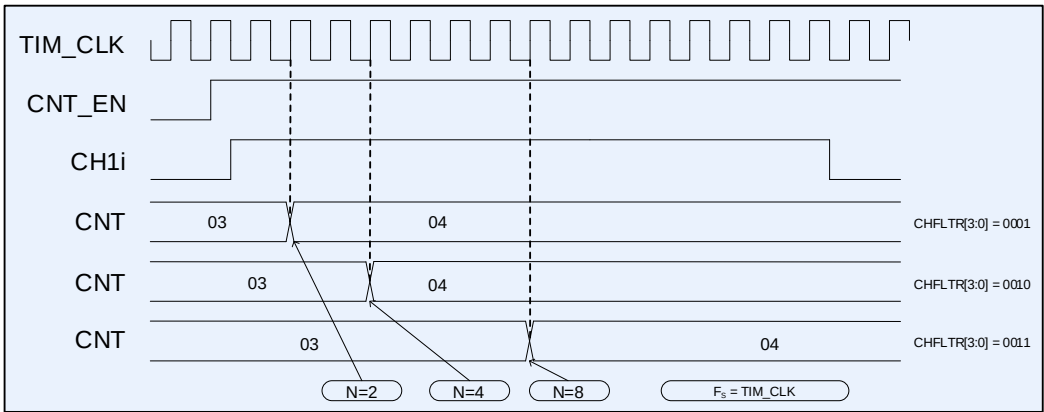


Рисунок 28-14 – Диаграмма внешнего тактирования с разными вариантами фильтра

28.4.4 Внешний тактовый сигнал, «Режим 2»: событие переднего или заднего фронта на входе ETR

Данный режим выбирается, когда $EVENT\_SEL[3:0] = 100x$ и $CNT\_MODE[1:0] = 10$ в регистре CNTRL. Основной счетчик таймера может тактироваться по переднему или по заднему фронту внешнего сигнала, поступающего на вход ETR, в зависимости от значения в поле $EVENT\_SEL[3:0]$.

Конфигурация тактового сигнала со входа ETR задается в регистре BRKETR_CNTRL. Бит ETR_INV позволяет установить инверсию входного сигнала. Поле $ETR\_PSC[1:0]$ задает коэффициент деления асинхронного предделителя внешней частоты (1, 2, 4 или 8). После предделителя тактовый сигнал поступает в блок цифрового фильтра, где он может быть дополнительно отфильтрован с целью устранения импульсов, длительность которых меньше заданного порога (см. подраздел 28.7 «Блок цифрового фильтра»). Настройки фильтра задаются в поле $ETR\_FILTER[3:0]$.

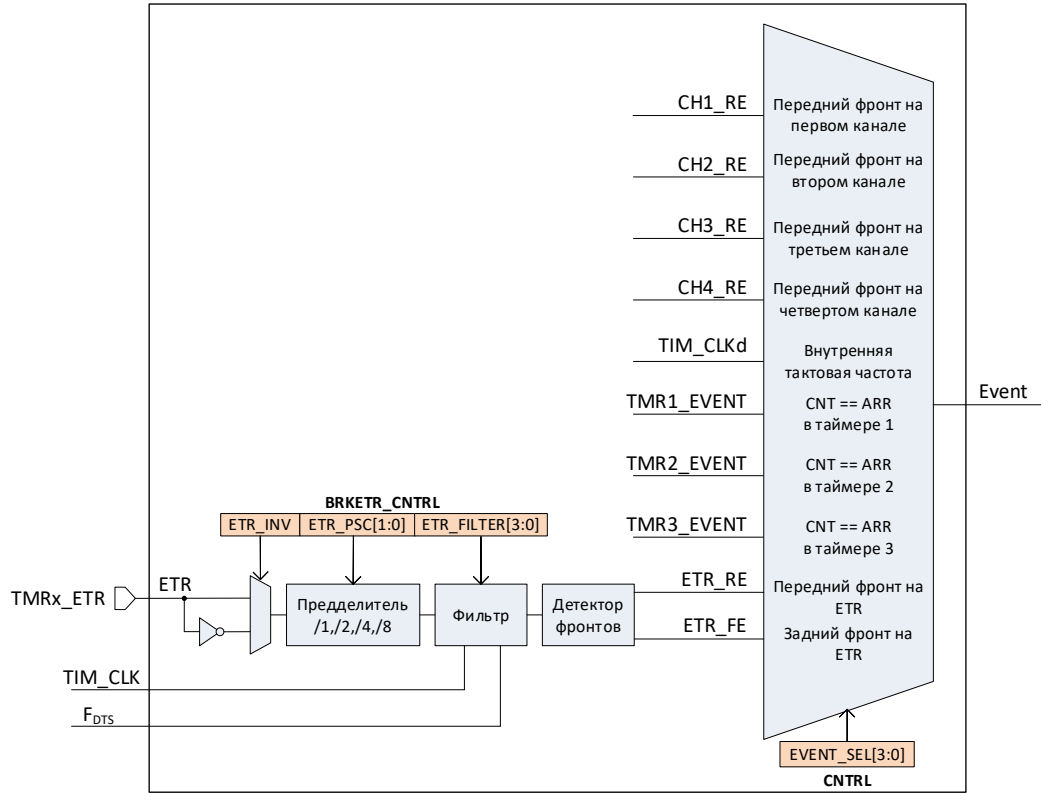


Рисунок 28-15 – Схема тактирования сигналом со входа ETR

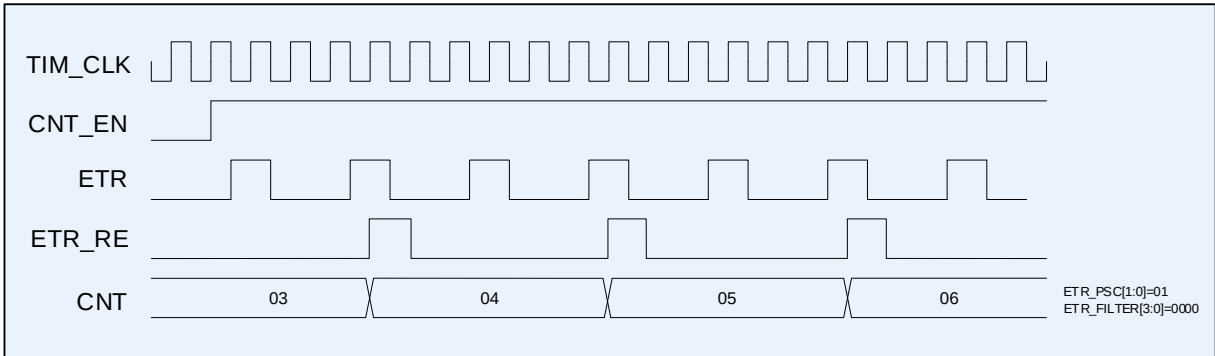


Рисунок 28-16 – Диаграмма тактирования со входа ETR, $EVENT\_SEL[3:0] = 1000$

28.5 Режим захвата

Каждый канал таймера может быть независимо переведен в режим захвата. В режиме захвата по событию от внешнего входного сигнала происходит фиксация значения основного счетчика CNT в регистры CCRy (CCR) и CCRy1 (CCR1). Регистрация событий осуществляется только по входам CHy_i (выводы, настроенные в функцию TMRx_CHy).

Рисунок 28-17 демонстрирует структурную схему блока захвата.

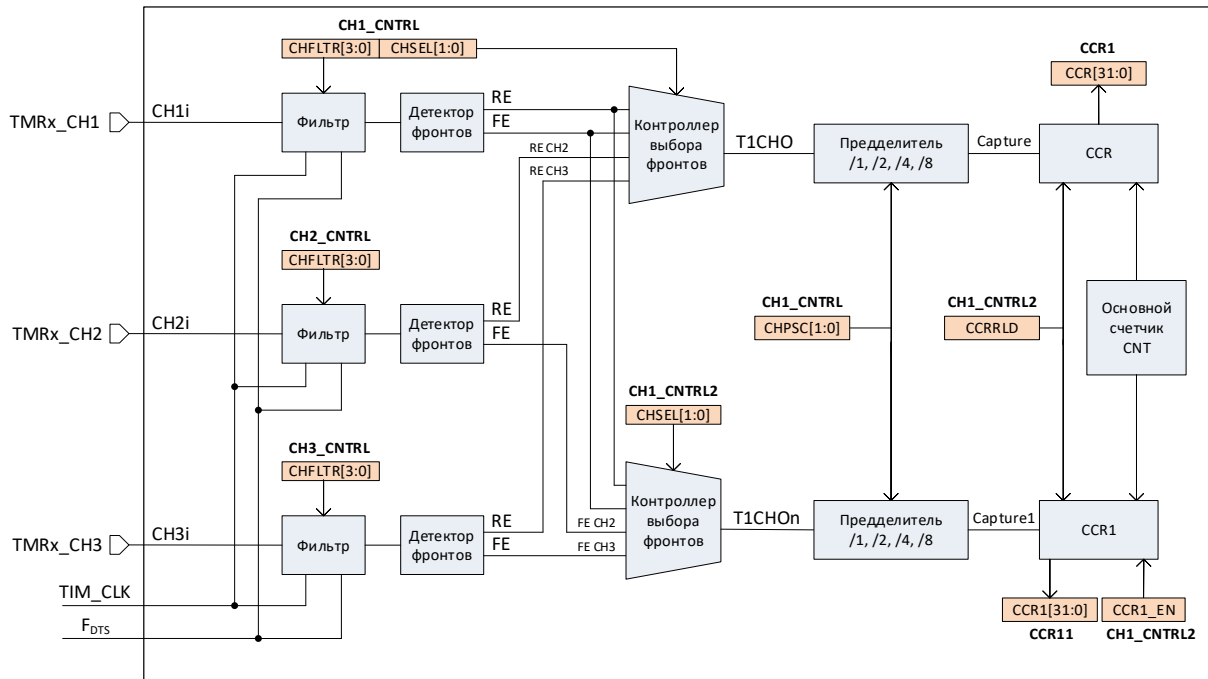


Рисунок 28-17 – Структурная схема блока захвата на примере канала 1

Для включения режима захвата для определенного канала необходимо записать «1» в бит CAP_NPWM регистра управления каналом CHy_CNTRL. Для использования регистра CCRy1(CCR1) необходимо записать «1» в бит CCR1_EN регистра CHy_CNTRL2.

Внешний сигнал со входа CHy_i сначала поступает в блок фильтра. Данный блок позволяет отфильтровать входной сигнал с целью устранения импульсов, длительность которых меньше заданного порога (см. подраздел 28.7 «Блок цифрового фильтра»). Настройки фильтра задаются в поле CHFLTR[3:0] регистра CHy_CNTRL.

Сигнал с блока фильтра поступает в блок «Детектор фронтов». При обнаружении положительного фронта входного сигнала данный блок вырабатывает сигнал RE, а при обнаружении отрицательного фронта входного сигнала – сигнал FE.

В блоке «Контроллер выбора фронтов» производится выбор используемого для захвата сигнала между положительным фронтом канала, отрицательным фронтом канала и положительными и отрицательными фронтами сигналов от других каналов. Настройка блока «Контроллер выбора фронтов» для регистра CCRy осуществляется в поле CHSEL[1:0] регистра CHy_CNTRL, а для регистра CCRy1 – в поле CHSEL[1:0] регистра CHy_CNTRL2. Выбранный для захвата сигнал поступает в предварительный делитель, который в зависимости от значения в поле CHPSC[1:0] регистра CHy_CNTRL позволяет фиксировать все события, либо каждое второе, каждое четвертое или каждое восьмое событие.

Предварительный делитель для регистра CCRy формирует сигнал Capture, а предварительный делитель для регистра CCRy1 формирует сигнал Capture1. По сигналам Capture и Capture1 выполняется запись текущего значения основного счетчика CNT в регистры CCRy и CCRy1.

Рисунок 28-18 показывает пример захвата значения основного счетчика CNT в регистр CCRy по положительному фронту на входе канала, а в регистр CCRy1 – по отрицательному фронту на входе канала.

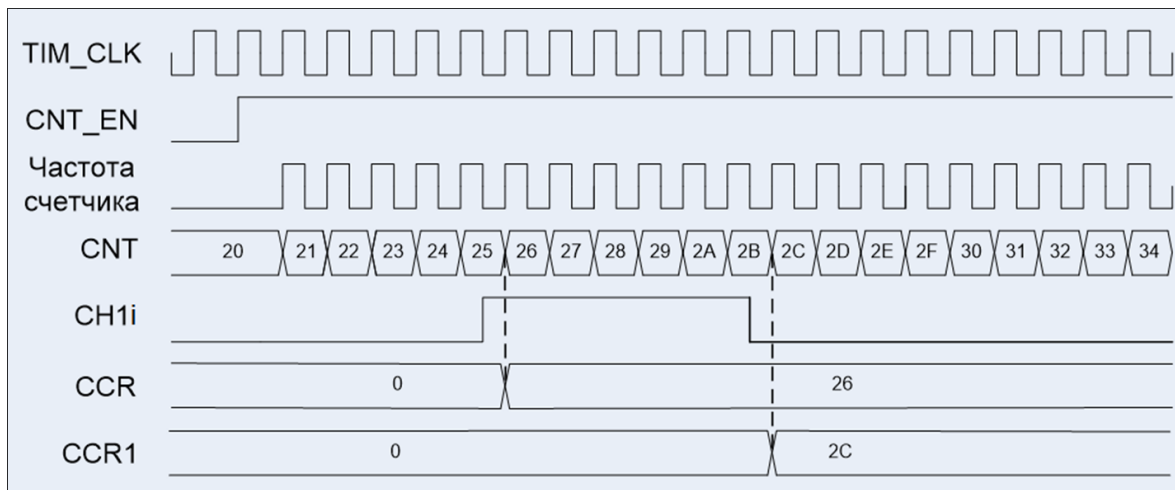


Рисунок 28-18 – Диаграмма захвата события со входа канала 1

По событию захвата на определенном канале в регистре IE можно разрешить выработку прерываний, а в регистре DMA_RE можно разрешить формирование запросов DMA.

28.6 Режим ШИМ

Каждый канал таймера может быть независимо переведен в режим ШИМ для формирования выходных сигналов с возможностью задания «мертвой зоны». Рисунок 28-19 демонстрирует структурную схему блока формирования ШИМ.

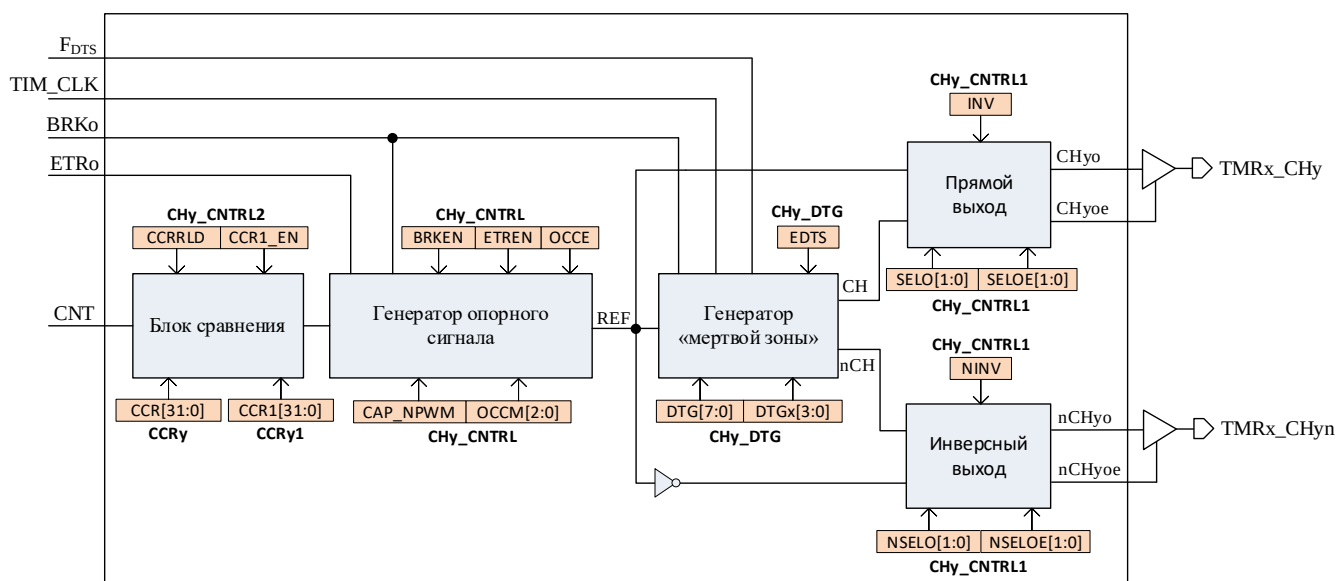


Рисунок 28-19 – Структурная схема блока формирования ШИМ

Для включения режима ШИМ для определенного канала необходимо в регистре управления каналом CHy_CNTRL записать «0» в бит CAP_NPWM.

28.6.1 Генератор опорного сигнала REF

При работе в режиме ШИМ блок генератора опорного сигнала формирует сигнал REF. Данный сигнал формируется на основании сравнения значения в регистрах CCRy (CCR), CCRy1 (CCR1) и основного счетчика CNT. Формат выработки сигнала REF устанавливается в поле OCCM[2:0] регистра управления каналом таймера CHy_CNTRL.

Если в регистре CHy_CNTRL2 бит CCR1_EN = 0, то для формирования сигнала REF используется только результат сравнения значения в регистре CCRy и основного счетчика CNT.

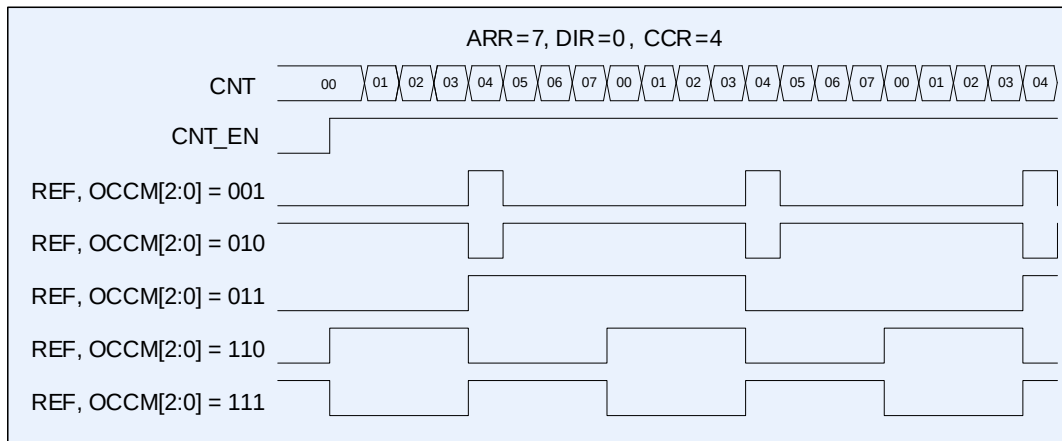


Рисунок 28-20 – Диаграмма работы в режиме ШИМ, CCR1_EN=0

Если в регистре CHy_CNTRL2 бит CCR1_EN = 1, то для формирования сигнала REF задействуются оба результата сравнения значения в регистрах CCRy, CCRy1 и основного счетчика CNT.

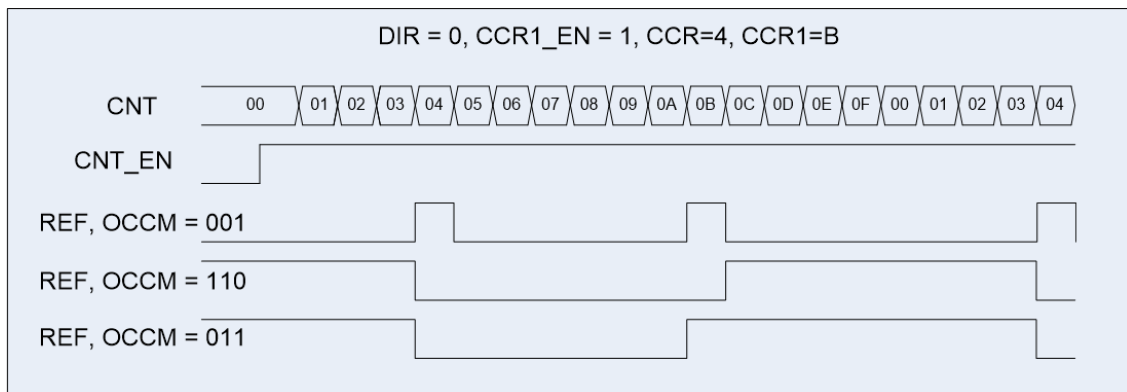


Рисунок 28-21 – Диаграмма работы в режиме ШИМ, CCR1_EN = 1

Запись новых значений в регистры CCRy и CCRy1 осуществляется немедленно, если в регистре CHy_CNTRL2 бит CCRRLD установлен в «0», иначе регистры CCRy и CCRy1 получают новые значения только при CNT == 0. Процесс обновления значений в регистрах CCRy и CCRy1 обозначается в регистре CHy_CNTRL с помощью флагов WR_CMPL и WR_CMPL1, соответственно. На время выполнения записи флаг WR_CMPL/WR_CMPL1 устанавливается в «1», по окончании записи флаг WR_CMPL/WR_CMPL1 сбрасывается в «0».

Сигнал REF может быть принудительно установлен в «0» с использованием внешнего сигнала сброса, поступающего со входа ETR (высокий активный уровень) или

со входа BRK (низкий активный уровень). Активный уровень на входах ETR и BRK может быть изменен с помощью инверсии входного сигнала, регистр BRKETR_CNTRL, биты ETR_INV и BRK_INV, соответственно.

Для разрешения сброса сигнала REF по входу ETR необходимо установить биты ETREN и OCCE в регистре CHy_CNTRL. Активный уровень на входе ETR сбрасывает сигнал REF в «0». После снятия активного уровня на входе ETR сигнал REF остается в «0» до следующего события установки REF в «1» (Рисунок 28-22).

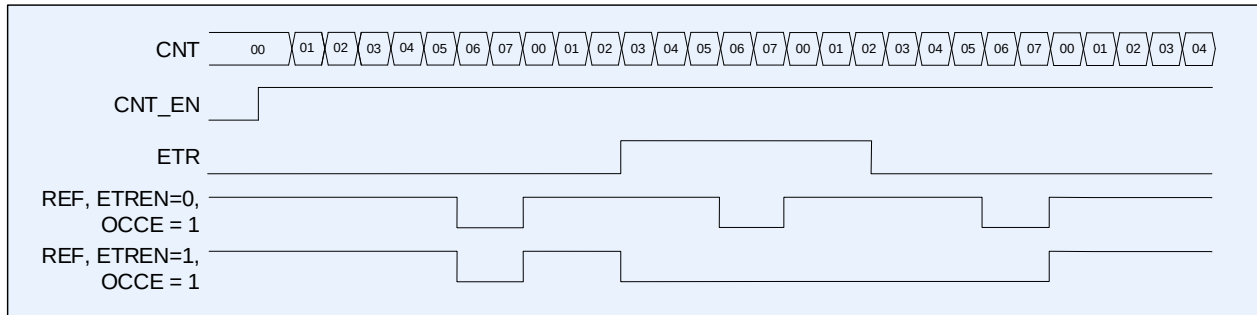


Рисунок 28-22 – Диаграмма сброса сигнала REF по выводу ETR

Для разрешения сброса сигнала REF по входу BRK необходимо установить бит BRKEN в регистре CHy_CNTRL. Активный уровень на входе BRK сбрасывает сигнал REF в «0» путем маскирования. После снятия активного уровня на входе BRK генерация сигнала REF сразу же восстанавливается (Рисунок 28-23).

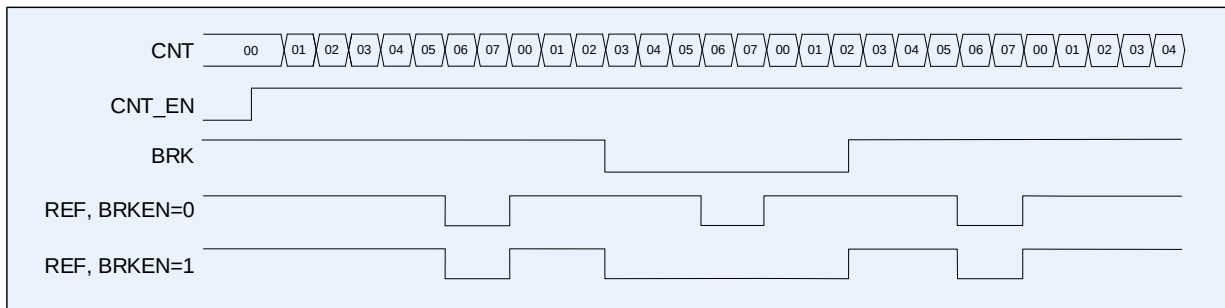


Рисунок 28-23 – Диаграмма сброса сигнала REF по выводу BRK

28.6.2 Генератор «мертвой зоны»

Блок генератора «мертвой зоны» (dead-time generator, DTG) позволяет на основе сигнала REF формировать комплементарную пару сигналов с «мертвой зоной». Выходные сигналы с блока DTG передаются на выходные блоки следующим образом:

- сигнал на прямом выходе (CHyo, CHyoe) представляет собой инвертированный сигнал REF, в котором передний фронт задержан на величину DTGdel относительно заднего фронта опорного сигнала REF;
- сигнал на инверсном выходе (nCHyo, nCHyoe) представляет собой сигнал REF, в котором передний фронт задержан на величину DTGdel относительно переднего фронта опорного сигнала REF.

Значение «мертвой зоны» между сигналами на прямом и инверсном выходах рассчитывается в тактах частоты TIM_CLK или F_{DTs} по формуле

$$DTG_{del} = DTG \cdot (DTG_x + 1), \quad (2)$$

где DTG_x – предварительный делитель частоты;

DTG – основной делитель частоты.

Управление блоком DTG осуществляется через регистр CHy_DTG . Выбор источника тактирования для задания «мертвой зоны» задается битом $EDTS$. Значения делителей DTG_x и DTG задаются в полях $DTG_x[3:0]$ и $DTG[7:0]$, соответственно. Если задержка DTG_{del} больше ширины импульса высокого уровня, то соответствующий импульс не генерируется.

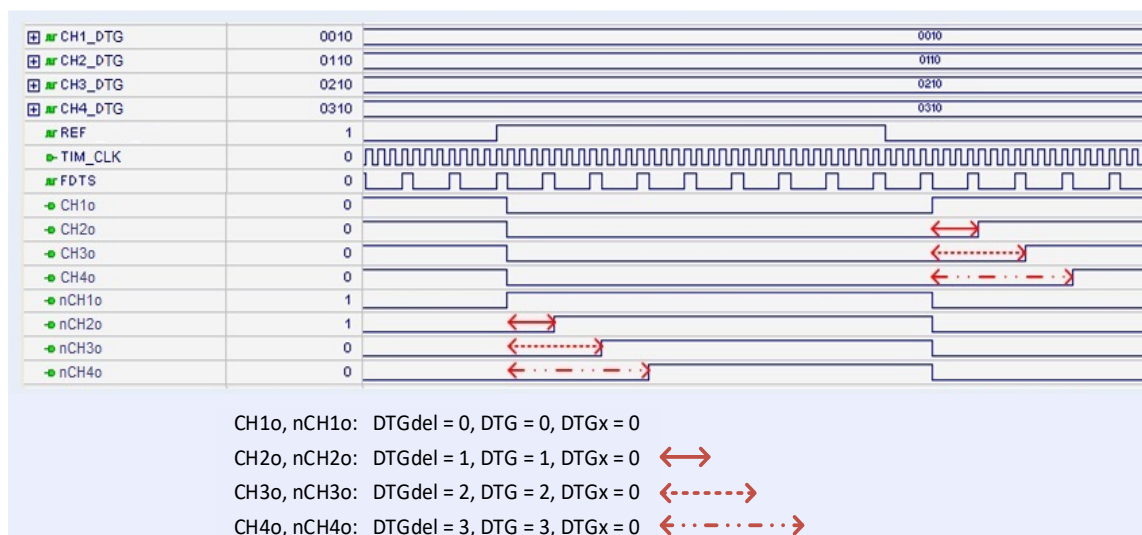


Рисунок 28-24 – Диаграмма работы блока DTG

Выходные сигналы блока DTG могут быть принудительно установлены в «0» с использованием внешнего сигнала сброса, поступающего со входа BRK (низкий активный уровень). Активный уровень на входе BRK может быть изменен с помощью инверсии входного сигнала, бит BRK_INV в регистре $BRKETR\_CNTRL$. Для разрешения сброса выходных сигналов блока DTG по входу BRK необходимо установить бит $BRKEN$ в регистре CHy_CNTRL . Активный уровень на входе BRK сбрасывает выходные сигналы блока DTG в «0» путем их маскирования.

28.6.3 Выходные блоки

Каждый канал таймера имеет два выходных блока – прямой и инверсный. Каждый выходной блок формирует как сигнал выдачи (CHy_o , $nCHy_o$), так и сигнал разрешения выдачи (CHy_{oe} , $nCHy_{oe}$). В качестве сигналов для прямого (CHy_o , CHy_{oe}) и инверсного ($nCHy_o$, $nCHy_{oe}$) выходов в регистре CHy_CNTRL1 могут быть заданы либо постоянные уровни (0 или 1), либо сигналы, формируемые на основе сигнала REF . К таким сигналам относится сам сигнал REF , а также сигналы, формируемые блоком DTG .

Выбор источника сигнала выдачи для прямого (CHy_o) и инверсного ($nCHy_o$) выходов задается в полях $SELO[1:0]$ и $NSELO[1:0]$ регистра CHy_CNTRL1 . Дополнительно каждый сигнал выдачи для прямого (CHy_o) и инверсного ($nCHy_o$) выходов может быть инвертирован путем установки битов INV и $NINV$ в регистре CHy_CNTRL1 . Выбор источника сигнала разрешения выдачи для прямого (CHy_{oe}) и инверсного ($nCHy_{oe}$) выходов задается в полях $SELOE[1:0]$ и $NSELOE[1:0]$

регистра CHy_CNTRL1 . При этом, если сигнал разрешения выдачи равен «0», то соответствующий вывод работает в режиме входа, если сигнал разрешения выдачи равен «1» – то в режиме выхода.

Каждый канал таймера имеет два выходных блока – прямой и инверсный. Каждый выходной блок формирует как сигнал выдачи (CHy_o , $nCHy_o$), так и сигнал разрешения выдачи (CHy_{oe} , $nCHy_{oe}$). В качестве сигналов для прямого (CHy_o , CHy_{oe}) и инверсного ($nCHy_o$, $nCHy_{oe}$) выходов в регистре CHy_CNTRL1 могут быть заданы либо постоянные уровни (0 или 1), либо сигналы, формируемые на основе сигнала REF. К таким сигналам относится сам сигнал REF, а также сигналы, формируемые блоком DTG.

Выбор источника сигнала выдачи для прямого (CHy_o) и инверсного ($nCHy_o$) выходов задается в полях $SELO[1:0]$ и $NSELO[1:0]$ регистра CHy_CNTRL1 . Дополнительно каждый сигнал выдачи для прямого (CHy_o) и инверсного ($nCHy_o$) выходов может быть инвертирован путем установки битов INV и NINV в регистре CHy_CNTRL1 . Выбор источника сигнала разрешения выдачи для прямого (CHy_{oe}) и инверсного ($nCHy_{oe}$) выходов задается в полях $SELOE[1:0]$ и $NSELOE[1:0]$ регистра CHy_CNTRL1 . При этом, если сигнал разрешения выдачи равен «0», то соответствующий вывод работает в режиме входа, если сигнал разрешения выдачи равен «1» – то в режиме выхода.

28.7 Блок цифрового фильтра

В тракте входа ETR и входов каналов таймера CHy_i предусмотрен блок цифрового фильтра, который позволяет исключить из входного сигнала импульсы высокого и низкого уровня, длительность которых меньше заданного порога.

Конфигурация фильтра для входа ETR выполняется в поле $ETR\_FILTER[3:0]$ регистра $BRKETR\_CNTRL$, для входов каналов CHy_i – в поле $CHFLTR[3:0]$ регистра CHy_CNTRL . Значение в данных полях позволяет настроить два параметра фильтра:

- частота выборки F_s , на которой входной сигнал захватывается в сдвиговый регистр для накопления. В качестве частоты F_s может использоваться частота TIM_CLK или F_{DTS} ;
- количество выборок (длина фильтра) N , на протяжении которых входной сигнал должен оставаться стабильным, чтобы не подвергнуться фильтрации.

Если в течение заданного количества выборок N на частоте F_s входной сигнал не изменяется, то значение входного сигнала передается на выход фильтра. Иначе внутренний счетчик накопления сбрасывается и захват сигнала начинается заново.

Таким образом, настраивая частоту F_s и количество выборок N , задается минимальная длительность импульсов входного сигнала, которые не будут отфильтрованы. Рисунок 28-25 демонстрирует диаграмму работы фильтра при использовании частоты TIM_CLK , частота $F_s = TIM\_CLK$, количество выборок $N = 4$.

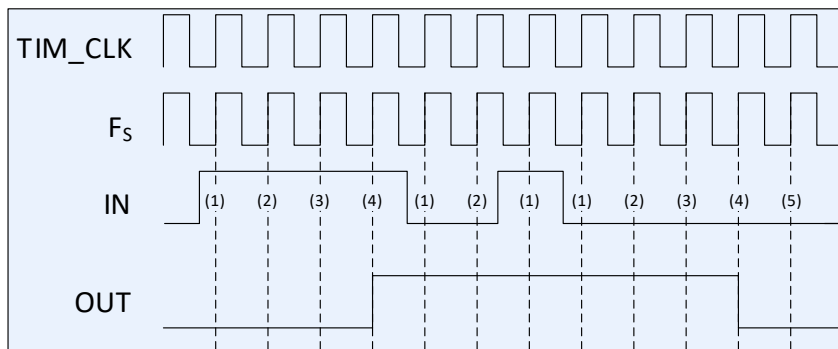


Рисунок 28-25 – Диаграмма работы фильтра, $F_s = \text{TIM\_CLK}$, $N = 4$

Для задания длительных интервалов накопления входного сигнала имеется возможность использовать частоту F_{DTS} , которая формируется из частоты TIM_CLK путем прореживания на заданный коэффициент (см. пункт 28.3.5 «Тактовая частота F_{DTS} »).

Рисунок 28-26 демонстрирует диаграмму работы фильтра при использовании частоты F_{DTS} , частота выборки $F_s = F_{DTS}/2$, количество выборок $N = 6$.

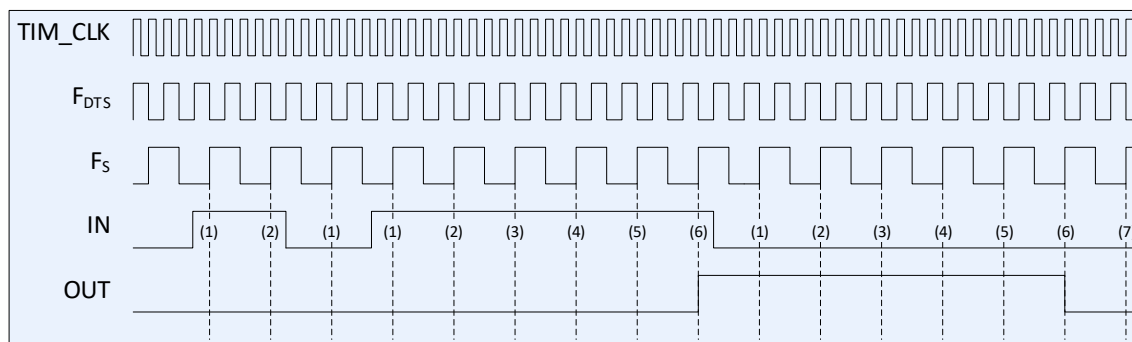


Рисунок 28-26 – Диаграмма работы фильтра, $F_s = F_{DTS}/2$, $N = 6$

Таблица 28-1 показывает возможные варианты настройки блока фильтра.

Таблица 28-1 – Возможные конфигурации фильтра

CHFLTR[3:0], ETR_FILTER[3:0]	Частота выборки F_s	Количество выборок N	Минимальная длительность импульсов, которые не будут отфильтрованы
0000	F_{DTS}	1	-
0001	TIM_CLK	2	$2 \times T_{\text{TIM\_CLK}}$
0010	TIM_CLK	4	$4 \times T_{\text{TIM\_CLK}}$
0011	TIM_CLK	8	$8 \times T_{\text{TIM\_CLK}}$
0100	$F_{DTS}/2$	6	$12 \times T_{F_{DTS}}$
0101	$F_{DTS}/2$	8	$16 \times T_{F_{DTS}}$
0110	$F_{DTS}/4$	6	$24 \times T_{F_{DTS}}$
0111	$F_{DTS}/4$	8	$32 \times T_{F_{DTS}}$
1000	$F_{DTS}/8$	6	$48 \times T_{F_{DTS}}$
1001	$F_{DTS}/8$	8	$64 \times T_{F_{DTS}}$
1010	$F_{DTS}/16$	5	$80 \times T_{F_{DTS}}$
1011	$F_{DTS}/16$	6	$96 \times T_{F_{DTS}}$
1100	$F_{DTS}/16$	8	$128 \times T_{F_{DTS}}$
1101	$F_{DTS}/32$	5	$160 \times T_{F_{DTS}}$
1110	$F_{DTS}/32$	6	$192 \times T_{F_{DTS}}$
1111	$F_{DTS}/32$	8	$256 \times T_{F_{DTS}}$

28.8 Флаги состояний, прерывания и запросы DMA

В процессе работы блок таймера отслеживает состояние внутренних блоков и формирует 17 событий:

- CNT ZERO EVENT – совпадение значения счетчика CNT с нулем;
- CNT ARR EVENT – совпадение значения счетчика CNT со значением в регистре ARR;
- ETR RE EVENT – фиксация переднего фронта на входе ETR;
- ETR FE EVENT – фиксация заднего фронта на входе ETR;
- BRK EVENT – фиксация высокого уровня на входе BRK;
- CCR CAP EVENT[3:0] – запись значения счетчика CNT в регистр CCRy по захвату настроенного фронта на входе канала CHy, события формируются индивидуально для каждого канала;
- CCR REF EVENT[3:0] – фиксация переднего фронта на выходе генератора опорного сигнала REF, события формируются индивидуально для каждого канала;
- CCR CAP1 EVENT[3:0] – запись значения счетчика CNT в регистр CCRy1 по захвату настроенного фронта на входе канала CHy, события формируются индивидуально для каждого канала.

28.8.1 Флаги состояний

При возникновении события устанавливается соответствующий флаг в регистре STATUS. Сброс флагов в регистре STATUS осуществляется записью «0», запись «1» не оказывает влияния. Если запись «0» выполняется одновременно с новым событием, то приоритет у нового события.

28.8.2 Прерывания

Блок таймера на основе флагов в регистре STATUS формирует один общий сигнал запроса прерывания INT_TMRx. Выбор флагов, формирующих запрос прерывания, осуществляется через регистр разрешения прерываний IE. При формировании запроса прерывания маскированные состояния флагов из регистра STATUS объединяются по схеме ИЛИ.

28.8.3 Запросы DMA

На основе отслеживаемых событий блок таймера также формирует сигнал запроса DMA TMRx_DMA_REQ. Выбор событий, формирующих запрос DMA, осуществляется через регистр DMA_RE.

28.9 Примеры

В данном разделе приведены примеры инициализации таймера 1 в различных режимах работы. Для других таймеров инициализация выполняется аналогично.

28.9.1 Обычный счетчик

```
MDR_RST_CLK->PER_CLOCK |= 0x4000; //Разрешение частоты PCLK для таймера 1
MDR_RST_CLK->TIM_CLOCK = 0x01000000; //Настройка и подача частоты TIM_CLK
//для таймера 1 (TIM_CLK = HCLK)

MDR_TIMER1->CNTRL = 0x00000000; //Режим инициализации таймера
//Настраиваем работу основного счетчика
MDR_TIMER1->CNT = 0x00000000; //Начальное значение счетчика
MDR_TIMER1->PSG = 0x00000000; //Предделитель частоты TIM_CLK
MDR_TIMER1->ARR = 0x0000000F; //Основание счета
MDR_TIMER1->IE = 0x00000002; //Разрешение генерировать прерывание при CNT = ARR
//Разрешение работы таймера
MDR_TIMER1->CNTRL = 0x00000001; //Счет прямой по TIM_CLK
```

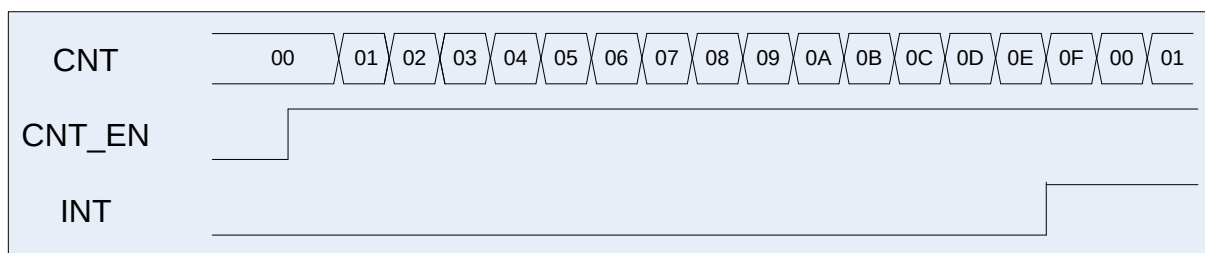


Рисунок 28-27 – Режим обычного счетчика

28.9.2 Режим захвата

```
MDR_RST_CLK->PER_CLOCK |= 0x4000; //Разрешение частоты PCLK для таймера 1
MDR_RST_CLK->TIM_CLOCK = 0x01000000; //Настройка и подача частоты TIM_CLK
//для таймера 1 (TIM_CLK = HCLK)

MDR_TIMER1->CNTRL = 0x00000000; //Режим инициализации таймера
//Настраиваем работу основного счетчика
MDR_TIMER1->CNT = 0x00000000; //Начальное значение счетчика
MDR_TIMER1->PSG = 0x00000000; //Предделитель частоты TIM_CLK
MDR_TIMER1->ARR = 0x000000FF; //Основание счета
MDR_TIMER1->IE = 0x000001E0; //Разрешение генерировать прерывание по событию
//настроенного фронта на входах CH1i-CH4i

MDR_TIMER1->CH1_CNTRL = 0x00008000; //Захват по положительному фронту
//сигнала на входе CH1i, фильтрация
//отключена
MDR_TIMER1->CH2_CNTRL = 0x00008010; //Захват по отрицательному фронту
//сигнала на входе CH2i, фильтрация
//отключена
MDR_TIMER1->CH3_CNTRL = 0x00008001; //Захват по положительному фронту
//сигнала на входе CH3i, фильтрация
//выполняется по двум выборкам на
//частоте TIM_CLK
MDR_TIMER1->CH4_CNTRL = 0x00008011; //Захват по отрицательному фронту
//сигнала на входе CH4i, фильтрация
//выполняется по двум выборкам на
//частоте TIM_CLK
```

//Режим работы выхода канала – канал работает на вход

MDR_TIMER1->CH1_CNTRL1= 0x00000000;

MDR_TIMER1->CH2_CNTRL1= 0x00000000;

MDR_TIMER1->CH3_CNTRL1= 0x00000000;

MDR_TIMER1->CH4_CNTRL1= 0x00000000;

//Разрешение работы таймера

MDR_TIMER1->CNTRL = 0x00000001; //Счет прямой по TIM_CLK

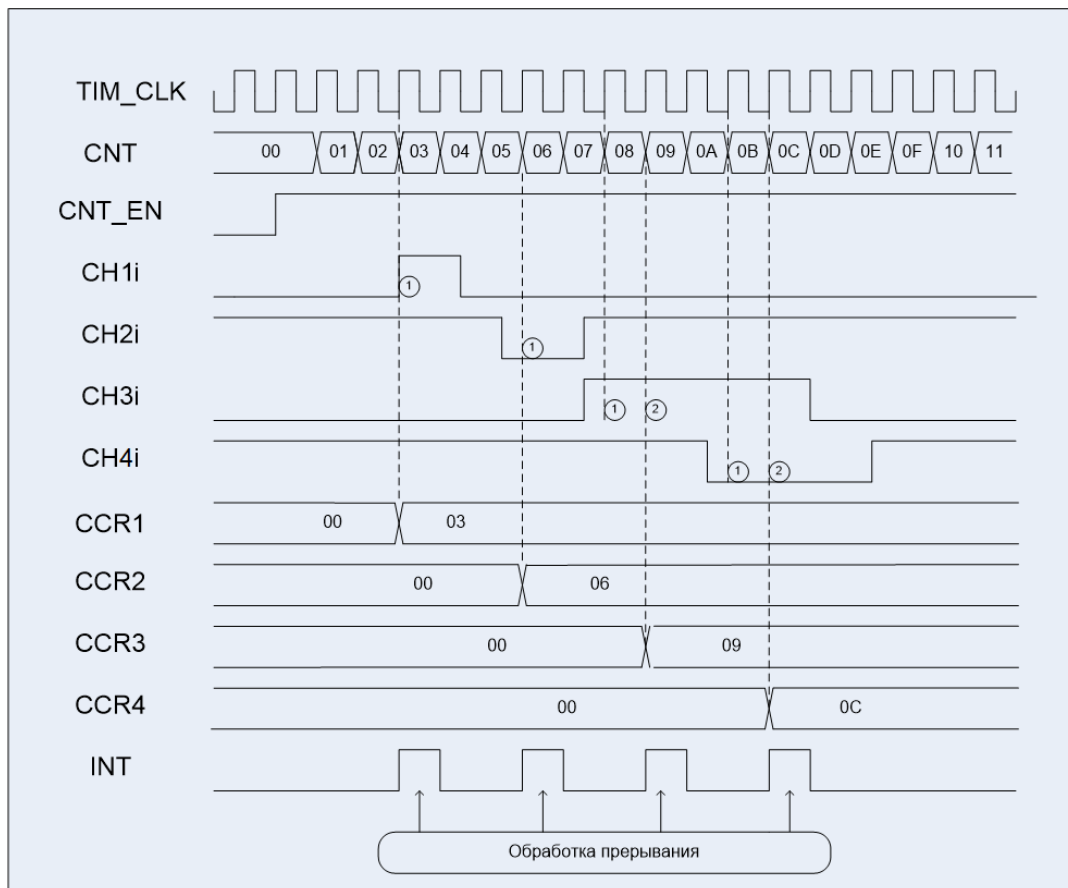


Рисунок 28-28 – Диаграммы примера работы в режиме захвата

28.9.3 Режим ШИМ

MDR_RST_CLK->PER_CLOCK |= 0x4000; //Разрешение частоты PCLK для таймера 1

MDR_RST_CLK->TIM_CLOCK = 0x01000000; //Настройка и подача частоты TIM_CLK

//для таймера 1 (TIM_CLK = HCLK)

MDR_TIMER1->CNTRL = 0x00000000; //Режим инициализации таймера

//Настраиваем работу основного счетчика

MDR_TIMER1->CNT = 0x00000000; //Начальное значение счетчика

MDR_TIMER1->PSG = 0x00000000; //Предделитель частоты TIM_CLK

MDR_TIMER1->ARR = 0x00000010; //Основание счета

MDR_TIMER1->IE = 0x00001E00; //Разрешение генерировать прерывание по
//событию переднего фронта на выходе REF для
//всех каналов

//Режим работы каналов – ШИМ

MDR_TIMER1->CH1_CNTRL = 0x00000200; //REF = 1, если CNT == CCR

MDR_TIMER1->CH2_CNTRL = 0x00000200; //REF = 1, если CNT == CCR

MDR_TIMER1->CH3_CNTRL = 0x00000400; //REF = 0, если CNT == CCR

MDR_TIMER1->CH4_CNTRL = 0x00000600; //Переключение REF, если CNT == CCR

```
//Режим работы выхода канала – канал работает на выход,
//на выходы канала выдается сигнал REF
MDR_TIMER1->CH1_CNTRL1= 0x00000909;
MDR_TIMER1->CH2_CNTRL1= 0x00000909;
MDR_TIMER1->CH3_CNTRL1= 0x00000909;
MDR_TIMER1->CH4_CNTRL1= 0x00000909;
//Установка значений CCR, с которыми сравнивается CNT при работе в режиме ШИМ
MDR_TIMER1->CCR1 = 0x00000003;
MDR_TIMER1->CCR2 = 0x00000006;
MDR_TIMER1->CCR3 = 0x00000009;
MDR_TIMER1->CCR4 = 0x0000000F;
//Разрешение работы таймера
MDR_TIMER1->CNTRL = 0x00000001; //Счет прямой по TIM_CLK
```

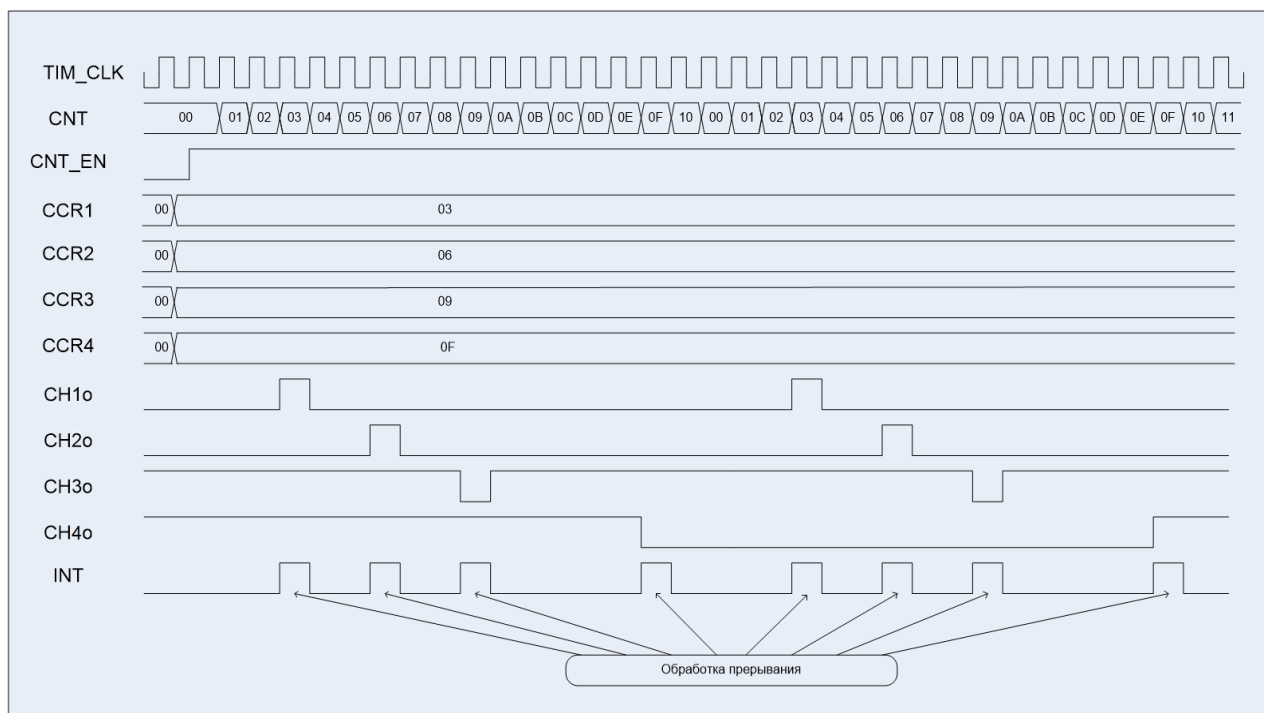


Рисунок 28-29 – Диаграммы примера работы в режиме ШИМ

28.10 Описание регистров блока таймера

Таблица 28-2 – Базовые адреса и смещения регистров управления таймера

Адрес	Название	Описание
0x4007_0000	MDR_TIMER1	Контроллер TIMER1
0x4007_8000	MDR_TIMER2	Контроллер TIMER2
0x4008_0000	MDR_TIMER3	Контроллер TIMER3
Смещение		
0x00	CNT[15:0]	Основной счетчик таймера MDR_TIMERx->CNT
0x04	PSG[15:0]	Делитель частоты TIM_CLK для тактирования основного счетчика MDR_TIMERx->PSG
0x08	ARR[15:0]	Основание счета основного счетчика MDR_TIMERx->ARR

Адрес	Название	Описание
0x0C	CNTRL[31:0]	Регистр управления основным счетчиком MDR_TIMERx->CNTRL
0x50	BRKETR_CNTRL[31:0]	Регистр управления входом BRK и ETR MDR_TIMERx->BRKETR_CNTRL
0x54	STATUS[31:0]	Регистр статуса таймера MDR_TIMERx->STATUS
0x58	IE[31:0]	Регистр разрешения прерываний таймера MDR_TIMERx->IE
0x5C	DMA_RE[31:0]	Регистр разрешения формирования запроса DMA MDR_TIMERx->DMA_RE
Канал 1		
0x10	CCR1[15:0]	Регистр сравнения/захвата для 1 канала таймера MDR_TIMERx->CCRy
0x20	CH1_CNTRL[31:0]	Регистр управления для 1 канала таймера MDR_TIMERx->CHy_CNTRL
0x30	CH1_CNTRL1[31:0]	Регистр управления 1 для 1 канала таймера MDR_TIMERx->CHy_CNTRL1
0x40	CH1_DTG[31:0]	Регистр управления DTG для 1 канала таймера MDR_TIMERx->CHy_DTG
0x60	CH1_CNTRL2[31:0]	Регистр управления 2 для 1 канала таймера MDR_TIMERx->CHy_CNTRL2
0x70	CCR11[15:0]	Регистр сравнения/захвата 1 для 1 канала таймера MDR_TIMERx->CCRy1
Канал 2		
0x14	CCR2[15:0]	Регистр сравнения/захвата для 2 канала таймера MDR_TIMERx->CCRy
0x24	CH2_CNTRL[31:0]	Регистр управления для 2 канала таймера MDR_TIMERx->CHy_CNTRL
0x34	CH2_CNTRL1[31:0]	Регистр управления 1 для 2 канала таймера MDR_TIMERx->CHy_CNTRL1
0x44	CH2_DTG[31:0]	Регистр управления DTG для 2 канала таймера MDR_TIMERx->CHy_DTG
0x64	CH2_CNTRL2[31:0]	Регистр управления 2 для 2 канала таймера MDR_TIMERx->CHy_CNTRL2
0x74	CCR21[15:0]	Регистр сравнения/захвата 1 для 2 канала таймера MDR_TIMERx->CCRy1
Канал 3		
0x18	CCR3[15:0]	Регистр сравнения/захвата для 3 канала таймера MDR_TIMERx->CCRy
0x28	CH3_CNTRL[31:0]	Регистр управления для 3 канала таймера MDR_TIMERx->CHy_CNTRL
0x38	CH3_CNTRL1[31:0]	Регистр управления 1 для 3 канала таймера MDR_TIMERx->CHy_CNTRL1
0x48	CH3_DTG[31:0]	Регистр управления DTG для 3 канала таймера MDR_TIMERx->CHy_DTG

Адрес	Название	Описание
0x68	CH3_CNTRL2[31:0]	Регистр управления 2 для 3 канала таймера MDR_TIMERx->CHy_CNTRL2
0x78	CCR31[15:0]	Регистр сравнения/захвата 1 для 3 канала таймера MDR_TIMERx->CCRy1
Канал 4		
0x1C	CCR4[15:0]	Регистр сравнения/захвата для 4 канала таймера MDR_TIMERx->CCRy
0x2C	CH4_CNTRL[31:0]	Регистр управления для 4 канала таймера MDR_TIMERx->CHy_CNTRL
0x3C	CH4_CNTRL1[31:0]	Регистр управления 1 для 4 канала таймера MDR_TIMERx->CHy_CNTRL1
0x4C	CH4_DTG[31:0]	Регистр управления DTG для 4 канала таймера MDR_TIMERx->CHy_DTG
0x6C	CH4_CNTRL2[31:0]	Регистр управления 2 для 4 канала таймера MDR_TIMERx->CHy_CNTRL2
0x7C	CCR41[15:0]	Регистр сравнения/захвата 1 для 4 канала таймера MDR_TIMERx->CCRy1

28.10.1 MDR_TIMERx->CNT

Таблица 28-3 – Основной счетчик таймера CNT

Номер	15...0
Доступ	R/W
Сброс	0
	CNT[15:0]

Таблица 28-4 – Описание бит регистра CNT

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15...0	CNT[15:0]	Значение основного счетчика таймера

28.10.2 MDR_TIMERx->PSG

Таблица 28-5 – Делитель частоты TIM_CLK для счета основного счетчика PSG

Номер	15..0
Доступ	R/W
Сброс	0
	PSG[15:0]

Таблица 28-6 – Описание бит регистра PSG

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15...0	PSG[15:0]	Значение предварительного делителя счетчика Основной счетчик считает на частоте: $TIM\_CLKd = TIM\_CLK/(PSG+1)$

28.10.3 MDR_TIMERx->ARR

Таблица 28-7 – Основание счета основного счетчика ARR

Номер	15...0
Доступ	R/W
Сброс	0
	ARR[15:0]

Таблица 28-8 – Описание бит регистра ARR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15...0	ARR[15:0]	Основание счета для основного счетчика. CNT = [0...ARR]

28.10.4 MDR_TIMERx->CNTRL

Таблица 28-9 – Регистр управления основным счетчиком CNTRL

Номер	31...12	11...8
Доступ	U	R/W
Сброс	0	0
	-	EVENT_SEL[3:0]

Номер	7, 6	5, 4	3	2	1	0
Доступ	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	00	00	0	0	0	0
	CNT_MODE[1:0]	FDTS[1:0]	DIR	WR_CMPL	ARRB_EN	CNT_EN

Таблица 28-10 – Описание бит регистра CNTRL

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...12	-	Зарезервировано
11...8	EVENT_SEL[3:0]	Биты выбора источника событий: 0000 – внутренняя тактовая частота TIM_CLKd (формируется путем деления частоты TIM_CLK); 0001 – CNT == ARR в таймере 1; 0010 – CNT == ARR в таймере 2; 0011 – CNT == ARR в таймере 3; 0100 – событие переднего фронта на канале 1, «Режим 1»; 0101 – событие переднего фронта на канале 2, «Режим 1»; 0110 – событие переднего фронта на канале 3, «Режим 1»; 0111 – событие переднего фронта на канале 4, «Режим 1»; 1000 – событие переднего фронта на ETR, «Режим 2»; 1001 – событие заднего фронта на ETR, «Режим 2»; 1010-1111 – зарезервировано

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
7, 6	CNT_MODE[1:0]	Режим счета основного счетчика: 00 – счетчик прямой при DIR = 0; счетчик обратный при DIR = 1; 01 – счетчик двунаправленный с автоматическим изменением DIR при CNT == 0 или CNT == ARR; 10 – счетчик прямой при DIR = 0; счетчик обратный при DIR = 1; 11 – зарезервировано. Режим счета CNT_MODE[1:0] необходимо устанавливать в соответствии со значением в поле EVENT_SEL[3:0]: – EVENT_SEL[3:0] = 0000: CNT_MODE[1:0] = 00 или 01; – EVENT_SEL[3:0] != 0000: CNT_MODE[1:0] = 10
5...4	FDTs[1:0]	Делитель тактовой частоты FDTs: 00 – FDTs = TIM_CLK; 01 – FDTs = TIM_CLK/2; 10 – FDTs = TIM_CLK/3; 11 – FDTs = TIM_CLK/4
3	DIR	Направление счета основного счетчика: 0 – прямой, от 0 до ARR; 1 – обратный, от ARR до 0
2	WR_CMPL	Флаг выполнения записи нового значения в регистры CNT, PSG и ARR: 0 – новые данные можно записывать; 1 – данные не записаны и идет запись
1	ARRB_EN	Режим обновления регистра ARR: 0 – ARR будет перезаписан в момент записи в ARR; 1 – ARR будет перезаписан при завершении счета CNT
0	CNT_EN	Разрешение работы таймера: 0 – таймер отключен; 1 – таймер включен

28.10.5 MDR_TIMERx->CCRy

Таблица 28-11– Регистр сравнения/захвата для 'у' канала таймера CCRy

Номер	15...0
Доступ	R/W
Сброс	0
	CCR[15:0]

Таблица 28-12– Описание бит регистра CCRy

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15...0	CCR[15:0]	Режим захвата: значение CNT, при котором произошел факт захвата события. Режим ШИМ: значение CCR, с которым сравнивается CNT

28.10.6 MDR_TIMERx->CCRy1

Таблица 28-13– Регистр сравнения/захвата для ‘у’ канала таймера CCRy1

Номер	15...0
Доступ	R/W
Сброс	0
	CCR1[15:0]

Таблица 28-14– Описание бит регистра CCRy1

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15...0	CCR1[15:0]	Режим захвата: значение CNT, при котором произошел факт захвата события. Режим ШИМ: значение CCR1, с которым сравнивается CNT

28.10.7 MDR_TIMERx->CHy_CNTRL

Таблица 28-15– Регистр управления для ‘у’ канала таймера CHy_CNTRL

Номер	31...17	16	15	14	13
Доступ	U	RO	R/W	RO	R/W
Сброс	0	0	0	0	0
	-	WR_CMPL1	CAP_NPWM	WR_CMPL	ETREN

Номер	31...17	16	15	14	13
Доступ	U	RO	R/W	RO	R/W
Сброс	0	0	0	0	0
	-	WR_CMPL1	CAP_NPWM	WR_CMPL	ETREN

Таблица 28-16– Описание бит регистра CHy_CNTRL

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...17	-	Зарезервировано
16	WR_CMPL1	Флаг выполнения записи нового значения в регистр CCRy1: 0 – новые данные можно записывать; 1 – данные не записаны и идет запись
15	CAP_NPWM	Режим работы канала: 0 – канал работает в режиме ШИМ; 1 – канал работает в режиме захвата
14	WR_CMPL	Флаг выполнения записи нового значения в регистр CCRy: 0 – новые данные можно записывать; 1 – данные не записаны и идет запись
13	ETREN	Разрешение сброса сигнала REF в «0» при высоком уровне на входе ETR: 0 – запрещен; 1 – разрешен

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
12	BRKEN	Разрешение сброса сигналов REF и DTG в «0» при низком уровне на входе BRK: 0 – запрещен; 1 – разрешен
11...9	OCCM[2:0]	Формат выработки сигнала REF в режиме ПИИМ: Если CCR1_EN=0: 000 – всегда 0; 001 – 1, если CNT==CCR; 010 – 0, если CNT==CCR; 011 – переключение REF, если CNT==CCR; 100 – всегда 0; 101 – всегда 1; 110 – 1, если DIR=0 (счет прямой), CNT<CCR, иначе 0; 0, если DIR=1 (счет обратный), CNT>CCR, иначе 1; 111 – 0, если DIR=0 (счет прямой), CNT<CCR, иначе 1; 1, если DIR=1 (счет обратный), CNT>CCR, иначе 0. Если CCR1_EN=1: 000 – всегда 0; 001 – 1, если CNT==CCR или CNT==CCR1; 010 – 0, если CNT==CCR или CNT==CCR1; 011 – переключение REF, если CNT==CCR или CNT==CCR1; 100 – всегда 0; 101 – всегда 1; 110 – 0, если DIR=0 (счет прямой), CCR≤CNT≤CCR1, иначе 1; 0, если DIR=1 (счет обратный), CCR<CNT<CCR1, иначе 1; 111 – 1, если DIR=0 (счет прямой), CCR≤CNT≤CCR1, иначе 0; 1, если DIR=1 (счет обратный), CCR<CNT<CCR1, иначе 0; Необходимо соблюдать условие CCR<CCR1
8	OCCE	Разрешение работы ETR: 0 – запрещен; 1 – разрешен
7...6	CHPSC[1:0]	Предварительный делитель входного канала: 00 – нет деления; 01 – /2; 10 – /4; 11 – /8
5...4	CHSEL[1:0]	Выбор события по входному каналу CHy _i для фиксации значения основного счетчика (регистр CNT) в регистр CCRy: 00 – положительный фронт на входном канале CHy _i ; 01 – отрицательный фронт на входном канале CHy _i ; 10 – положительный фронт от других каналов: для первого канала от второго канала; для второго канала от третьего канала; для третьего канала от четвертого канала; для четвертого канала от первого канала;

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
		11 – положительный фронт от других каналов: для первого канала от третьего канала; для второго канала от четвертого канала; для третьего канала от первого канала; для четвертого канала от второго канала
3...0	CHFLTR[3:0]	Конфигурация фильтра на входе канала 'у'. Выбор частоты выборки F_S и количества выборок N : 0000 – нет фильтрации, $F_S = F_{DTS}$; 0001 – $F_S = TIM\_CLK$, $N = 2$; 0010 – $F_S = TIM\_CLK$, $N = 4$; 0011 – $F_S = TIM\_CLK$, $N = 8$; 0100 – $F_S = F_{DTS}/2$, $N = 6$; 0101 – $F_S = F_{DTS}/2$, $N = 8$; 0110 – $F_S = F_{DTS}/4$, $N = 6$; 0111 – $F_S = F_{DTS}/4$, $N = 8$; 1000 – $F_S = F_{DTS}/8$, $N = 6$; 1001 – $F_S = F_{DTS}/8$, $N = 8$; 1010 – $F_S = F_{DTS}/16$, $N = 5$; 1011 – $F_S = F_{DTS}/16$, $N = 6$; 1100 – $F_S = F_{DTS}/16$, $N = 8$; 1101 – $F_S = F_{DTS}/32$, $N = 5$; 1110 – $F_S = F_{DTS}/32$, $N = 6$; 1111 – $F_S = F_{DTS}/32$, $N = 8$

28.10.8 MDR_TIMERx->CHy_CNTRL1

Таблица 28-17– Регистр управления 1 для 'у' канала таймера CHy_CNTRL1

Номер	31...13	12	11...10	9...8	7...5	4	3...2	1...0
Доступ	U	R/W	R/W	R/W	U	R/W	R/W	R/W
Сброс	0	0	00	00	0	0	00	00
	-	NINV	NSELO [1:0]	NSELOE [1:0]	-	INV	SELO [1:0]	SELOE [1:0]

Таблица 28-18– Описание бит регистра CHy_CNTRL1

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...13	-	Зарезервировано
12	NINV	Инверсия инверсного выхода nCHy: 0 – выход не инвертируется; 1 – выход инвертируется
11...10	NSELO[1:0]	Выбор источника сигнала для инверсного выхода nCHy: 00 – на nCHyо выдается 0; 01 – на nCHyо выдается 1; 10 – на nCHyо выдается сигнал nREF; 11 – на nCHyо выдается сигнал с DTG

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
9...8	NSELOE[1:0]	Режим работы инверсного выхода nCHy: 00 – на nCHyое выдается 0; 01 – на nCHyое выдается 1; 10 – на nCHyое выдается сигнал nREF; 11 – на nCHyое выдается сигнал с DTG. При nCHyое = 0 вывод канала в третьем состоянии, при nCHyое = 1 вывод канала работает в режиме выхода
7...5	-	Зарезервировано
4	INV	Инверсия прямого выхода CHy: 0 – выход не инвертируется; 1 – выход инвертируется
3...2	SELO[1:0]	Выбор источника сигнала для прямого выхода CHy: 00 – на CHyоо выдается 0; 01 – на CHyоо выдается 1; 10 – на CHyоо выдается сигнал REF; 11 – на CHyоо выдается сигнал с DTG
1...0	SELOE[1:0]	Режим работы прямого выхода CHy: 00 – на CHyое выдается 0; 01 – на CHyое выдается 1; 10 – на CHyое выдается сигнал REF; 11 – на CHyое выдается сигнал с DTG. При CHyое = 0 вывод канала работает в режиме входа, при CHyое = 1 вывод канала работает в режиме выхода

28.10.9 MDR_TIMERx->CHy_CNTRL2

Таблица 28-19– Регистр управления 2 для ‘у’ канала таймера CHy_CNTRL2

Номер	31...4	3	2	1...0
Доступ	U	R/W	R/W	R/W
Сброс	0	0	0	00
	-	CCRRLD	CCR1_EN	CHSEL[1:0]

Таблица 28-20– Описание бит регистра CHy_CNTRL2

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...4	-	Зарезервировано
3	CCRRLD	Режим обновления регистров CCRy и CCRy1: 0 – обновление возможно в любой момент времени; 1 – обновление будет осуществлено только при CNT == 0
2	CCR1_EN	Разрешение работы регистра CCRy1: 0 – CCRy1 не используется; 1 – CCRy1 используется

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
1...0	CHSEL[1:0]	Выбор события по входному каналу CHy _i для фиксации значения основного счетчика (регистр CNT) в регистр CCRy1: 00 – положительный фронт на входном канале CHy _i ; 01 – отрицательный фронт на входном канале CHy _i ; 10 – отрицательный фронт от других каналов: для первого канала от второго канала; для второго канала от третьего канала; для третьего канала от четвертого канала; для четвертого канала от первого канала; 11 – отрицательный фронт от других каналов: для первого канала от третьего канала; для второго канала от четвертого канала; для третьего канала от первого канала; для четвертого канала от второго канала

28.10.10 MDR_TIMERx->CHy_DTG

Таблица 28-21– Регистр управления генератором «мертвой зоны» CHy_DTG

Номер	31...16	15...8	7...5	4	3...0
Доступ	U	R/W	U	R/W	R/W
Сброс	0	00000000	000	0	0000
	-	DTG[7:0]	-	EDTS	DTGx[3:0]

Таблица 28-22– Описание бит регистра CHy_DTG

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...16	-	Зарезервировано
15...8	DTG[7:0]	Основной делитель частоты DTG. Задержка DTGdel = DTG • (DTGx + 1)
7...5	-	Зарезервировано
4	EDTS	Частота работы DTG: 0 – TIM_CLK; 1 – F _{DTS}
3...0	DTGx[3:0]	Предварительный делитель частоты DTG

28.10.11 MDR_TIMERx->BRKETR_CNTRL

Таблица 28-23– Регистр BRKETR_CNTRL управления входом BRK и ETR

Номер	31...8	7...4	3...2	1	0
Доступ	U	R/W	R/W	R/W	R/W
Сброс	0	0000	00	0	0
	-	ETR_FILTER[3:0]	ETR_PSC[1:0]	ETR_INV	BRK_INV

Таблица 28-24– Описание бит регистра BRKETR_CNTRL

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...8	-	Зарезервировано
7...4	ETR_FILTER[3:0]	Конфигурация фильтра на входе ETR. Выбор частоты выборки F_s и количества выборок N : 0000 – нет фильтрации, $F_s = F_{DTS}$; 0001 – $F_s = TIM\_CLK$, $N = 2$; 0010 – $F_s = TIM\_CLK$, $N = 4$; 0011 – $F_s = TIM\_CLK$, $N = 8$; 0100 – $F_s = F_{DTS}/2$, $N = 6$; 0101 – $F_s = F_{DTS}/2$, $N = 8$; 0110 – $F_s = F_{DTS}/4$, $N = 6$; 0111 – $F_s = F_{DTS}/4$, $N = 8$; 1000 – $F_s = F_{DTS}/8$, $N = 6$; 1001 – $F_s = F_{DTS}/8$, $N = 8$; 1010 – $F_s = F_{DTS}/16$, $N = 5$; 1011 – $F_s = F_{DTS}/16$, $N = 6$; 1100 – $F_s = F_{DTS}/16$, $N = 8$; 1101 – $F_s = F_{DTS}/32$, $N = 5$; 1110 – $F_s = F_{DTS}/32$, $N = 6$; 1111 – $F_s = F_{DTS}/32$, $N = 8$
3...2	ETR_PSC[1:0]	Асинхронный предделитель частоты со входа ETR: 00 – без деления; 01 – /2; 10 – /4; 11 – /8
1	ETR_INV	Инверсия входа ETR: 0 – без инверсии; 1 – инверсия
0	BRK_INV	Инверсия входа BRK: 0 – без инверсии; 1 – инверсия

28.10.12 MDR_TIMERx->STATUS

Таблица 28-25– Регистр статуса таймера STATUS

Номер	31...17	16...13	12...9	8...5
Доступ	U	R/W	R/W	R/W
Сброс	0	0	0	0
	-	CCR CAP1 EVENT[3:0]	CCR REF EVENT[3:0]	CCR CAP EVENT[3:0]

Номер	4	3	2	1	0
Доступ	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0
	BRK EVENT	ETR FE EVENT	ETR RE EVENT	CNT ARR EVENT	CNT ZERO EVENT

Таблица 28-26– Описание бит регистра STATUS

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...17	-	Зарезервировано
16...13	CCR CAP1 EVENT[3:0]	Событие записи значения счетчика CNT в регистр CCRy1 по захвату настроенного фронта на входе канала CHy1: 0 – нет события; 1 – есть событие. Сбрасывается записью «0», если запись происходит одновременно с новым событием, приоритет у нового события. Бит 0 – первый канал; Бит 3 – четвертый канал
12...9	CCR REF EVENT[3:0]	Событие переднего фронта на выходе генератора опорного сигнала REF: 0 – нет события; 1 – есть событие. Сбрасывается записью «0», если запись происходит одновременно с новым событием, приоритет у нового события. Бит 0 – первый канал; Бит 3 – четвертый канал
8...5	CCR CAP EVENT[3:0]	Событие записи значения счетчика CNT в регистр CCRy по захвату настроенного фронта на входе канала CHy1: 0 – нет события; 1 – есть событие. Сбрасывается записью «0», если запись происходит одновременно с новым событием, приоритет у нового события. Бит 0 – первый канал; Бит 3 – четвертый канал
4	BRK EVENT	Событие высокого уровня на входе BRK: 0 – нет события; 1 – есть событие. Сбрасывается записью «0», при условии наличия низкого уровня на входе BRK
3	ETR FE EVENT	Событие заднего фронта на входе ETR: 0 – нет события; 1 – есть событие. Сбрасывается записью «0», если запись происходит одновременно с новым событием, приоритет у нового события
2	ETR RE EVENT	Событие переднего фронта на входе ETR: 0 – нет события; 1 – есть событие. Сбрасывается записью «0», если запись происходит одновременно с новым событием, приоритет у нового события

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
1	CNT ARR EVENT	Событие совпадения CNT с ARR: 0 – нет события; 1 – есть событие. Сбрасывается записью «0», если запись происходит одновременно с новым событием совпадения, приоритет у нового события. Если с момента совпадения до момента программного сброса флага регистра CNT и ARR не изменили состояния, то флаг повторно не взводится
0	CNT ZERO EVENT	Событие совпадения CNT с нулем: 0 – нет события; 1 – есть событие. Сбрасывается записью «0», если запись происходит одновременно с новым событием совпадения, приоритет у нового события. Если с момента совпадения до момента программного сброса флага регистра CNT не изменил состояния, то флаг повторно не взводится
Примечание – Для организации надежного сброса статусных флагов прерываний необходимо учитывать требования, изложенные в подразделе 36.3		

28.10.13 MDR_TIMERx->IE

Таблица 28-27– Регистр разрешения прерываний таймера IE

Номер	31...17	16...13	12...9	8...5
Доступ	U	R/W	R/W	R/W
Сброс	0	0	0	0
	-	CCR CAP1 EVENT IE[3:0]	CCR REF EVENT IE[3:0]	CCR CAP EVENT IE[3:0]

Номер	4	3	2	1	0
Доступ	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0
	BRK EVENT IE	ETR FE EVENT IE	ETR RE EVENT IE	CNT ARR EVENT IE	CNT ZERO EVENT IE

Таблица 28-28– Описание бит регистра IE

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...17	-	Зарезервировано
16...13	CCR CAP1 EVENT IE[3:0]	Флаг разрешения прерывания по событию CCR CAP1 EVENT[3:0] в регистре STATUS: 0 – прерывание запрещено; 1 – прерывание разрешено. Бит 0 – первый канал; Бит 3 – четвертый канал

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
12...9	CCR REF EVENT IE[3:0]	Флаг разрешения прерывания по событию CCR REF EVENT[3:0] в регистре STATUS: 0 – прерывание запрещено; 1 – прерывание разрешено. Бит 0 – первый канал; Бит 3 – четвертый канал
8...5	CCR CAP EVENT IE[3:0]	Флаг разрешения прерывания по событию CCR CAP EVENT[3:0] в регистре STATUS: 0 – прерывание запрещено; 1 – прерывание разрешено. Бит 0 – первый канал; Бит 3 – четвертый канал
4	BRK EVENT IE	Флаг разрешения прерывания по событию BRK EVENT в регистре STATUS: 0 – прерывание запрещено; 1 – прерывание разрешено
3	ETR FE EVENT IE	Флаг разрешения прерывания по событию ETR FE EVENT в регистре STATUS: 0 – прерывание запрещено; 1 – прерывание разрешено
2	ETR RE EVENT IE	Флаг разрешения прерывания по событию ETR RE EVENT в регистре STATUS: 0 – прерывание запрещено; 1 – прерывание разрешено
1	CNT ARR EVENT IE	Флаг разрешения прерывания по событию CNT ARR EVENT в регистре STATUS: 0 – прерывание запрещено; 1 – прерывание разрешено
0	CNT ZERO EVENT IE	Флаг разрешения прерывания по событию CNT ZERO EVENT в регистре STATUS: 0 – прерывание запрещено; 1 – прерывание разрешено

28.10.14 MDR_TIMERx->DMA_RE

Таблица 28-29– Регистр DMA_RE разрешения запросов DMA

Номер	31...17	16...13	12...9	8...5
Доступ	U	R/W	R/W	R/W
Сброс	0	0	0	0
	-	CCR CAP1 EVENT RE[3:0]	CCR REF EVENT RE[3:0]	CCR CAP EVENT RE[3:0]

Номер	4	3	2	1	0
Доступ	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0
	BRK EVENT RE	ETR FE EVENT RE	ETR RE EVENT RE	CNT ARR EVENT RE	CNT ZERO EVENT RE

Таблица 28-30 – Описание бит регистра DMA_RE

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...17	-	Зарезервировано
16...13	CCR CAP1 EVENT RE [3:0]	Флаг разрешения запроса DMA по событию CCR CAP1 EVENT[3:0]: 0 – запрос DMA запрещен; 1 – запрос DMA разрешен. Бит 0 – первый канал; Бит 3 – четвертый канал
12...9	CCR REF EVENT RE[3:0]	Флаг разрешения запроса DMA по событию CCR REF EVENT[3:0]: 0 – запрос DMA запрещен; 1 – запрос DMA разрешен. Бит 0 – первый канал; Бит 3 – четвертый канал
8...5	CCR CAP EVENT RE [3:0]	Флаг разрешения запроса DMA по событию CCR CAP EVENT[3:0]: 0 – запрос DMA запрещен; 1 – запрос DMA разрешен. Бит 0 – первый канал; Бит 3 – четвертый канал
4	BRK EVENT RE	Флаг разрешения запроса DMA по событию BRK EVENT: 0 – запрос DMA запрещен; 1 – запрос DMA разрешен
3	ETR FE EVENT RE	Флаг разрешения запроса DMA по событию ETR FE EVENT: 0 – запрос DMA запрещен; 1 – запрос DMA разрешен
2	ETR RE EVENT RE	Флаг разрешения запроса DMA по событию ETR RE EVENT: 0 – запрос DMA запрещен; 1 – запрос DMA разрешен
1	CNT ARR EVENT RE	Флаг разрешения запроса DMA по событию CNT ARR EVENT: 0 – запрос DMA запрещен; 1 – запрос DMA разрешен
0	CNT ZERO EVENT RE	Флаг разрешения запроса DMA по событию CNT ZERO EVENT: 0 – запрос DMA запрещен; 1 – запрос DMA разрешен

29 Контроллер MDR_ADC

В микросхеме реализовано два 12-разрядных АЦП. С помощью АЦП можно оцифровать сигнал от 16-внешних аналоговых выводов порта D и от двух внутренних каналов, на которые выводятся датчик температуры и источник опорного напряжения. Скорость выборки составляет до 512 тысяч преобразований в секунду для каждого АЦП.

В качестве опорного напряжения преобразования могут выступать:

- питание АЦП с выводов U_{CCA} и GND;
- внешние сигналы с выводов ADC0_REF+ и ADC1_REF-.

Контроллер АЦП позволяет:

- оцифровать один из 16-ти внешних каналов;
- оцифровать значение встроенного датчика температуры;
- оцифровать значение встроенного источника опорного напряжения;
- осуществить автоматический опрос заданных каналов;
- выработать прерывание при выходе оцифрованного значения за заданные пределы;
- запускать два АЦП синхронно для увеличения скорости выборки.

Для осуществления преобразования требуется не менее 28-ми тактов синхронизации C_ADC. В качестве синхросигнала может выступать частота процессора PCLKd, либо частота ADC_CLK. Выбор частоты осуществляется с помощью бита Cfg_REG_CLKS. Частота PCLKd формируется из частоты процессорного ядра делением на коэффициент Cfg_REG_DIVCLK[3:0]. Максимальная частота C_ADC не может превышать 14 МГц.

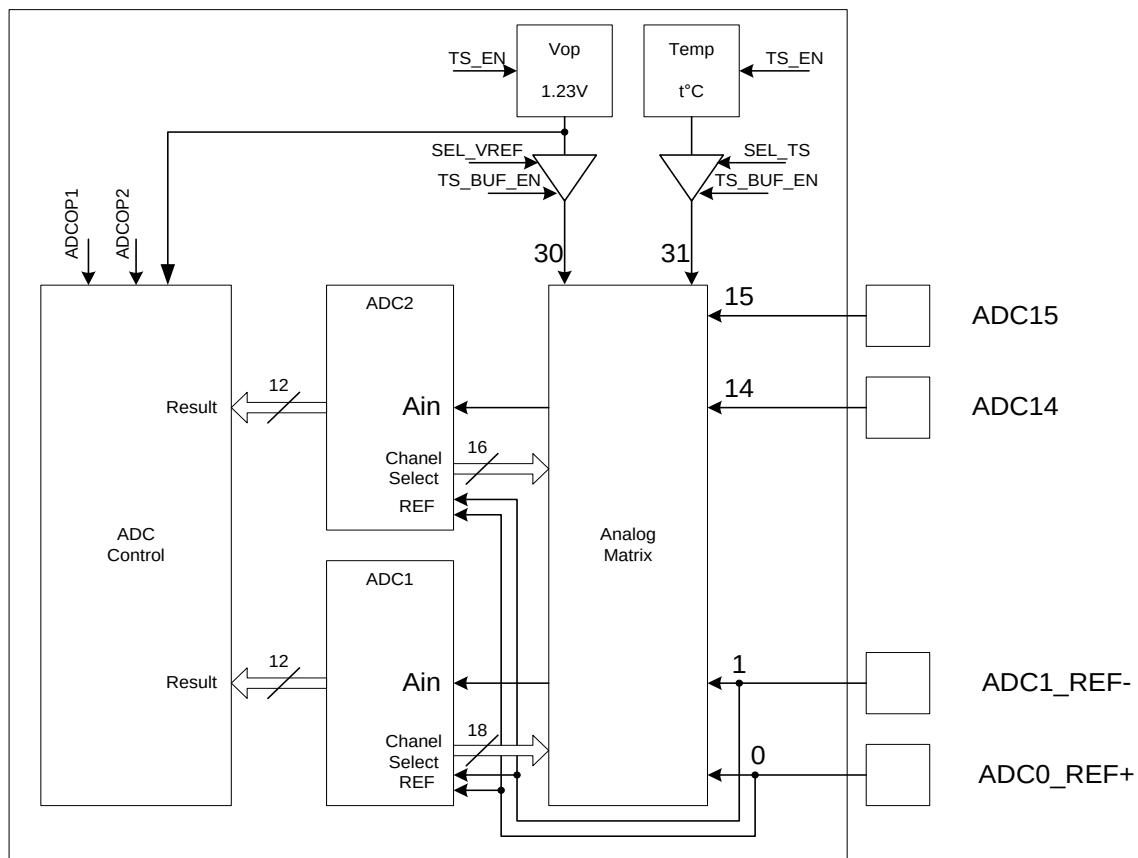


Рисунок 29-1 – Структурная схема контроллера АЦП

Для включения АЦП необходимо установить бит Cfg_REG_ADON. Для снижения тока потребления вместо собственного источника опорного напряжения в АЦП может использоваться источник датчика температуры. Для этого необходимо включить блок датчика температуры и источник опорного напряжения, установив бит TS_EN в «1». После включения можно использовать источник опорного напряжения для первого и второго АЦП вместо их собственных. Для этого необходимо установить биты ADCx_OP в «1». Для преобразования необходимо, чтобы выводы, используемые АЦП у порта D, были сконфигурированы как аналоговые и были отключены какие-либо внутренние подтяжки.

29.1 Преобразование внешнего канала

В регистре ADCx_CFG в битах Cfg_REG_CHS[4:0] необходимо задать соответствующий выводу номер канала. Преобразование может осуществляться при внутренней опоре бит Cfg_M_REF = 0 и внешней Cfg_M_REF = 1, в этом случае опора берется с выводов ADC0_REF+ и ADC1_REF-. Биты Cfg_REG_CHCH, Cfg_REG_RNGC, Cfg_REG_SAMPLE, TS_BUF_EN, SEL_VREF, SEL_TS и Cfg_Sync_Conver должны быть сброшены.

Для начала преобразования необходимо записать 1 в бит Cfg_REG_GO.

После завершения преобразования будет взведен бит Flg_REG_EOCIF в регистре ADCx_STATUS, а в регистре ADCx_RESULT будет результат преобразования.

После считывания результата бит Flg_REG_EOCIF сбросится.

Если после первого преобразования результат не был считан, и было выполнено второе преобразование, то в регистре результата ADCx_RESULT будет значение от последнего преобразования, и помимо бита Flg_REG_EOCIF будет взведен бит Flg_REG_OVERWRITE. Флаг Flg_REG_OVERWRITE может быть сброшен только записью в регистр ADCx_STATUS.

29.2 Последовательное преобразование нескольких каналов

Для автоматического последовательного преобразования нескольких каналов или одного канала в регистре ADCx_CHSEL необходимо установить единицы в битах, соответствующих выбранным для преобразования каналам. Выставление данных бит необходимо обеспечить до установки конфигурации АЦП, то есть до записи в регистр ADCx_CFG. Преобразование может осуществляться при внутренней опоре бит Cfg_M_REF = 0 и внешней Cfg_M_REF = 1. В этом случае опора берется с выводов ADC0_REF+ и ADC1_REF-. Биты Cfg_REG_RNGC, TS_BUF_EN, SEL_VREF, SEL_TS и Cfg_Sync_Conver должны быть сброшены, а биты Cfg_REG_SAMPLE и Cfg_REG_CHCH должны быть установлены. С помощью бит Delay_GO можно задать паузу между преобразованиями при переборе каналов. Эта определяется в тактах CPU_CLK, независимо от того на какой частоте ADC_CLK или CPU_CLK идет само преобразование. Для начала преобразования необходимо записать «1» в бит Cfg_REG_GO.

После завершения преобразования будет взведен бит Flg_REG_EOCIF в регистре ADCx_STATUS, а в регистре ADCx_RESULT будет результат преобразования.

После считывания результата бит Flg_REG_EOCIF сбросится.

Если после первого преобразования результат не был считан, и было выполнено второе преобразование, то в регистре результата ADCx_RESULT будет значение от последнего преобразования, и помимо бита Flg_REG_EOCIF будет взведен бит Flg_REG_OVERWRITE. Флаг Flg_REG_OVERWRITE может быть сброшен только записью в регистр ADCx_STATUS.

Для последовательного преобразования одного и того же канала можно в регистре ADCx_CHSEL выбрать только один канал и установить бит Cfg_REG_CHCH в «1», либо установить номер канала в битах Cfg_REG_CHS[4:0] и сбросить бит Cfg_REG_CHCH в 0. В этом случае процесс последовательного преобразования будет выполняться только для данного канала. Последовательное преобразование значения датчика температуры и источника опорного напряжения могут выполняться только в режиме последовательного преобразования одного канала.

29.3 Преобразование с контролем границ

При необходимости отслеживать нахождение оцифрованных значений в допустимых пределах можно задать нижнюю и верхнюю допустимые границы в регистрах ADCx_L_LEVEL и ADCx_H_LEVEL. При этом, если установлен бит Cfg_REG_RNGC, то в случае, если результат преобразования выходит за границы, выставляется флаг Flg_REG_AWOIFEN, а в регистре результата будет полученное значение.

29.4 Внутренний источник опорного напряжения

С помощью первого АЦП можно осуществить преобразования внутреннего источника опорного напряжения (блок V_{OP}). Для этого необходимо включить блок датчика температуры и источник опорного напряжения, установив бит TS_EN в «1». После включения можно использовать источник опорного напряжения для первого и второго АЦП вместо их собственных, что позволяет снизить ток потребления. Для этого необходимо установить биты ADCx_OP в единицу. Для выбора источника опорного напряжения в качестве источника для преобразования необходимо в битах Cfg_REG_CHS установить значение 30 канала, установить биты TS_BUF_EN и SEL_VREF, после чего можно запустить процесс преобразования. Для запуска преобразования необходимо записать «1» в бит Cfg_REG_GO.

После завершения преобразования будет взведен бит Flg_REG_EOCIF в регистре ADC1_STATUS, а в регистре ADC1_RESULT будет результат преобразования.

После считывания результата бит Flg_REG_EOCIF сбросится.

Если после первого преобразования результат не был считан, и было выполнено второе преобразование, то в регистре результата ADC1_RESULT будет значение от последнего преобразования, а помимо бита Flg_REG_EOCIF будет взведен бит Flg_REG_OVERWRITE. Флаг Flg_REG_OVERWRITE может быть сброшен только записью в регистр ADC1_STATUS.

Для последовательного преобразования только источника опорного напряжения можно в регистре ADC1_CHSEL выбрать только 30 канал и установить бит Cfg_REG_CHCH в «1», либо установить номер 30-го канала в битах Cfg_REG_CHS[4:0] и сбросить бит Cfg_REG_CHCH в 0. В этом случае процесс

последовательного преобразования будет выполняться только для данного канала. При этом должны быть также установлены биты TS_BUF_EN и SEL_VREF.

Источник опорного напряжения может быть выбран для более точного результата преобразования АЦП и не может быть использован для задания опорного напряжения преобразования.

29.5 Датчик температуры

С помощью первого АЦП можно осуществить преобразования датчика опорного напряжения. Для этого необходимо включить блок датчика температуры и источник опорного напряжения, установив бит TS_EN в «1». После включения можно использовать источник опорного напряжения для первого и второго АЦП вместо их собственных, что позволяет снизить ток потребления. Для этого необходимо установить биты ADCx_OR в единицу. Для выбора датчика температуры в качестве источника для преобразования необходимо в битах Cfg_REG_CHS установить значение 31 канала, установить биты TS_BUF_EN и SEL_TS, после чего можно запустить процесс преобразования. Для начала преобразования необходимо записать «1» в бит Cfg_REG_GO.

После завершения преобразования будет взведен бит Flg_REG_EOCIF в регистре ADC1_STATUS, а в регистре ADC1_RESULT будет результат преобразования.

После считывания результата бит Flg_REG_EOCIF сбросится.

Если после первого преобразования результат не был считан, и было выполнено второе преобразование, то в регистре результата ADC1_RESULT будет значение от последнего преобразования, и помимо бита Flg_REG_EOCIF будет взведен бит Flg_REG_OVERWRITE. Флаг Flg_REG_OVERWRITE может быть сброшен только записью в регистр ADC1_STATUS.

Для последовательного преобразования только датчика температуры можно в регистре ADC1_CHSEL выбрать только 31 канал и установить бит Cfg_REG_CHCH в «1», либо установить номер 31-го канала в битах Cfg_REG_CHS[4:0] и сбросить бит Cfg_REG_CHCH в 0. В этом случае процесс последовательного преобразования будет выполняться только для данного канала. При этом должны быть также установлены биты TS_BUF_EN и SEL_TS.

Параметры температурного датчика не регламентируются. В зависимости от необходимой точности может быть достаточно провести градуировку в двух – трех точках. При необходимости более точных измерений необходимо построить градуировочную таблицу.

Градуировка производится индивидуально для каждой микросхемы.

29.6 Синхронный запуск двух АЦП

Для ускорения оцифровки одного канала можно использовать оба АЦП, запускаемые с задержкой одного относительно другого по времени. Время задержки запуска второго АЦП относительно первого задается битами Delay_ADC. При этом задержка Delay_ADC определяется в тактах PCLKd, независимо от того на какой частоте ADC_CLK или PCLKd идет само преобразование. Для одновременного запуска процесса преобразования необходимо установить бит Cfg_Sync_Conver и запустить процесс преобразования установкой бита Cfg_REG_GO. Синхронный запуск двух АЦП

может работать также и в режиме последовательного преобразования нескольких каналов.

29.7 Время заряда внутренней емкости

Процесс преобразования состоит из двух этапов: сначала происходит заряд внутренней емкости до уровня внешнего сигнала, и затем происходит преобразование уровня заряда внутренней емкости в цифровой вид. Таким образом, для точного преобразования внешнего сигнала в цифровой вид, за время первого этапа внутренняя емкость должна зарядиться до уровня внешнего сигнала. Это время определяется соотношением номинальной внутренней емкости, входным сопротивлением тракта АЦП и выходным сопротивлением источника сигнала. Максимальное выходное сопротивление источника R_{AIN} для обеспечения качественного преобразования определяется по формулам

$$R_{AIN} < \frac{T_{track}}{C_{ADC} \cdot \ln(2^N)} - R_{ADC}, \quad (3)$$

$$T_{track} = 4 \cdot T_{C\_ADC} + N_{PCLKd} \cdot T_{PCLKd} = \frac{4}{f_{C\_ADC}} + \frac{(DelayGo + 1)}{f_{PCLKd}}, \quad (4)$$

где C_{ADC} – внутренняя емкость АЦП (~15 – 20 пФ);

N – требуемая точность, в разрядах;

R_{ADC} – входное сопротивление тракта АЦП (~500 Ом);

T_{track} – время заряда внутренней емкости (определяется формулой (4)), с;

f_{C_ADC} – рабочая частота АЦП (определяется Cfg REG CLKS в регистре ADC1_CFG), с⁻¹;

f_{PCLKd} – определяется по формуле

$$f_{PCLKd} = \frac{f_{PCLK}}{2^{Cfg\_REG\_DIVCLK}} \quad (5)$$

Если необходимо обеспечить преобразование с точностью 12 разрядов $\pm 1/4$ LSB, то $N = 14$. Если необходимо обеспечить преобразование с точностью 10 разрядов ± 1 LSB, то $N = 10$. Время заряда T_{track} определяется битами DelayGo[2:0] и схемой самого АЦП и представлено в таблице ниже (Таблица 29-1).

Таблица 29-1 – Время заряда внутренней емкости АЦП и время преобразования

DelayGo[2:0]	Дополнительная задержка перед началом преобразования	Общее время Ttrack заряда емкости АЦП перед началом преобразования	Общее время преобразования АЦП
000	$1 \times PCLKd$	$4 \times C\_ADC + 1 \times PCLKd$	$28 \times C\_ADC + 1 \times PCLKd$
001	$2 \times PCLKd$	$4 \times C\_ADC + 2 \times PCLKd$	$28 \times C\_ADC + 2 \times PCLKd$
010	$3 \times PCLKd$	$4 \times C\_ADC + 3 \times PCLKd$	$28 \times C\_ADC + 3 \times PCLKd$
011	$4 \times PCLKd$	$4 \times C\_ADC + 4 \times PCLKd$	$28 \times C\_ADC + 4 \times PCLKd$
100	$5 \times PCLKd$	$4 \times C\_ADC + 5 \times PCLKd$	$28 \times C\_ADC + 5 \times PCLKd$
101	$6 \times PCLKd$	$4 \times C\_ADC + 6 \times PCLKd$	$28 \times C\_ADC + 6 \times PCLKd$

DelayGo[2:0]	Дополнительная задержка перед началом преобразования	Общее время Ttrack заряда емкости АЦП перед началом преобразования	Общее время преобразования АЦП
110	$7 \times \text{PCLKd}$	$4 \times \text{C\_ADC} + 7 \times \text{PCLKd}$	$28 \times \text{C\_ADC} + 7 \times \text{PCLKd}$
111	$8 \times \text{PCLKd}$	$4 \times \text{C\_ADC} + 8 \times \text{PCLKd}$	$28 \times \text{C\_ADC} + 8 \times \text{PCLKd}$

Помимо точности определяемой временем зарядки внутренней емкости АЦП, точность преобразования имеет ошибки, связанные с технологическими разбросами схемы и шумами, и определяемые параметрами E_{DLADC} , E_{ILADC} и E_{OFFADC} .

Задание режимов работы АЦП в регистре ADCx_CFG необходимо производить до задания бита Cfg_REG_GO, иначе новая конфигурация будет действовать со следующего преобразования.

29.8 Описание регистров блока контроллера АЦП

Таблица 29-2 – Описание регистров блока контроллера АЦП

Базовый Адрес	Название	Описание
0x4008_8000	MDR_ADC	Контроллер ADC
Смещение		
0x00	MDR_ADC->ADC1_CFG	Регистр управления ADC1
0x04	MDR_ADC->ADC2_CFG	Регистр управления ADC2
0x08	ADC1_H_LEVEL	Регистр MDR_ADC->ADCx_H_LEVEL верхней границы ADC1
0x0C	ADC2_H_LEVEL	Регистр MDR_ADC->ADCx_H_LEVEL верхней границы ADC2
0x10	ADC1_L_LEVEL	Регистр MDR_ADC->ADCx_L_LEVEL нижней границы ADC1
0x14	ADC2_L_LEVEL	Регистр MDR_ADC->ADCx_L_LEVEL нижней границы ADC2
0x18	ADC1_RESULT	Регистр MDR_ADC->ADCx_RESULT результата ADC1
0x1C	ADC2_RESULT	Регистр MDR_ADC->ADCx_RESULT результата ADC2
0x20	ADC1_STATUS	Регистр MDR_ADC->ADCx_STATUS статуса ADC1
0x24	ADC2_STATUS	Регистр MDR_ADC->ADCx_STATUS статуса ADC2
0x28	ADC1_CHSEL	Регистр MDR_ADC->ADCx_CHSEL выбора каналов перебора ADC1
0x2C	ADC2_CHSEL	Регистр MDR_ADC->ADCx_CHSEL выбора каналов перебора ADC2

29.8.1 MDR_ADC->ADC1_CFG

Таблица 29-3 – Регистр ADC1_CFG

Номер	11	10	9	8...4	3	2	1	0
Доступ	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0	0	0
	Cfg M_REF	Cfg REG RNGC	Cfg REG CHCH	Cfg REG CHS[4:0]	Cfg REG SAMPLE	Cfg REG CLKS	Cfg REG GO	Cfg REG ADON

Номер	31...28	27...25	24...21	20	19	18	17	16	15...12
Доступ	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0	0	0	0
	Delay ADC [3:0]	Delay Go[2:0]	TR[3:0]	SEL VREF	SEL TS	TS_BUF EN	TS_EN	Cfg Sync Conver	Cfg REG DIVCLK[3:0]

Таблица 29-4 – Описание бит регистра ADC1_CFG

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...28	Delay ADC[3:0]	Задержка между началом преобразования ADC1 и ADC2 при последовательном переборе, либо работе на один канал: 0000 – 1 такт PCLKd; 0001 – 2 такта PCLKd; ... 1111 – 16 тактов PCLKd
27...25	Delay Go [2:0]	Дополнительная задержка перед началом преобразования после выбора канала: 000 – 1 такт PCLKd; 001 – 2 такта PCLKd; ... 111 – 8 тактов PCLKd
24...21	TR[3:0]	Подстройка опорного напряжения. Смотрите диаграмму зависимости источника опорного напряжения от подстройки (Рисунок 29-2)
20	SEL VREF	Выбор для оцифровки источника опорного напряжения на 1,23 В: 0 – не выбран; 1 – выбран. Должен использоваться совместно с выбором канала Cfg_REG_CHS = 30
19	SEL TS	Выбор для оцифровки датчика температуры: 0 – не выбран; 1 – выбран. Должен использоваться совместно с выбором канала Cfg_REG_CHS = 31
18	TS BUF EN	Включения выходного усилителя для датчика температуры и источника опорного напряжения: 0 – выключен;

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
		1 – включен. Используется при TS_EN = 1
17	TS EN	Включения датчика температуры и источника опорного напряжения: 0 – выключен; 1 – включен. При включении датчика температуры и источника опорного напряжения выходной сигнал стабилизируется в течении времени 1 мс
16	Cfg Sync Conver	Запускает работу двух АЦП одновременно, при этом биты конфигурации второго АЦП, такие как Cfg_REG_DIVCLK, Cfg_REG_ADON, Cfg_M_REF и Cfg_REG_CHS берутся из регистра конфигурации первого: 0 – независимые АЦП; 1 – синхронные АЦП
15...12	Cfg REG DIVCLK[3:0]	Выбор коэффициента деления частоты процессора: 0000 – PCLKd = PCLK; 0001 – PCLKd = PCLK /2; 0010 – PCLKd = PCLK /4; 0011 – PCLKd = PCLK/8; ... 1011 – PCLKd = PCLK/2048 Остальные комбинации – PCLKd = PCLK
11	Cfg M_REF	Выбор источника опорных напряжений: 0 – внутренне опорное напряжение (от U _{ССА} и GND (вывод 60 корпуса 4229.132-3, выводы 65, 66 корпуса LQFP144)); 1 – внешнее опорное напряжение (от ADC0_REF+ и ADC1_REF-)
10	Cfg REG RNGC	Разрешение автоматического контроля уровней: 0 – не разрешена; 1 – разрешена выработка флага при выходе за диапазон в регистрах границы
9	Cfg REG CHCH	Выбор переключения каналов: 0 – используется только выбранный канал; 1 – переключение включено (перебираются каналы, выбранные в регистре выбора канала)
8...4	Cfg REG CHS[4:0]	Выбор аналогового канала, по которому поступает сигнал для преобразования: 00000 – 0 канал; 00001 – 1 канал; ... 11111 – 31 канал
3	Cfg REG SAMPLE	Выбор способа запуска АЦП: 0 – одиночное; 1 – последовательное. Автоматический запуск после завершения предыдущего преобразования
2	Cfg	Выбор источника синхросигнала C_ADC работы ADC:

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
	REG CLKS	0 – PCLKd (определяется по формуле (5)); 1 – ADC_CLK
1	Cfg REG GO	Начало преобразования. Запись «1» начинает процесс преобразования, сбрасывается автоматически
0	Cfg REG ADON	Включение АЦП: 0 – выключено; 1 – включено

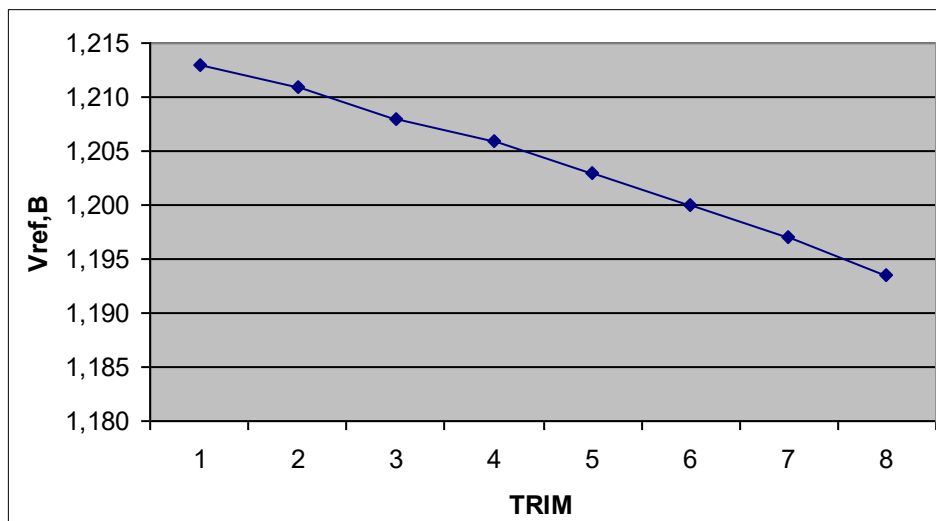


Рисунок 29-2 – Зависимость источника опорного напряжения от подстройки

29.8.2 MDR_ADC->ADC2_CFG

Таблица 29-5 – Регистр ADC2_CFG

Номер	11	10	9	8...4	3	2	1	0
Доступ	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0	0	0
	Cfg M_REF	Cfg REG RNGC	Cfg REG CHCH	Cfg REG CHS[4:0]	Cfg REG SAMPLE	Cfg REG CLKS	Cfg REG GO	Cfg REG ADON

Номер	31...28	27...25	24...19	18	17	16	15...12
Доступ	U	R/W	U	R/W	R/W	U	R/W
Сброс	0	0	0	0	0	0	0
	-	Delay Go[2:0]	-	ADC2 OP	ADC1 OP	-	Cfg REG DIVCLK[3:0]

Таблица 29-6 – Описание бит регистра ADC2_CFG

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...28	-	Зарезервировано

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
27...25	Delay Go[2:0]	Задержка перед началом следующего преобразования после завершения предыдущего при последовательном переборе каналов: 000 – 1 такт PCLKd; 001 – 2 такта PCLKd; ... 111 – 8 тактов PCLKd
24...19	-	Зарезервировано
18	ADC2 OP	Выбор источника опорного напряжения 1.23 В: 0 – внутренний (неточный); 1 – от датчика температуры (точный)
17	ADC1 OP	Выбор источника опорного напряжения 1.23 В: 0 – внутренний (неточный); 1 – от датчика температуры (точный)
16	-	Зарезервировано
15...12	Cfg REG DIVCLK[3:0]	Выбор коэффициента деления частоты процессора: 0000 – PCLKd = PCLK; 0001 – PCLKd = PCLK /2; 0010 – PCLKd = PCLK /4; 0011 – PCLKd = PCLK/8; ... 1011 – PCLKd = PCLK/2048; Остальные комбинации – PCLKd = PCLK
11	Cfg M_REF	Выбор источника опорных напряжений: 0 – внутренне опорное напряжение (от U _{ССА} и GND (вывод 60 корпуса 4229.132-3, выводы 65, 66 корпуса LQFP144)); 1 – внешнее опорное напряжение (от ADC0_REF+ и ADC1_REF-)
10	Cfg REG RNGC	Разрешение автоматического контролирования уровней: 1 – разрешено, выработка прерывания при выходе за диапазон в регистрах границы обработки; 0 – не разрешено
9	Cfg REG CHCH	Выбор переключения каналов: 0 – используется только выбранный канал; 1 – переключение включено (перебираются каналы, выбранные в регистре выбора канала)
8...4	Cfg REG CHS [4:0]	Выбор аналогового канала, по которому поступает сигнал для преобразования: 00000 – 0 канал; 00001 – 1 канал; ... 11111 – 31 канал
3	Cfg REG SAMPLE	Выбор способа запуска АЦП: 0 – одиночное; 1 – последовательное (автоматический запуск после завершения предыдущего преобразования).

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
2	Cfg REG CLKS	Выбор источника синхросигнала C_ADC работы ADC: 0 – PCLKd (определяется по формуле (5)); 1 – ADC_CLK
1	Cfg REG GO	Начало преобразования. Запись «1» начинает процесс преобразования. Сбрасывается автоматически
0	Cfg REG ADON	Включение АЦП: 0 – выключено; 1 – включено

29.8.3 MDR_ADC->ADCx_H_LEVEL

Таблица 29-7 – Регистр ADCx_H_LEVEL

Номер	31...12	11...0
Доступ	U	R/W
Сброс	0	0
	-	REG H LEVEL [11:0]

Таблица 29-8 – Описание бит регистра ADCx_H_LEVEL

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...12	-	Зарезервировано
11...0	REG H LEVEL [11:0]	Верхняя граница зоны допуска.

29.8.4 MDR_ADC->ADCx_L_LEVEL

Таблица 29-9 – Регистр ADCx_L_LEVEL

Номер	31...12	11...0
Доступ	U	R/W
Сброс	0	0
	-	REG L LEVEL [11:0]

Таблица 29-10 – Описание бит регистра ADCx_L_LEVEL

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...12	-	Зарезервировано
11...0	REG L LEVEL [11:0]	Нижняя граница зоны допуска

29.8.5 MDR_ADC->ADCx_RESULT

Таблица 29-11 – Регистр ADCx_RESULT

Номер	31...21	20...16	15...12	11...0
Доступ	U	RO	U	RO

Сброс	0	0	0	0
	-	CHANNEL[11:0]	-	RESULT[11:0]

Таблица 29-12 – Описание бит регистра ADCx_RESULT

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...21	-	Зарезервировано
20...16	CHANNEL [11:0]	Канал результата преобразования
15...12	-	Зарезервировано
11...0	RESULT [11:0]	Значение результата преобразования

29.8.6 MDR_ADC->ADCx_STATUS

Таблица 29-13 – Регистр ADCx_STATUS

Номер	31...5	4	3	2	1	0
Доступ	U	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0
	-	ECOIF IE	AWOIFIE	Flg REG EOCIF	Flg REG AWOIFEN	Flg REG OVERWRITE

Таблица 29-14 – Описание бит регистра ADCx_STATUS

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...5	-	Зарезервировано
4	ECOIF_IE	Флаг разрешения генерирования прерывания по событию Flg_REG_ECOIF: 0 – прерывания не генерируется; 1 – прерывание генерируется
3	AWOIF_IE	Флаг разрешения генерирования прерывания по событию Flg_REG_AWOIFEN: 0 – прерывания не генерируется; 1 – прерывание генерируется
2	Flg REG EOCIF	Флаг выставляется, когда закончено преобразования и данные еще не считаны. Очищается считыванием результата из регистра ADCx_RESULT: 1 – есть готовый результат преобразования; 0 – нет результата
1	Flg REG AWOIFEN	Флаг выставляется, когда результат преобразования выше верхней или ниже нижней границы автоматического контролирования уровней. Сбрасывается только при записи нуля в данный бит регистр флагов: 0 – результат в допустимой зоне; 1 – вне допустимой зоны

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
0	Flg REG OVERWRITE	Данные в регистре результата были перезаписаны, данный флаг сбрасывается только при записи нуля в данный бит регистр флагов: 0 – не было события перезаписи несчитанного результата; 1 – был результат преобразования, который не был считан
Примечание – Для организации надежного сброса статусных флагов прерываний необходимо учитывать требования, изложенные в подразделе 36.3		

29.8.7 MDR_ADC->ADCx_CHSEL

Таблица 29-15 – Регистр ADCx_CHSEL

Номер	31... 0
Доступ	R/W
Сброс	0
	Sl_Ch_Ch_REF[31:0]

Таблица 29-16 – Описание бит регистра ADCx_CHSEL

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...0	Sl_Ch_Ch_REF[31:0]	Выбор каналов автоматического перебора: 0 – в соответствующем бите канал не участвует в переборе; 1 – канал участвует в переборе

30 Контроллер MDR_DAC

В микросхеме реализовано два ЦАП. Для включения ЦАП необходимо установить бит Cfg_ON_DACx в 1, используемые выводы ЦАП порта E были сконфигурированы, как аналоговые и были отключены какие-либо внутренние подтяжки. Оба ЦАП могут работать независимо или совместно. При независимой работе ЦАП (бит Cfg_SYNC_A=0) после записи данных в регистр данных DACx_DATA на выходе DACx_OUT формируется уровень напряжения, соответствующий записанному значению. При синхронной работе (бит Cfg_SYNC_A=1) данные обоих ЦАП могут быть обновлены одной записью в один из регистров DACx_DATA. ЦАП может работать от внутренней опоры Cfg_M_REFx=0, тогда ЦАП формирует выходной сигнал в диапазоне от 0 до напряжения питания U_{ССА}. В режиме работы с внешней опорой Cfg_M_REFx=1 ЦАП формирует выходное напряжение в диапазоне от 0 до значения DACx_REF.

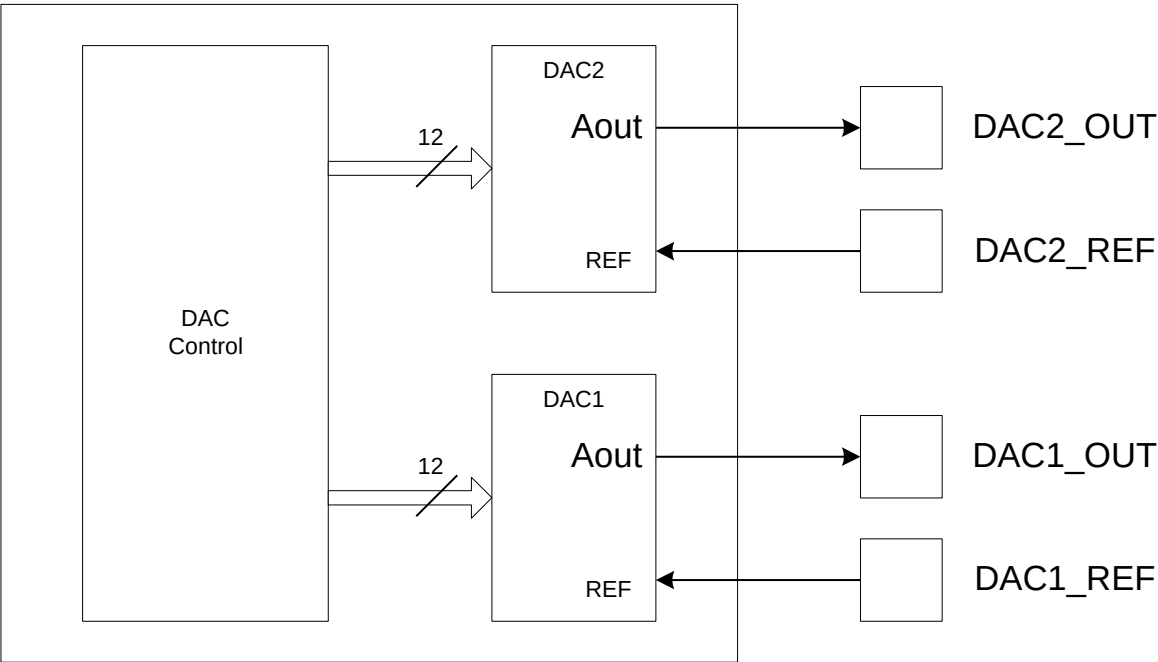


Рисунок 30-1 – Структурная схема контроллера ЦАП

30.1 Описание регистров блока контроллера ЦАП

Таблица 30-1 – Описание регистров блока контроллера ЦАП

Базовый Адрес	Название	Описание
0x4009_0000	MDR_DAC	Контроллер DAC
Смещение		
0x00	MDR_DAC->CFG	Регистр управления DAC
0x04	MDR_DAC->DAC1_DATA	Регистр данных DAC1
0x08	MDR_DAC->DAC2_DATA	Регистр данных DAC2

30.1.1 MDR_DAC->CFG

Таблица 30-2 – Регистр CFG

Номер	31...5	4	3	2	1	0
Доступ	U	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0
	-	Cfg_SYNC_A	Cfg_ON_DAC1	Cfg_ON_DAC0	Cfg_M_REF1	Cfg_M_REF0

Таблица 30-3 – Описание бит регистра CFG

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...5	-	Зарезервировано
4	Cfg_SYNC_A	Синхронизация DAC1 и DAC2: 0 – асинхронные; 1 – синхронные
3	Cfg_ON_DAC1	Включение DAC2: 1 – включен; 0 – выключен
2	Cfg_ON_DAC0	Включение DAC1: 1 – включен; 0 – выключен
1	Cfg_M_REF1	Выбор источника опорного напряжения DAC2: 0 – в качестве опорного напряжения используется напряжение питания с вывода U_{CCA} ; 1 – в качестве опорного напряжения используется напряжение на входе опорного напряжения DAC2_REF
0	Cfg_M_REF0	Выбор источника опорного напряжения DAC1: 0 – в качестве опорного напряжения используется напряжение питания с вывода U_{CCA} ; 1 – в качестве опорного напряжения используется напряжение на входе опорного напряжения DAC1_REF

30.1.2 MDR_DAC->DAC1_DATA

Таблица 30-4 – Регистр DAC1_DATA

Номер	31...28	27...16	15...12	11...0
Доступ	U	WO	U	R/W
Сброс	0	0	0	0
	-	DAC1_DATA[11:0]	-	DAC0_DATA[11:0]

Таблица 30-5 – Описание бит регистра DAC1_DATA

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...28	-	Зарезервировано
27...16	DAC1 DATA[11:0]	Данные DAC2 при Cfg_SYNC_A=1. При чтении всегда равны нулю. Читать из DAC2_DATA
15...12	-	Зарезервировано
11...0	DAC0 DATA[11:0]	Данные DAC1

30.1.3 MDR_DAC->DAC2_DATA

Таблица 30-6 – Регистр DAC2_DATA

Номер	31...28	27...16	15...12	11...0
Доступ	U	WO	U	R/W
Сброс	0	0	0	0
	-	DAC0_DATA[11:0]	-	DAC1_DATA[11:0]

Таблица 30-7 – Описание бит регистра DAC2_DATA

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...28	-	Зарезервировано
27...16	DAC0 DATA[11:0]	Данные DAC1 при Cfg_SYNC_A=1. При чтении всегда равны нулю. Читать из DAC1_DATA
15...12	-	Зарезервировано
11...0	DAC1 DATA[11:0]	Данные DAC2
Примечание – Если бит конфигурации Cfg_SYNC_A установлен, то данные для DAC1 и DAC2 задаются записью в один из регистров DACx_DATA		

31 Контроллер схемы компаратора MDR_COMP

В микросхеме реализована схема компаратора, обеспечивающая следующие режимы работы:

- сравнение двух сигналов с трех различных выводов микросхемы;
- сравнение сигнала с трех различных выводов с внутренней шкалой напряжений;
- сравнение сигнала с вывода IN1 с внутренним источником опорного напряжения;
- формирование внутренней шкалы напряжений от питания микросхемы и от внешних выводов.

Для включения компаратора необходимо установить бит ON в 1, используемые выводы порта E должны быть сконфигурированы как аналоговые и должны быть отключены какие-либо внутренние подтяжки. После появления флага Ready компаратор готов к работе.

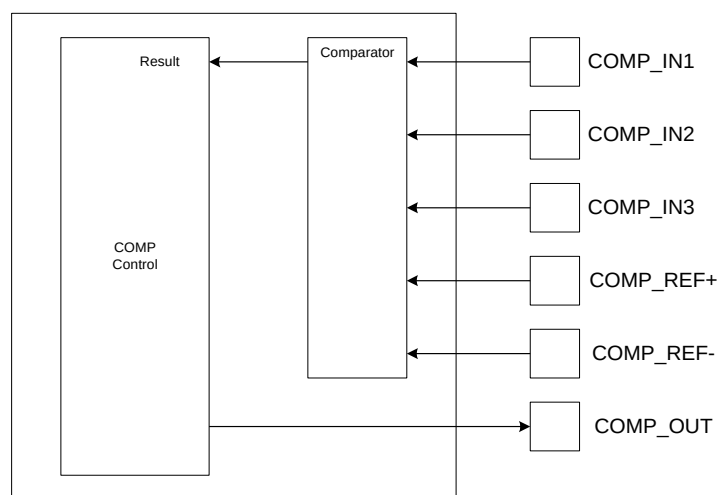
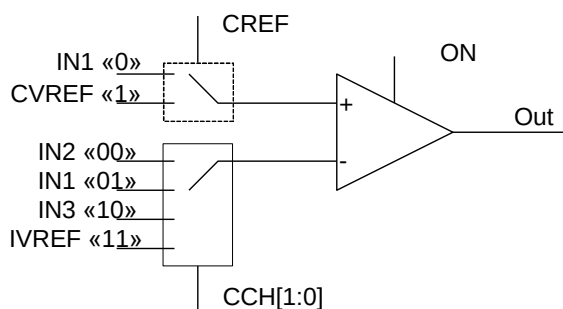


Рисунок 31-1 – Структура блока компаратора



*IVREF – выход внутреннего источника опорного напряжения 1,2 В.

Рисунок 31-2 – Структура мультиплексирования входов компаратора

31.1 Сравнение внешних сигналов

Компаратор позволяет проводить сравнение двух сигналов, поступающих с трех выводов микросхемы. На вход «+» компаратора может быть подан сигнал IN1 (бит CREF=0), на вход «-» могут быть поданы сигналы IN1 (CCH=01), IN2 (CCH=00) и IN2 (CCH=10), при этом, если уровень на входе «+» будет больше уровня на входе «-», то выход Out установится в «1».

31.2 Сравнение сигнала с внутренним источником опорного напряжения

Компаратор позволяет проводить сравнение сигнала, поступающего с вывода IN1 микросхемы, с внутренним источником опорного напряжения IVREF. Для этого на вход «+» компаратора должен быть подан сигнал IN1 (бит CREF=0), на вход «-» должен быть подан сигнал IVREF (CCH=11), при этом, если уровень на входе «+» будет больше уровня на входе «-», то выход Out установится в «1».

31.3 Сравнение внешних сигналов с внутренней шкалой напряжений

Компаратор позволяет проводить сравнение внешних сигналов, поступающих с трех выводов микросхемы, со шкалой напряжений, формируемых внутри микросхемы. На вход «+» компаратора должен быть подан сигнал CVREF (бит CREF=1), на вход «-» могут быть поданы сигналы IN1 (CCH=01), IN2 (CCH=00) и IN2 (CCH=10), при этом, если уровень на входе «+» будет больше уровня на входе «-», то выход Out установится в «1».

31.4 Формирование внутренней шкалы напряжений

Внутренняя шкала напряжений формируется на резистивном делителе (см. Рисунок 31-3), который включается битом CVREN=1.

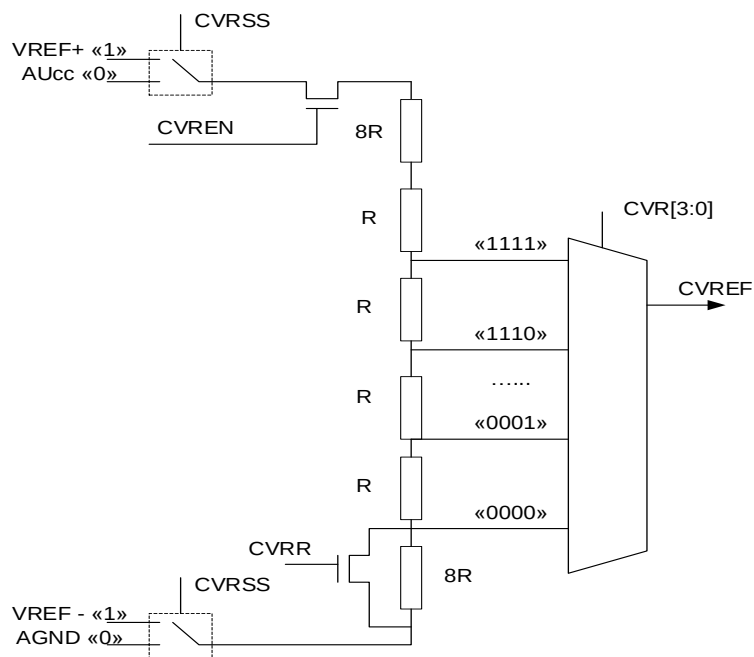


Рисунок 31-3 – Структура блока формирования CVREF

При этом в качестве опорного напряжения делителя может выступать питание микросхемы U_{CCA} (CVRSS = 0), либо напряжение на выводе COMP_VREF+ (CVRSS = 1). Нижнее опорное напряжение компаратора задается на выводе COMP_VREF-. Напряжение на выводе CVREF формируется на основании комбинации бит CVRR и CVR и приведено как справочные данные в таблице ниже (Таблица 31-1). Реальные значения в конкретном кристалле могут отличаться за счет технологического разброса параметров.

Таблица 31-1 – Формирование внутренней шкалы напряжений CVREF

CVRR	CVR[3:0]	Отношение резисторов	Напряжение CVREF при $U_{CC}=3,3\text{ В}$, В	Входной импеданс VREF+, кОм	Примечание
0	0000	8/32	0,83	12	
	0001	9/32	0,93	13	
	0010	10/32	1,03	13,8	
	0011	11/32	1,13	14,4	
	0100	12/32	1,24	15	
	0101	13/32	1,34	15,4	
	0110	14/32	1,44	15,8	
	0111	15/32	1,55	15,9	
	1000	16/32	1,65	16	
	1001	17/32	1,75	15,9	
	1010	18/32	1,86	15,8	
	1011	19/32	1,96	15,4	
	1100	20/32	2,06	15	
	1101	21/32	2,17	14,4	
	1110	22/32	2,27	13,8	
	1111	23/32	2,37	12,9	
1	0000	0/24	0,00	0,5	
	0001	1/24	0,14	1,9	
	0010	2/24	0,28	3,7	
	0011	3/24	0,41	5,3	
	0100	4/24	0,55	6,7	
	0101	5/24	0,69	7,9	
	0110	6/24	0,83	9	
	0111	7/24	0,96	9,9	
	1000	8/24	1,10	10,7	
	1001	9/24	1,24	11,3	
	1010	10/24	1,38	11,7	
	1011	11/24	1,51	11,9	
	1100	12/24	1,65	12	
	1101	13/24	1,79	11,9	
	1110	14/24	1,93	11,7	
	1111	15/24	2,06	11,3	

Результат работы компаратора сигнал Out может быть проинвертирован с помощью бита INV и выдан на вывод микросхемы OUT_COMP. Также результат сравнения доступен внутри микросхемы. Комбинационный сигнал с компаратора отображается в бите Rslt_As (при чтении может быть считан как 1, но при этом не выработать прерывания). Зафиксированный в триггере по тактовой частоте HCLK сигнал сравнения отображается в бите Rslt_Sy. Флаг Rst_lch фиксирует событие появления положительного сигнала сравнения и устанавливается в 1 до тех пор, пока не будет считан регистр COMP_RESULT_LATCH.

31.5 Описание регистров блока контроллера компаратора

Таблица 31-2 – Описание регистров блока контроллера компаратора

Базовый Адрес	Название	Описание
0x4009_8000	MDR_COMP	Контроллер компаратора
Смещение		
0x00	MDR_COMP->CFG	Регистр управления компаратора
0x04	MDR_COMP->RESULT	Регистр результата компаратора
0x08	MDR_COMP->RESULT_LATCH	Регистр результата компаратора - защелка

31.5.1 MDR_COMP->CFG

Таблица 31-3 – Регистр CFG

Номер	31...14	13	12	11	10, 9	8	7...4	3	2	1	0
Доступ	U	R/W	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0	0	0	0	0	0
	-	CMP IE	Ready	INV	CCH [1:0]	CREF	CVR [3:0]	CVR EN	CVRSS	CVRR	ON

Таблица 31-4 – Описание бит регистра CFG

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...14	-	Зарезервировано
13	CMP IE	Флаг разрешения генерации прерывания по событию Rst_lch: 0 – запрещено прерывание; 1 – разрешено прерывание
12	Ready	Сигнал готовности аналогового компаратора при включении: 0 – компаратор не включен или не готов к работе; 1 – компаратор готов к работе
11	INV	Инверсия выхода компаратора: 0 – прямой; 1 – инверсный
10, 9	CCH[1:0]	Биты выбора сигнал управления мультиплексора канала: 00 – на «-» компаратора сигнал подается с IN2; 01 – на «-» компаратора сигнал подается с IN1; 10 – на «-» компаратора сигнал подается с IN3; 11 – на «-» компаратора сигнал подается с выхода внутреннего источника опорного напряжения 1.2 В (IVREF).
8	CREF	Бит выбора сигнал управления мультиплексора канала: 0 – на «+» компаратора сигнал подается с IN1; 1 – на «+» компаратора сигнал подается с CVREF
7...4	CVR[3:0]	Биты выбора сигнал управления мультиплексора выбора CVREF. Смотрите Таблица 31-1 – Формирование внутренней шкалы напряжений CVREF
3	CVREN	Бит разрешения работы источника CVREF: 0 – не разрешен; 1 – разрешен

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
2	CVRSS	Бит выбора опоры CVREF: 0 – источник CVREF работает в границах $U_{CCA} GND$; 1 – источник CVREF работает в границах $V_{ref+} V_{ref-}$
1	CVRR	Бит выбора диапазона CVREF: 0 – источник CVREF работает в верхнем диапазоне; 1 – источник CVREF работает в нижнем диапазоне
0	ON	Включение компаратора: 0 – выключен; 1 – включен

31.5.2 MDR_COMP->RESULT

Таблица 31-5 – Регистр RESULT

Номер	31...3	2	1	0
Доступ	U	R/W	R/W	R/W
Сброс	0	0	0	0
	-	Rst_lch	Rslt_As	Rslt_Sy

Таблица 31-6 – Описание бит регистра RESULT

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...3	-	Зарезервировано
2	Rst_lch	Значение компарирования хранится до момента считывания из регистра COMP_RESULT_LATCH, после чего сбрасывается. Вводится по переднему фронту сигнала с компаратора
1	Rslt_As	Значение компарирования непосредственно из компаратора
0	Rslt_Sy	Значение результата компарирования, синхронизированное с частотой HCLK

31.5.3 MDR_COMP->RESULT_LATCH

Таблица 31-7 – Регистр RESULT_LATCH

Номер	31...1	0
Доступ	U	R/W
Сброс	0	0
	-	Rst_lch

Таблица 31-8 – Описание бит регистра RESULT_LATCH

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...1	-	Зарезервировано
0	Rst_lch	Значение компарирования хранится до момента считывания из регистра COMP_RESULT_LATCH, после чего сбрасывается. Вводится по переднему фронту сигнала с компаратора

32 Контроллер интерфейса MDR_I2C

I2C является двухпроводным, двунаправленным последовательным каналом связи с простым и эффективным методом обмена данными между устройствами. Интерфейс применяется, когда надо организовать обмен на коротком расстоянии между несколькими устройствами. Стандарт интерфейса I2C является многомастерным с обнаружением коллизий и арбитражем, исключающим потерю данных при обмене, когда два или более мастера пытаются осуществить передачу одновременно.

Интерфейс работает на трех скоростях:

- нормальная: 100 Кбит/с;
- быстрая: 400 Кбит/с;
- очень быстрая: 3,5 Мбит/с.

Приблизительная скорость обмена данными блока I2C рассчитывается по формуле:

$$F_{SCI} = \frac{HCLK}{5 \cdot (DIV + 1)}. \quad (6)$$

Более точное значение скорости обмена можно установить опытным путем, значение предделителя DIV настраивается в регистрах PRL (младшая часть) и PRH (старшая часть) – если примерно рассчитанное значение предделителя DIV не задействует регистр PRH, то он должен быть равен нулю.

Контроллер интерфейса I2C работает только в режиме Master.

32.1 Конфигурация системы

I2C системы используют последовательную линию данных SDA и линию тактового сигнала SCL. Все устройства, подсоединенные к этим двум линиям, должны работать в режиме открытого стока, обеспечивая тем самым создание на линии «проводного И» за счет внешних резисторов подтяжки обеих линий к питанию.

Передача данных между мастером и ведомым осуществляется по линии SDA и синхронизируется по линии SCL. После завершения передачи информации осуществляется передача в обратную сторону одного бита подтверждения. Каждый принимаемый бит фиксируется принимающей стороной при высоком уровне SCL и может изменяться передатчиком при низком уровне. Изменение линии SDA при высоком уровне SCL является командным состоянием (см. подразделы 32.3 «Сигнал START» и 32.6 «Сигнал STOP»).

32.2 Протокол I2C

Нормальная передача по интерфейсу I2C содержит четыре фазы:

- сигнал START;
- передача адреса;
- передача данных;
- сигнал STOP.

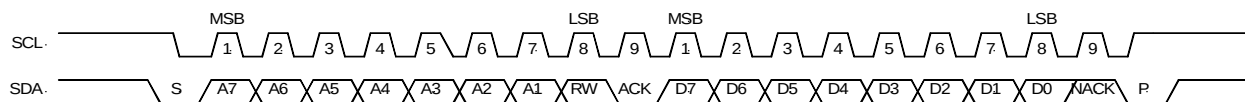


Рисунок 32-1 – Передача по I2C

32.3 Сигнал START

Когда шина находится в свободном состоянии, т.е. не одно из устройств не осуществляет передачи (на линиях SCL и SDA высокий уровень), мастер может инициализировать процесс передачи через создание сигнала START на линии. Сигнал START или S бит задается, когда уровень на линии SDA переходит из высокого в низкий при высоком уровне на линии SCL. Появление сигнала START не означает начала передачи данных.

Повторный сигнал START является обычным сигналом START, но без предварительно сгенерированного до этого сигнала STOP. Мастер может использовать метод для начала соединения с другим ведомым или с тем же ведомым, но с изменением режима работы (например, чтение после записи, или, наоборот) без перевода шины в свободное состояние.

Контроллер интерфейса генерирует сигнал START при записи единицы в бит START регистра I2C_CMD при установленных битах RD или WR. В зависимости от состояния линии SCL генерируется либо сигнал START, либо повторный сигнал START.

32.4 Передача адреса

Первым байтом данных, передаваемым мастером сразу после сигнала START, является адрес ведомого. Это семибитный адрес и следующий за ним бит RW. Бит RW определяет дальнейшее направление передачи данных. В системе не может быть несколько ведомых устройств с одним адресом. Ведомое устройство, у которого совпадает адрес с адресом в сообщении, подтверждает прием, выставив ACK и опуская линию SDA в низкий уровень на 9-й SCL тактовый импульс. Контроллер также поддерживает 10-битный адрес путем генерации двух циклов передачи адреса.

Процесс выдачи адреса выполняется как цикл записи. Необходимо записать адрес ведомого в регистр I2C_TXD и установить бит WR в регистре I2C_CMD. Контроллер осуществит передачу адреса в линию.

32.5 Передача данных

После успешного подтверждения приема адреса одним ведомым устройством может быть начата передача данных в направлении, задаваемым битом RW в послышке мастера. Каждый передаваемый бит подтверждается ACK на 9-й SCL тактовый импульс. Если ведомое устройство выдало NACK (нет подтверждения), то мастер может сгенерировать либо сигнал STOP для прекращения передачи, либо повторный сигнал START для начала нового цикла передачи.

Если мастер является принимающим устройством и выдает NACK, то ведомое устройство отпускает линию SDA и мастер может сгенерировать сигнал STOP или повторный сигнал START.

Для записи данных в ведомое устройство запишите данные в регистр I2C_TXD и установите бит WR. Для чтения данных из устройства установите бит RD. На время выполнения передачи контроллер интерфейса выставляет флаг TR_PROG в регистре I2C_STA. Когда передача завершена, этот флаг снимается и устанавливается флаг INT. Если при этом установлен бит разрешения INT_EN, то генерируется прерывание контроллеру прерываний. Регистр I2C_RXD содержит корректные принятые данные после установки флага INT. Пользователь может начать новый цикл чтения или записи только тогда, когда флаг TR_PROG сброшен.

32.6 Сигнал STOP

Мастер может завершить соединение путем создания сигнала STOP. Сигнал STOP или Р бит определяется переходом линии SDA из низкого состояния в высокое, когда SCL находится в высоком состоянии.

32.7 Описание регистров контроллера I2C

Таблица 32-1 – Описание регистров контроллера I2C

Базовый Адрес	Название	Описание
0x4005_0000	MDR_I2C	Контроллер I2C
Смещение		
0x00	MDR_I2C->PRL	Младшая часть предделителя частоты
0x04	MDR_I2C->PRH	Старшая часть предделителя частоты
0x08	MDR_I2C->CTR	Управление контроллером I2C
0x0C	MDR_I2C->RXD	Принятые данные по I2C
0x10	MDR_I2C->STA	Статус I2C
0x14	MDR_I2C->TXD	Передаваемые данные по I2C
0x18	MDR_I2C->CMD	Управление I2C

32.7.1 MDR_I2C->PRL

Таблица 32-2 – Регистр PRL

Номер	31...8	7... 0
Доступ	U	R/W
Сброс	0	0
	-	PR[7:0]

Таблица 32-3 – Описание бит регистра PRL

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...8	-	Зарезервировано
7...0	PR[7:0]	Младшая часть предделителя

32.7.2 MDR_I2C->PRH

Таблица 32-4 – Регистр PRH

Номер	31...8	7... 0
Доступ	U	R/W
Сброс	0	0
	-	PR[15:8]

Таблица 32-5 – Описание бит регистра PRH

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...8	-	Зарезервировано
7...0	PR[15:8]	Старшая часть предделителя

32.7.3 MDR_I2C->CTR

Таблица 32-6 – Регистр CTR

Номер	31...8	7	6	5	4...0
Доступ	U	R/W	R/W	R/W	U
Сброс	0	0	0	0	0
	-	EN_I2C	EN_INT	S_I2C	-

Таблица 32-7 – Описание бит регистра CTR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...8	-	Зарезервировано
7	EN_I2C	Разрешение работы контроллера I2C: 0 – выключен; 1 – включен
6	EN_INT	Разрешение прерывания от I2C: 0 – запрещено; 1 – разрешено
5	S_I2C	Скорость работы I2C: 0 – до 400 кГц; 1 – до 1 МГц
4...0	-	Зарезервировано

32.7.4 MDR_I2C->RXD

Таблица 32-8 – Регистр RXD

Номер	31...8	7... 0
Доступ	U	RO
Сброс	0	0
	-	RXD[7:0]

Таблица 32-9 – Описание бит регистра RXD

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...8	-	Зарезервировано
7...0	RXD[7:0]	Последний полученный по I2C байт

32.7.5 MDR_I2C->STA

Таблица 32-10 – Регистр STA

Номер	31...8	7	6	5	4...2	1	0
Доступ	U	RO	RO	RO	U	RO	RO
Сброс	0	0	0	0	0	0	0
	-	Rx ACK	BUSY	LOST ARB	-	TR PROG	INT

Таблица 32-11 – Описание бит регистра STA

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...8	-	Зарезервировано
7	Rx ACK	Полученный от ведомого ACK: 0 – ACK получен; 1 – получен NACK
6	BUSY	Состояние шины I2C: 0 – после получения Stop bit; 1 – после получения состояния Start bit
5	LOST ARB	Потеря арбитража: 0 – нет потери арбитража; 1 – потерян арбитраж. Этот бит выставляется если: – получен Stop bit, но он не был инициализирован этим контроллером; – контроллер пытается выставить SDA в высокий уровень, но SDA остается в низком
4...2	-	Зарезервировано
1	TR PROG	Процесс передачи: 0 – передача завершена; 1 – передаются данные
0	INT	Флаг прерывания, выставляется всегда. Прерывание для процессора выдается, если есть флаг EN_INT: 0 – нет прерывания; 1 – есть прерывание. Флаг выставляется если: – передача байта завершена; – был потерян арбитраж

32.7.6 MDR_I2C->TXD

Таблица 32-12 – Регистр TXD

Номер	31...8	7... 0
Доступ	U	R/W
Сброс	0	0
	-	TXD[7:0]

Таблица 32-13 – Описание бит регистра TXD

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...8	-	Зарезервировано
7...0	TXD[7:0]	Байт для отправки по I2C. При передаче адреса нулевой бит определяет режим передачи: 0 – запись в ведомое устройство; 1 – чтение из ведомого устройства

32.7.7 MDR_I2C->CMD

Таблица 32-14 – Регистр CMD

Номер	31...8	7	6	5	4	3	2...1	0
Доступ	U	R/W	R/W	R/W	R/W	R/W	U	R/W
Сброс	0	0	0	0	0	0	0	0
	-	START	STOP	RD	WR	ACK	-	CLR INT

Таблица 32-15 – Описание бит регистра CMD

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...8	-	Зарезервировано
7	START	Отправить START bit. Инициализируется записью 1. После завершения отправки автоматически не сбрасывается, очищается записью нуля
6	STOP	Отправить STOP bit. Инициализируется записью 1. После завершения отправки автоматически не сбрасывается, а очищается записью нуля
5	RD	Чтение из ведомого: 0 – нет действия; 1 – начать чтение
4	WR	Запись в ведомого; 0 – нет действия; 1 – начать запись
3	ACK	Отправить ACK при чтении: 0 – отправить ACK; 1 – отправить NACK
2...1	-	Зарезервировано
0	CLR INT	Очистить прерывание INT. Запись 1 очищает прерывание

33 Контроллер MDR_SSP

Модуль порта синхронной последовательной связи (SSP – Synchronous Serial Port) выполняет функции интерфейса последовательной синхронной связи в режиме ведущего и ведомого устройства и обеспечивает обмен данными с подключенным ведомым или ведущим периферийным устройством в соответствии с одним из протоколов:

- интерфейс SPI фирмы Motorola;
- интерфейс SSI фирмы Texas Instruments;
- интерфейс Microwire фирмы National Semiconductor.

Как в ведущем, так и в ведомом режиме работы модуль SSP обеспечивает:

- преобразование данных, размещенных во внутреннем буфере FIFO передатчика (восемь 16-разрядных ячеек данных) из параллельного в последовательный формат;
- преобразование данных из последовательного в параллельный формат и их запись в аналогичный буфер FIFO приемника (восемь 16-разрядных ячеек данных).

Модуль формирует сигналы прерываний по следующим событиям:

- необходимость обслуживания буферов FIFO приемника и передатчика;
- переполнение буфера FIFO приемника;
- наличие данных в буфере FIFO приемника по истечении времени таймаута.

Основные сведения о модуле представлены в следующих разделах:

- характеристики интерфейса SPI;
- характеристики интерфейса Microwire;
- характеристики интерфейса SSI.

33.1 Основные характеристики модуля SSP

- функционирование как в ведущем, так и в ведомом режиме;
- программное управление скоростью обмена;
- состоит из независимых буферов приема и передачи (8 ячеек по 16 бит) с организацией доступа типа FIFO (First In First Out – первый вошел, первый вышел);
- программный выбор одного из интерфейсов обмена: SPI, Microwire, SSI;
- программируемая длительность информационного кадра от 4 до 16 бит;
- независимое маскирование прерываний от буфера FIFO передатчика, буфера FIFO приемника, а также по переполнению буфера приемника;
- доступна возможность тестирования по шлейфу, соединяющему вход с выходом;
- поддержка прямого доступа к памяти (DMA).

Структурная схема модуля представлена далее – см. Рисунок 33-1.

ТСКЯ.431296.024СП

708

- полнодуплексный обмен данными по четырехпроводной линии;
- программное задание фазы и полярности тактового сигнала.

33.4 Характеристики интерфейса Microwire

Интерфейс Microwire фирмы National Semiconductor обеспечивает:

- полудуплексный обмен данными с использованием восьмибитных управляющих последовательностей.

33.5 Характеристики интерфейса SSI

Интерфейс SSI фирмы Texas Instruments обеспечивает:

- полнодуплексный обмен данными по четырехпроводной линии;
- возможность перевода линии передачи данных в третье (высокоимпедансное) состояние.

33.6 Общий обзор модуля SSP

Модуль SSP представляет собой интерфейс синхронного последовательного обмена данными, способный функционировать в качестве ведущего или ведомого устройства и поддерживающий протоколы передачи данных SPI фирмы Motorola, Microwire фирмы National Semiconductor, а также SSI фирмы Texas Instruments.

Модуль выполняет следующие функции:

- преобразование данных, полученных от периферийного устройства, из последовательной в параллельную форму;
- преобразование данных, передаваемых на периферийное устройство, из параллельной и последовательную форму;
- центральный процессор читает и записывает данные, а также управляющую информацию и информацию о состоянии;
- прием и передача данных буферизуются с помощью буферов FIFO, обеспечивающих хранение до восьми слов данных шириной 16 бит независимо для режимов приема и передачи.

Последовательные данные передаются по линии SSP_TXD и принимаются с линии SSP_RXD.

Модуль SSP содержит программируемые делители частоты, формирующие тактовый сигнал обмена данными SSP_CLK из сигнала, поступающего на линию SSPCLK. Скорость передачи данных может достигать более 2 МГц, в зависимости от частоты SSPCLK и характеристик подключенного периферийного устройства.

Режим обмена данными, формат информационного кадра и количество бит данных задаются программно с помощью регистров управления CR0 и CR1.

Модуль формирует четыре независимо маскируемых прерывания:

SSPTXINTR – запрос на обслуживание буфера передатчика;

SSPRXINTR – запрос на обслуживание буфера приемника;

SSPRORINTR – переполнение приемного буфера FIFO;

SSPRTINTR – таймаут ожидания чтения данных из приемного FIFO.

Кроме того, формируется общий сигнал прерывания SSPINTR, возникающий в случае активности одного из вышеуказанных независимых немаскированных прерываний, который идет на контроллер NVIC.

Модуль также формирует сигналы запроса на прямой доступ к памяти (DMA) для совместной работы с контроллером DMA.

В зависимости от режима работы модуля сигнал SSPFSSOUT используется либо для кадровой синхронизации (интерфейс SSI, активное состояние – высокий уровень), либо для выбора ведомого режима (интерфейсы SPI и Microwire, активное состояние – низкий уровень).

33.6.1 Блок формирования тактового сигнала

В режиме ведущего устройства модуль формирует тактовый сигнал обмена данными SSP_CLK с помощью внутреннего делителя частоты, состоящего из двух последовательно соединенных счетчиков без цепи сброса.

Путем записи значения в регистр SSPCPSR можно задать коэффициент предварительного деления частоты в диапазоне от 2 до 254 с шагом 2. Так как младший значащий разряд коэффициента деления не используется, то исключается возможность деления частоты на нечетный коэффициент деления. Это, в свою очередь, гарантирует формирование тактового сигнала симметричной формы (с одинаковой длительностью полупериодов высокого и низкого уровней).

Сформированный описанным образом сигнал далее поступает на второй делитель частоты, с выхода которого и снимается тактовый сигнал обмена данными SSP_CLK.

Коэффициент деления второго делителя задается программно в диапазоне от 1 до 256, путем записи соответствующего значения в регистр управления SSPCR0.

33.6.2 Буфер FIFO передатчика

Буфер передатчика имеет ширину 16 бит, глубину 8 слов, схему организации доступа типа FIFO – «первый вошел, первый вышел». Данные от центрального процессора сохраняются в буфере до тех пор, пока не будут считаны блоком передачи данных.

33.6.3 Буфер FIFO приемника

Буфер приемника имеет ширину 16 бит, глубину 8 слов, схему организации доступа типа FIFO – «первый вошел, первый вышел». Принятые от периферийного устройства данные сохраняются в этом буфере блоком приема данных до тех пор, пока не будут считаны центральным процессором.

33.6.4 Блок приема и передачи данных

В режиме ведущего устройства модуль формирует тактовый сигнал обмена данными SSP_CLK для подключенных ведомых устройств. Как было описано ранее, данный сигнал формируется путем деления частоты сигнала SSPCLK.

Блок передатчика последовательно считывает данные из буфера FIFO передатчика и производит их преобразование из параллельной формы в последовательную. Далее поток последовательных данных и элементов кадровой

синхронизации, тактированный сигналом SSP_CLK, передается по линии SSP_TXD к подключенным ведомым устройствам.

Блок приемника выполняет преобразование данных, поступающих синхронно с линии SSP_RXD, из последовательной в параллельную форму, после чего загружает их в буфер FIFO приемника, откуда они могут быть считаны процессором.

В режиме ведомого устройства тактовый сигнал обмена данными формируется одним из подключенных к модулю периферийных устройств и поступает по линии SSP_CLK.

При этом блок передатчика, тактируемый этим внешним сигналом, считывает данные из буфера FIFO, преобразует их из параллельной формы в последовательную, после чего выдает поток последовательных данных и элементов кадровой синхронизации в линию SSP_TXD.

Аналогично, блок приемника выполняет преобразование данных, поступающих с линии SSP_RXD синхронно с сигналом SSP_CLK, из последовательной в параллельную форму, после чего загружает их в буфер FIFO приемника, откуда они могут быть считаны процессором.

33.6.5 Блок формирования прерываний

Модуль SSP генерирует независимые маскируемые прерывания с активным высоким уровнем. Кроме того, формируется комбинированное прерывание путем объединения указанных независимых прерываний по схеме ИЛИ.

Комбинированный сигнал прерывания подается на контроллер прерываний NVIC, при этом появляется дополнительная возможность маскирования устройства в целом, что облегчает построение модульных драйверов устройств.

33.6.6 Интерфейс прямого доступа к памяти

Модуль обеспечивает интерфейс с контроллером DMA согласно схеме взаимодействия приемопередатчика и контроллера DMA.

33.6.7 Конфигурирование приемопередатчика

После сброса работа блоков приемопередатчика запрещается до выполнения процедуры задания конфигурации.

Для этого необходимо выбрать ведущий или ведомый режим работы устройства, а также используемый протокол передачи данных (SPI фирмы Motorola, SSI фирмы Texas Instruments, либо Microwave фирмы National Semiconductor), после чего записать необходимую информацию в регистры управления CR0 и CR1.

Кроме того, для установки требуемой скорости передачи данных необходимо выбрать параметры блока формирования тактового сигнала с учетом значения частоты сигнала SSPCLK и записать соответствующую информацию в регистр PSR.

33.6.8 Разрешение работы приемопередатчика

Разрешение осуществляется путем установки бита SSE регистра управления CR1. Буфер FIFO передатчика может быть либо проинициализирован путем записи в него до

восемью 16-разрядных слов заблаговременно перед установкой этого бита, либо может заполняться передаваемыми данными в процедуре обслуживания прерывания.

После разрешения работы модуля приемопередатчик начинает обмен данными по линиям SSP_TXD и SSP_RXD.

33.6.9 Соотношения между тактовыми сигналами

В модуле имеется ограничение на соотношение между частотами тактовых сигналов CPU_CLK и SSPCLK. Частота SSPCLK должна меньше или равна частоте CPU_CLK. Выполнение этого требования гарантирует синхронизацию сигналов управления, передаваемых из зоны действия тактового сигнала SSPCLK в зону действия сигнала CPU_CLK в течение времени, меньшего продолжительности передачи одного информационного кадра:

$$F_{SSPCLK} \leq F_{PCLK}$$

В режиме ведомого устройства сигнал SSP_CLK от ведущего внешнего устройства поступает на схемы синхронизации, задержки и обнаружения фронта. Для того, чтобы обнаружить фронт сигнала SSP_CLK, необходимо три такта сигнала SSP_CLK. Сигнал SSP_TXD имеет меньшее время установки по отношению к заднему фронту SSP_CLK, по которому и происходит считывание данных из линии. Время установки и удержания сигнала SSP_RXD по отношению к сигналу SSP_CLK должно выбираться с запасом, гарантирующим правильное считывание данных. Для обеспечения корректной работы устройства необходимо, чтобы частота SSPCLK была как минимум в 12 раз больше, чем максимальная предполагаемая частота сигнала SSP_CLK.

Выбор частоты тактового сигнала SSPCLK должен обеспечивать поддержку требуемого диапазона скоростей обмена данными. Отношение минимальной частоты сигнала SSPCLK к максимальной частоте сигнала SSP_CLK в режиме ведомого устройства равно 12, в режиме ведущего – двум.

Так, в режиме ведущего устройства для обеспечения максимальной скорости обмена 1,8432 Мбит/с частота сигнала SSPCLK должна составлять не менее 3,6864 МГц. В этом случае в регистр CPSR должно быть записано значение 2, а поле SCR[7:0] регистра CR0 должно быть установлено в 0.

В режиме ведомого устройства для обеспечения той же информационной скорости необходимо использовать тактовый сигнал SSPCLK с частотой не менее 22,12 МГц. При этом в регистр CPSR должно быть записано значение 12, а поле SCR[7:0] регистра CR0 должно быть установлено в 0.

Соотношение между максимальной частотой сигнала SSPCLK и минимальной частотой SSPCLKOUT составляет $254 * 256$.

Минимальная допустимая частота сигнала SSPCLK определяется следующей системой соотношений, которые должны выполняться одновременно:

$$F_{SSPCLK}(\min) \Rightarrow 2 \times F_{SSPCLKOUT}(\max) \text{ [for master mode]}$$

$$F_{SSPCLK}(\min) \Rightarrow 12 \times F_{SSPCLKIN}(\max) \text{ [for slave mode].}$$

Аналогично, максимальная допустимая частота сигнала SSPCLK определяется следующей системой соотношений, которые должны выполняться одновременно:

$$FSSPCLK(max) \leq 254 \times 256 \times FSSPCLKOUT(min) \text{ [for master mode]}$$

$$FSSPCLK(max) \leq 254 \times 256 \times FSSPCLKIN(min) \text{ [for slave mode]}.$$

33.6.10 Программирование регистра управления CR0

Регистр CR0 предназначен для:

- установки скорости информационного обмена;
- выбора одного из трех протоколов обмена данными;
- выбора размера слова данных.

Скорость информационного обмена зависит от частоты внешнего тактового сигнала SSPCLK и коэффициента деления блока формирования тактового сигнала. Последний задается совместно значением поля SCR (Serial Clock Rate – скорость информационного обмена) регистра SSPCR0 и значением поля CPSDVSR (clock prescale divisor value – коэффициент деления тактового сигнала) регистра SSPCPSR.

Формат информационного кадра задается путем установки значения поля FRF, а размер слова данных – путем установки значения поля DSS регистра SSPCR0.

Для протокола SPI фирмы Motorola также задаются полярность и фаза сигнала (биты SPH и SPO).

33.6.11 Программирование регистра управления CR1

Регистр SSPCR1 предназначен для:

- выбора ведущего или ведомого режима функционирования приемопередатчика;
- включения режима проверки канала по шлейфу;
- разрешения или запрещения работы модуля.

Выбор ведущего режима осуществляется путем записи 0 в поле MS регистра SSPCR1 (это значение устанавливается после сброса автоматически).

Запись 1 в поле MS переводит приемопередатчик в режим ведомого устройства. В этом режиме разрешение или запрещение формирования сигнала передатчика SSP_TXD осуществляется путем установки бита SOD (slave mode SSP_TXD output disable – запрет линии SSP_TXD для ведомого режима) регистра CR1. Указанная функция полезна при подключении к одной линии нескольких подчиненных устройств.

Для того, чтобы разрешить функционирование приемопередатчика, необходимо установить в 1 бит SSE (Synchronous Serial Port Enable – разрешение последовательного синхронного порта).

33.6.12 Формирование тактового сигнала обмена данными

Тактовый сигнал обмена данными формируется путем деления частоты тактового сигнала SSPCLK. На первом этапе формирования частота этого сигнала делится на четный коэффициент CPSDVSR, лежащий в диапазоне от 2 до 254, доступный для программирования через регистр CPSR. Сформированный сигнал далее поступает на

делитель частоты с коэффициентом $(1 + SCR)$ от 1 до 256, где значение SCR доступно для программирования через CR0.

Частота выходного тактового сигнала обмена данными SSP_CLK определяется следующим соотношением:

$$FSSPCLKOUT = FSSPCLK / (CPSDVR \cdot (1+SCR)).$$

Например, в случае, если частота сигнала SSPCLK составляет 3,6864 МГц, а значение CPSDVR = 2, частота сигнала SSP_CLK лежит в интервале от 7,2 кГц до 1,8432 МГц.

33.6.13 Формат информационного кадра

Каждый информационный кадр содержит в зависимости от запрограммированного значения от 4 до 16 бит данных. Передача данных начинается со старшего значащего разряда. Возможно выбрать три базовых структуры построения кадра:

- SSI фирмы Texas Instruments;
- SPI фирмы Motorola;
- Microwire фирмы National Semiconductor.

Во всех трех режимах построения кадра тактовый сигнал SSP_CLK формируется только тогда, когда приемопередатчик готов к обмену данными. Перевод сигнала SSP_CLK в неактивное состояние используется как признак таймаута приемника, то есть наличия в буфере приемника необработанных данных по истечении заданного интервала времени.

В режимах SPI и Microwire выходной сигнал кадровой синхронизации передатчика SSP_FSS имеет активный низкий уровень и поддерживается в низком уровне в течение всего периода передачи информационного кадра.

В режиме построения кадра SSI фирмы Texas Instruments перед началом каждого информационного кадра на выходе SSP_FSS формируется импульс с длительностью, равной одному тактовому интервалу обмена данными. В этом режиме приемопередатчик SSP, равно как и ведомые периферийные устройства, передает данные в линию по переднему фронту сигнала SSP_CLK, а считывает данные из линии по заднему фронту этого сигнала.

В отличие от полнодуплексных режимов передачи данных SSI и SPI, режим Microwire фирмы National Semiconductor использует специальный способ обмена данными между ведущим и ведомым устройством, функционирующий в режиме полудуплекса. В указанном режиме на внешнее ведомое устройство перед началом передачи информационного кадра посылается специальная восьмибитная управляющая последовательность. В течение всего времени передачи этой последовательности приемник не обрабатывает каких-либо входных данных. После того как сигнал передан и декодирован ведомым устройством, оно выдерживает паузу в один тактовый интервал после передачи последнего бита управляющей последовательности, после чего передает в адрес ведущего устройства запрошенные данные. Длительность блока данных от ведомого устройства может составлять от 4 до 16 бит, таким образом общая длительность информационного кадра составляет от 13 до 25 бит.

33.6.14 Формат синхронного обмена SSI фирмы Texas Instruments

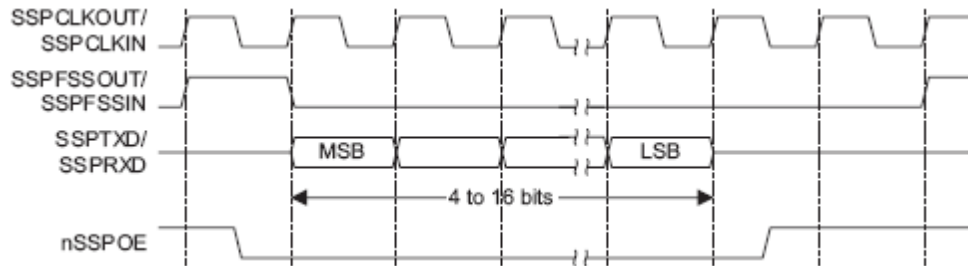


Рисунок 33-2 – Формат синхронного обмена протокола SSI (единичный обмен)

В данном режиме при неактивном приемопередатчике SSP сигналы SSP_CLK и SSP_FSS переводятся в низкий логический уровень, а линия передачи данных SSP_TXD поддерживается в третьем состоянии.

После появления хотя бы одного элемента в буфере FIFO передатчика сигнал SSP_FSS переводится в высокий логический уровень на время, соответствующее одному периоду сигнала SSP_CLK. Значение из буфера FIFO при этом переносится в сдвиговый регистр блока передатчика. По следующему переднему фронту сигнала SSP_CLK старший значащий разряд информационного кадра (4–16 бит данных) выдается на выход линии SSP_TXD и т.д.

В режиме приема данных как модуль SSP, так и ведомое внешнее устройство последовательно загружают биты данных в сдвиговый регистр по заднему фронту сигнала SSP_CLK. Принятые данные переносятся из сдвигового регистра в буфер FIFO после загрузки в него младшего значащего бита данных по очередному переднему фронту сигнала SSP_CLK.

Временные диаграммы последовательного синхронного обмена по протоколу SSI фирмы Texas Instruments представлены на рисунках ниже (Рисунок 33-2, Рисунок 33-3).

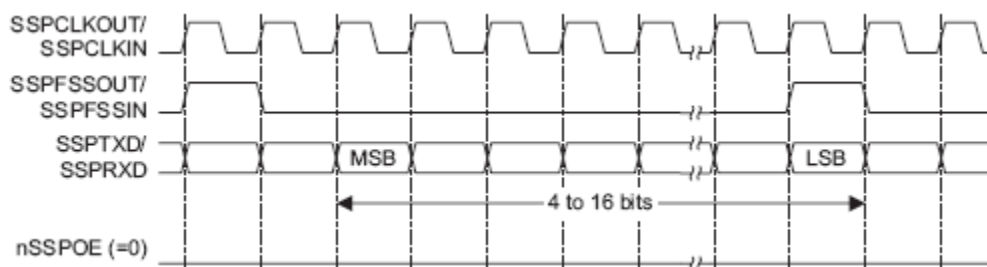


Рисунок 33-3 – Формат синхронного обмена протокола SSI (непрерывный обмен)

33.6.15 Формат синхронного обмена SPI фирмы Motorola

Интерфейс SPI фирмы Motorola осуществляется по четырем сигнальным линиям, при этом сигнал SSP_FSS выполняет функцию выбора ведомого устройства. Главной особенностью протокола SPI является возможность выбора состояния и фазы сигнала SSP_CLK в режиме ожидания (неактивном приемопередатчике) путем задания значений бит SPO и SPH регистра управления SSPSCR0.

Выбор полярности тактового сигнала – бит SPO

Если бит SPO равен 0, то в режиме ожидания линия SSP_CLK переводится в низкий логический уровень. В противном случае при отсутствии обмена данными линия SSP_CLK переводится в высокий логический уровень.

Выбор фазы тактового сигнала – бит SPH

Значение бита SPH определяет фронт тактового сигнала, по которому осуществляется выборка данных и изменение состояния на выходе линии.

В случае, если бит SPH установлен в 0, регистрация данных приемником осуществляется после первого обнаружения фронта тактового сигнала, в противном случае – после второго.

33.6.16 Формат синхронного обмена SPI фирмы Motorola, SPO=0, SPH=0

Временные диаграммы последовательного синхронного обмена в режиме SPI с SPO=0, SPH=0 показаны на рисунках (Рисунок 33-4, Рисунок 33-5).

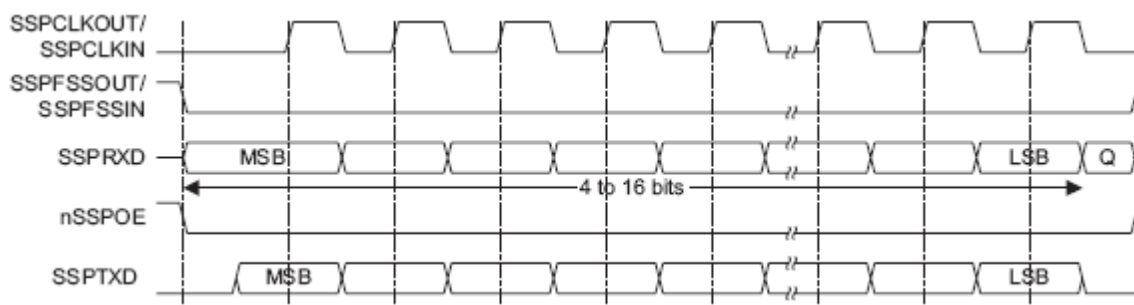


Рисунок 33-4 – Формат синхронного обмена протокола SPI, SPO=0, SPH=0 (одиночный обмен)

Примечание – На рисунке буквой Q обозначен сигнал с неопределенным уровнем.

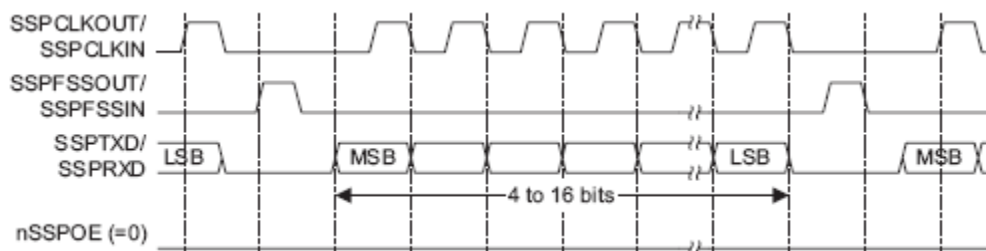


Рисунок 33-5 – Формат синхронного обмена протокола SPI, SPO=0, SPH=0 (непрерывный обмен)

В данном режиме во время ожидания приемопередатчика:

- сигнал SSP_CLK имеет низкий логический уровень;
- сигнал SSP_FSS имеет высокий логический уровень;
- сигнал SSP_TXD переводится в высокоимпедансное состояние.

Если работа модуля разрешена и в буфере FIFO передатчика содержатся корректные данные, сигнал SSP_FSS переводится в низкий логический уровень, что указывает на начало обмена данными и разрешает передачу данных от ведомого устройства на входную линию SSP_RXD ведущего. Контакт передатчика SSPTXD переходит из высокоимпедансного в активное состояние.

По истечении полутакта сигнала SSP_CLK на линии SSP_TXD формируется значение первого бита передаваемых данных. К этому моменту должны быть сформированы данные на линиях обмена, как ведущего, так и ведомого устройства. По истечении следующего полутакта сигнал SSP_CLK переводится в высокий логический уровень.

Далее данные регистрируются по переднему фронту и выдаются в линию по заднему фронту сигнала SSP_CLK.

В случае передачи одного слова данных после приема его последнего бита линия SSP_FSS переводится в высокий логический уровень по истечении одного периода тактового сигнала SSP_CLK.

В режиме непрерывной передачи данных на линии SSP_FSS должны формироваться импульсы высокого логического уровня между передачами каждого из слов данных. Это связано с тем, что в режиме SPH=0 линия выбора ведомого устройства в низком уровне блокирует запись в сдвиговый регистр. Поэтому ведущее устройство должно переводить линию SSP_FSS в высокий уровень по окончании передачи каждого кадра, разрешая таким образом запись новых данных. По окончании приема последнего бита блока данных линия SSP_FSS переводится в состояние, соответствующее режиму ожидания, по истечении одного такта сигнала SSP_CLK.

33.6.17 Формат синхронного обмена SPI фирмы Motorola, SPO=0, SPH=1

Временные диаграммы последовательного синхронного обмена в режиме SPI с SPO=0, SPH=1 показывает Рисунок 33-6 – одиночный и непрерывный обмен.

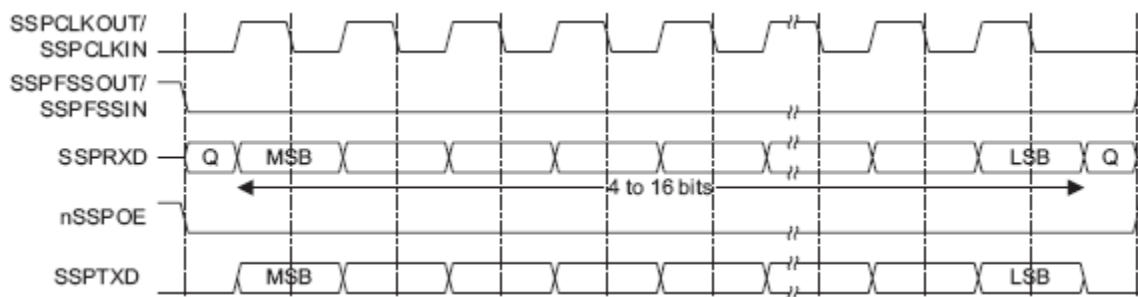


Рисунок 33-6 – Формат синхронного обмена протокола SPI, SPO=0, SPH=1

Примечание – На рисунке буквой Q обозначен сигнал с неопределенным уровнем.

В данном режиме во время ожидания приемопередатчика:

- сигнал SSP_CLK имеет низкий логический уровень;
- сигнал SSP_FSS имеет высокий логический уровень;
- сигнал SSP_TXD переводится в высокоимпедансное состояние.

Если работа модуля разрешена и в буфере FIFO передатчика содержатся корректные данные, сигнал SSP_FSS переводится в низкий логический уровень, что указывает на начало обмена данными и разрешает передачу данных от ведомого устройства на входную линию SSP_RXD ведущего. Выходной контакт передатчика SSPTXD переходит из высокоимпедансного в активное состояние.

По истечении полутака сигнала SSP_CLK на линиях обмена, как ведущего, так и ведомого устройств будут сформированы значения первых бит передаваемых данных. В это же время включается линия SSP_CLK и на ней формируется передний фронт сигнала.

Далее данные регистрируются по заднему фронту и выдаются в линию по переднему фронту сигнала SSP_CLK.

В случае передачи одного слова данных после приема его последнего бита линия SSP_FSS переводится в высокий логический уровень по истечении одного периода тактового сигнала SSP_CLK.

В режиме непрерывной передачи данных линия SSP_FSS постоянно находится в низком логическом уровне, и переводится в высокий уровень по окончании приема последнего бита блока данных, как и в режиме передачи одного слова.

33.6.18 Формат синхронного обмена SPI фирмы Motorola, SPO=1, SPH=0

Временные диаграммы последовательного синхронного обмена в режиме SPI с SPO=1, SPH=0 показаны на рисунках ниже: Рисунок 33-7 – одиночный обмен и Рисунок 33-8 – непрерывный обмен.

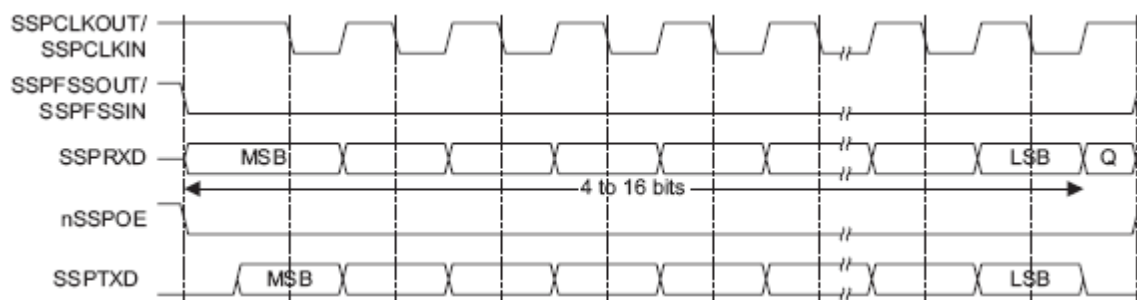


Рисунок 33-7 – Формат синхронного обмена протокола SPI, SPO=1, SPH=0 (одиночный обмен)

Примечание – На рисунке буквой Q обозначен сигнал с неопределенным уровнем.

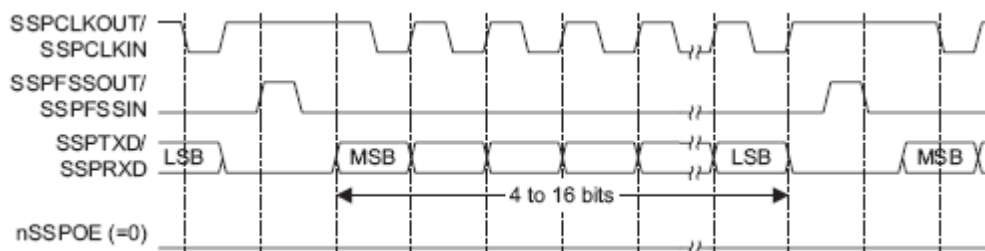


Рисунок 33-8 – Формат синхронного обмена протокола SPI, SPO=1, SPH=0 (непрерывный обмен)

В данном режиме во время ожидания приемопередатчика:

- сигнал SSP_CLK имеет высокий логический уровень;
- сигнал SSP_FSS имеет высокий логический уровень;
- сигнал SSP_TXD переводится в высокоимпедансное состояние.

Если работа модуля разрешена и в буфере FIFO передатчика содержатся корректные данные, сигнал SSP_FSS переводится в низкий логический уровень, что указывает на начало обмена данными и разрешает передачу данных от ведомого устройства на входную линию SSP_RXD ведущего. Выходной контакт передатчика SSPTXD переходит из высокоимпедансного в активное состояние.

По истечении полутакта сигнала SSP_CLK, на линии SSP_TXD формируется значение первого бита передаваемых данных. К этому моменту должны быть сформированы данные на линиях обмена, как ведущего, так и ведомого устройства. По истечении следующего полутакта сигнал SSP_CLK переводится в низкий логический уровень.

Далее данные регистрируются по заднему фронту и выдаются в линию по переднему фронту сигнала SSP_CLK.

В случае передачи одного слова данных после приема его последнего бита линия SSP_FSS переводится в высокий логический уровень по истечении одного периода тактового сигнала SSP_CLK.

В режиме непрерывной передачи данных на линии SSP_FSS должны формироваться импульсы высокого логического уровня между передачами каждого из слов данных. Это связано с тем, что в режиме SPH=0 линия выбора ведомого устройства в низком уровне блокирует запись в сдвиговый регистр. Поэтому ведущее устройство должно переводить линию SSP_FSS в высокий уровень по окончании передачи каждого кадра, разрешая таким образом запись новых данных. По окончании приема последнего бита блока данных линия SSP_FSS переводится в состояние, соответствующее режиму ожидания, по истечении одного такта сигнала SSP_CLK.

33.6.19 Формат синхронного обмена SPI фирмы Motorola, SPO=1, SPH=1

Временные диаграммы последовательного синхронного обмена в режиме SPI с SPO=1, SPH=1 показывает Рисунок 33-9 – одиночный и непрерывный обмен.

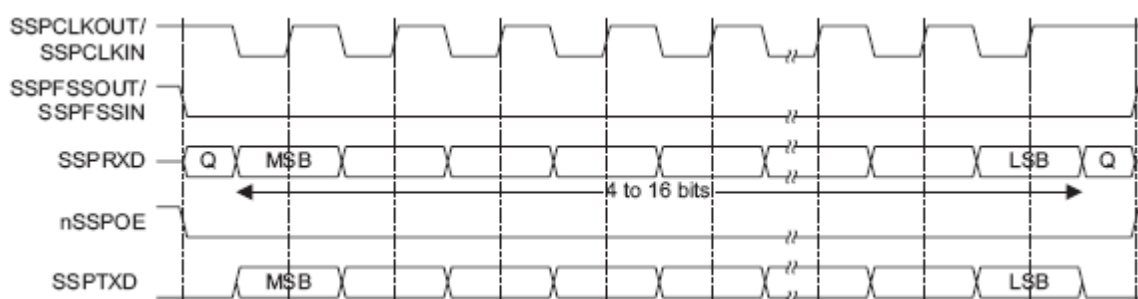


Рисунок 33-9 – Формат синхронного обмена протокола SPI, SPO=1, SPH=1

Примечание – На рисунке буквой Q обозначен сигнал с неопределенным уровнем.

В данном режиме во время ожидания приемопередатчика:

- сигнал SSP_CLK имеет высокий логический уровень;
- сигнал SSP_FSS имеет высокий логический уровень;
- сигнал SSP_TXD переводится в высокоимпедансное состояние.

Если работа модуля разрешена и в буфере FIFO передатчика содержатся корректные данные, сигнал SSP_FSS переводится в низкий логический уровень, что указывает на начало обмена данными и разрешает передачу данных от ведомого устройства на входную линию SSP_RXD ведущего. Выходной контакт передатчика SSP_TXD переходит из высокоимпедансного в активное состояние.

По истечении полутакта сигнала SSP_CLK на линиях обмена как ведущего, так и ведомого устройств сформированы значения первых бит передаваемых данных. В это же время включается линия SSP_CLK и на ней формируется передний фронт сигнала.

Далее данные регистрируются по переднему фронту и выдаются в линию по заднему фронту сигнала SSP_CLK.

В случае передачи одного слова данных после приема его последнего бита линия SSP_FSS переводится в высокий логический уровень по истечении одного периода тактового сигнала SSP_CLK.

В режиме непрерывной передачи данных линия SSP_FSS постоянно находится в низком логическом уровне и переводится в высокий уровень по окончании приема последнего бита блока данных, как и в режиме передачи одного слова.

33.6.20 Формат синхронного обмена Microwire фирмы National Semiconductor

Временные диаграммы последовательного синхронного обмена в режиме Microwire показаны на рисунках ниже: Рисунок 33-10 – одиночный обмен и Рисунок 33-11 – непрерывный обмен.

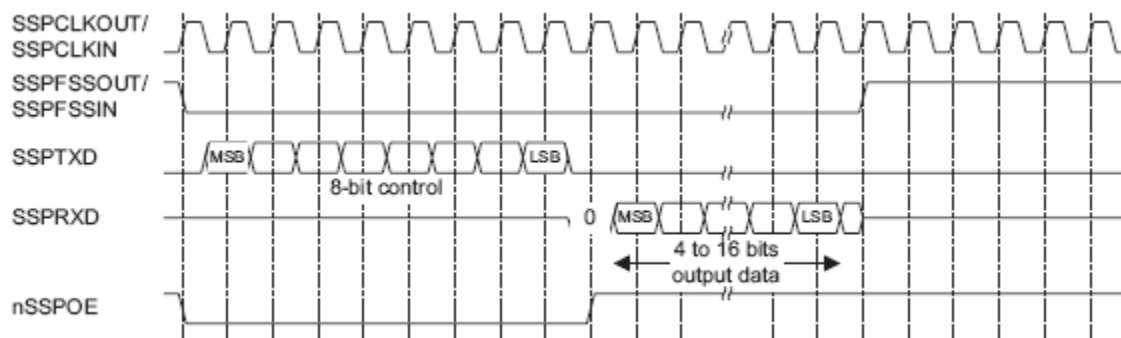


Рисунок 33-10 – Формат синхронного обмена протокола Microwire (одиночный обмен)

Протокол передачи данных Microwire во многом схож с протоколом SPI, за исключением того, что обмен в нем осуществляется в полудуплексном режиме, с использованием служебных последовательностей. Каждая информационный обмен начинается с передачи ведущим устройством специальной восьмибитной управляющей последовательности. В течение всего времени ее передачи приемник не обрабатывает каких-либо входных данных. После того, как сигнал передан и декодирован ведомым устройством, оно выдерживает паузу в один тактовый интервал после передачи последнего бита управляющей последовательности, после чего передает в адрес ведущего устройства запрошенные данные. Длительность блока данных от ведомого устройства может составлять от 4 до 16 бит, таким образом, общая длительность информационного кадра составляет от 13 до 25 бит.

В данном режиме во время ожидания приемопередатчика:

- сигнал SSP_CLK имеет низкий логический уровень;
- сигнал SSP_FSS имеет высокий логический уровень;
- сигнал SSP_TXD переводится в высокоимпедансное состояние.

Переход в режим информационного обмена происходит после записи управляющего байта в буфер FIFO передатчика. По заднему фронту сигнала SSP_FSS данные из буфера переносятся в регистр сдвига блока передатчика, откуда, начиная со старшего значащего разряда, последовательно выдаются в линию SSP_TXD. Линия SSP_FSS остается в низком логическом уровне в течение всей передачи кадра. Линия SSP_RXD при этом находится в высокоимпедансном состоянии.

Внешнее ведомое устройство осуществляет прием бит данных по переднему фронту сигнала SSP_CLK. По окончании приема последнего бита управляющей последовательности она декодируется в течение одного тактового интервала, после чего

ведомое устройство передает запрошенные данные в адрес модуля SSP. Биты данных выдаются в линию SSP_RXD по заднему фронту сигнала SSP_CLK. Ведущее устройство, в свою очередь, регистрирует их по переднему фронту этого тактового сигнала. В случае одиночного информационного обмена по окончании приема последнего бита слова данных сигнал SSP_FSS переводится в высокий уровень на время, соответствующее одному тактовому интервалу, что служит командой для переноса принятого слова данных из регистра сдвига в буфер FIFO приемника.

Примечание – Внешнее устройство может перевести линию приемника в третье состояние по заднему фронту сигнала SSP_CLK после приема последнего бита слова данных, либо после перевода линии SSP_FSS в высокий логический уровень.

Непрерывный обмен данными начинается и заканчивается также, как и одиночный обмен. Однако линия SSP_FSS удерживается в низком логическом уровне в течение всего сеанса передачи данных. Управляющий байт следующего информационного кадра передается сразу же после приема младшего значащего разряда текущего кадра. Данные из сдвигового регистра передаются в буфер приемника после регистрации младшего разряда очередного слова по заднему фронту сигнала SSP_CLK.

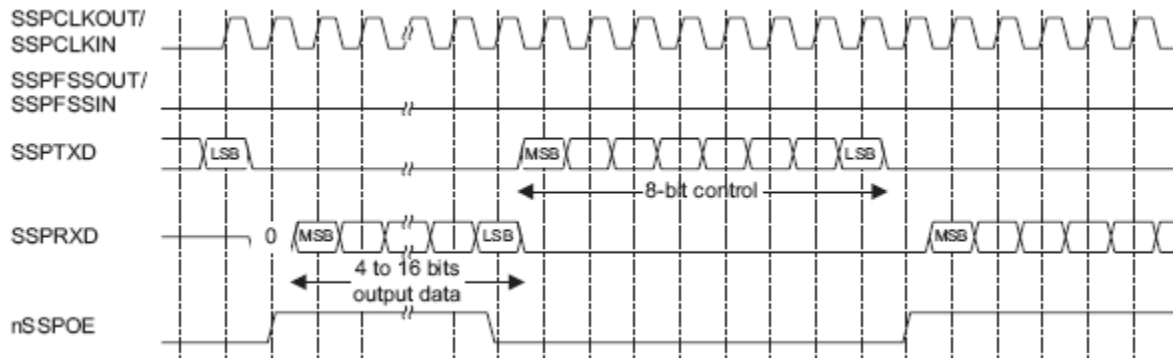


Рисунок 33-11 – Формат синхронного обмена протокола Microwire (непрерывный обмен)

33.6.20.1 Требования к временным параметрам сигнала SSP_FSS относительно тактового сигнала SSP_CLK в режиме Microwire

Модуль SSP, работающий в режиме Microwire как ведомое устройство, регистрирует данные по переднему фронту сигнала SSP_CLK после установки сигнала SSP_FSS в низкий логический уровень. Ведущие устройства, формирующие сигнал SSP_CLK, должны гарантировать достаточное время установки и удержания сигнала SSP_FSS по отношению к переднему фронту сигнала SSP_CLK.

Данные требования иллюстрирует Рисунок 33-12. По отношению к переднему фронту сигнала SSP_CLK, по которому осуществляется регистрация данных в приемнике ведомого модуля SSP, время установки сигнала SSP_FSS должно быть как минимум в два раза больше периода SSP_CLK, на котором работает модуль. По отношению к предыдущему переднему фронту сигнала SSP_CLK должно обеспечиваться время удержания не менее одного периода этого тактового сигнала.

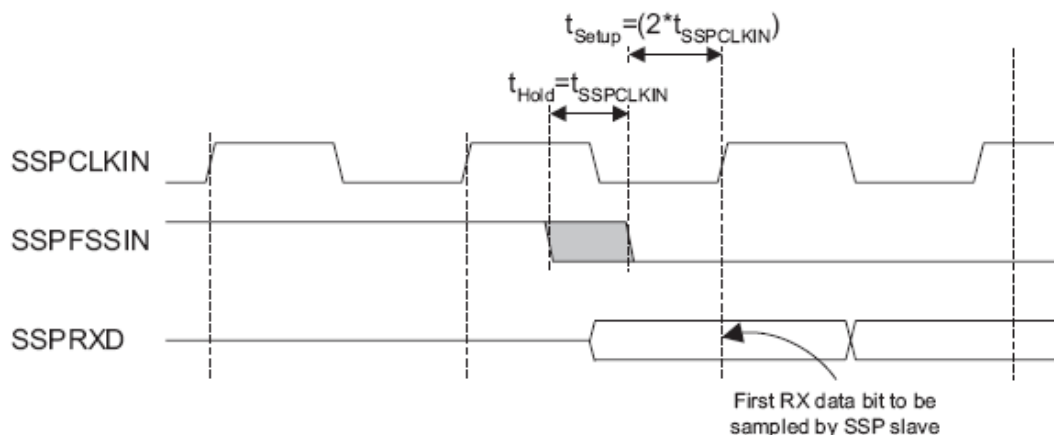


Рисунок 33-12 – Формат Microwire, требования к времени установки и удержания сигнала

33.6.21 Примеры конфигурации модуля в ведущем и ведомом режимах

На рисунках ниже показаны варианты подключения модуля SSP к периферийным устройствам, работающим в ведущем или ведомом режиме (Рисунок 33-13, Рисунок 33-14 и Рисунок 33-15).

Примечание – Модуль SSP не поддерживает динамическое изменение режима «ведущий – ведомый». Каждый приемопередатчик должен быть изначально сконфигурирован в одном из этих режимов.

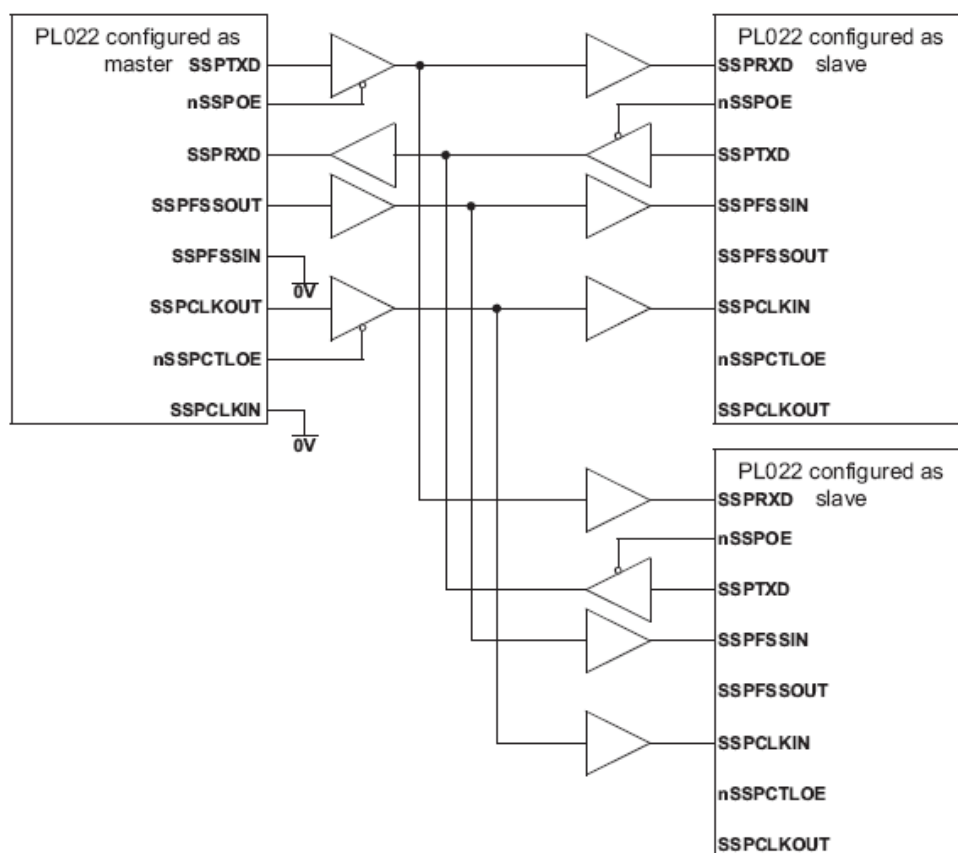


Рисунок 33-13 – Ведущее устройство SSP подключено к двум ведомым

Рисунок 33-13 показывает совместную работу трех модулей SSP, один из которых сконфигурирован в качестве ведущего, а два – в качестве ведомых устройств. Ведущее устройство способно передавать данные циркулярно в адрес двух ведомых по линии SSP_TXD.

Для ответной передачи данных один из ведомых модулей разрешает прохождение сигнала от своей линии SSP_TXD на вход SSP_RXD ведущего.

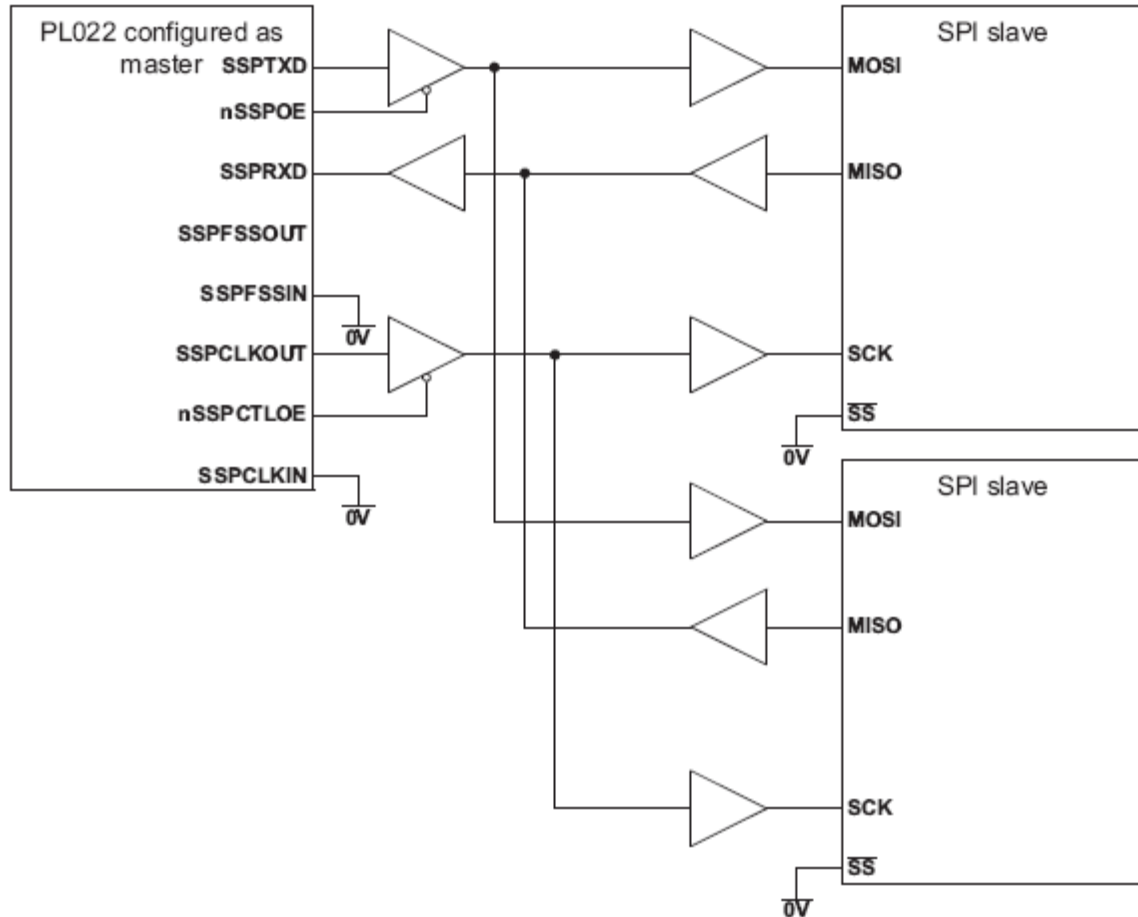


Рисунок 33-14 – Ведущее устройство SSP подключено к двум ведомым, поддерживающим SPI

Рисунок 33-14 показывает подключение модуля SSP, сконфигурированного как ведущее устройство, к двум ведомым устройствам, поддерживающим протокол SPI фирмы Motorola. Внешние устройства сконфигурированы как ведомые путем установки в низкий логический уровень сигнала выбора ведомого устройства Slave Select (SS). Как и в предыдущем примере, ведущее устройство способно передавать данные в адрес ведомых циркулярно по линии SSP_TXD. Ответная передача данных на входную линию SSP_RXD ведущего устройства одновременно осуществляется только одним из ведомых по соответствующей линии MISO.

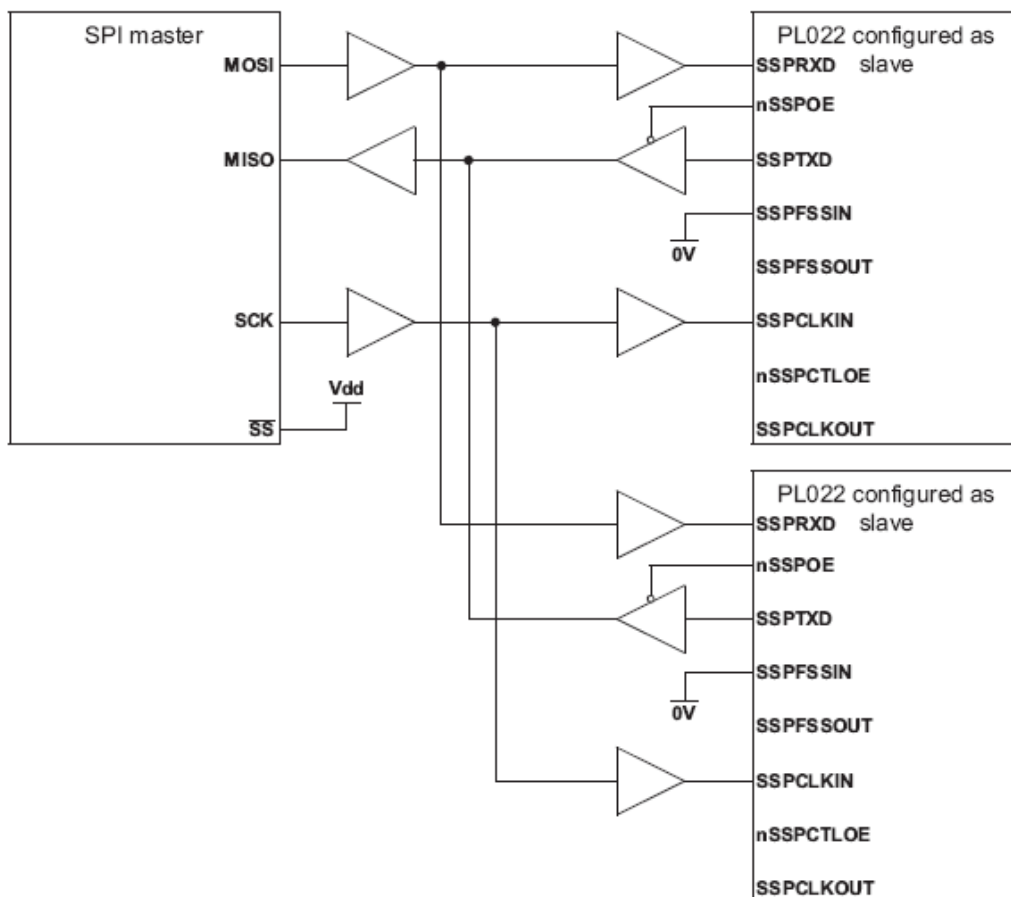


Рисунок 33-15 – Ведущее устройство, протокол SPI, подключено к двум ведомым модулям SSP

Рисунок 33-15 показывает ведущее устройство, поддерживающее протокол SPI фирмы Motorola, соединенное с двумя модулями SSP, сконфигурированными для работы в ведомом режиме. Линия Slave Select (SS) ведущего устройства в этом случае установлена в высокий логический уровень. Ведущее устройство осуществляет передачу данных по линии MOSI циркулярно в адрес двух ведомых модулей.

Для ответной передачи данных один из ведомых модулей переводит линию SSP_TXD в активное состояние, разрешая, таким образом, прохождение сигнала от своей линии SSP_TXD на вход SSP_RXD ведущего.

33.6.22 Интерфейс прямого доступа к памяти

Модуль SSP предоставляет интерфейс подключения к контроллеру прямого доступа к памяти. Работа в данном режиме контролируется регистром управления DMA SSPDMACR.

Интерфейс DMA включает в себя следующие сигналы:

– Для приема:

– SSPRXDMASREQ – запрос передачи отдельного символа, инициируется приемопередатчиком. Сигнал переводится в активное состояние в случае, если буфер FIFO приемника содержит по меньшей мере один символ;

– SSPRXDMABREQ – запрос блочного обмена данными, инициируется модулем приемопередатчика. Сигнал переходит в активное состояние в случае, если буфер FIFO приемника содержит четыре или более символов;

– SSPRXDMACLR – сброс запроса на DMA, инициируется контроллером DMA с целью сброса принятого запроса. В случае, если был запрошен блочный обмен данными, сигнал сброса формируется в ходе передачи последнего символа данных в блоке.

– Для передачи:

– SSPTXDMASREQ – запрос передачи отдельного символа, инициируется модулем приемопередатчика. Сигнал переводится в активное состояние в случае, если буфер FIFO передатчика содержит по меньшей мере одну свободную ячейку;

– SSPTXDMABREQ – запрос блочного обмена данными, инициируется модулем приемопередатчика. Сигнал переводится в активное состояние в случае, если буфер FIFO передатчика содержит четыре или менее символов;

– SSPTXDMACLR – сброс запроса на DMA, инициируется контроллером DMA с целью сброса принятого запроса. В случае, если был запрошен блочный обмен данными, сигнал сброса формируется в ходе передачи последнего символа данных в блоке.

Сигналы блочного и одноэлементного обмена данными не являются взаимоисключающими, они могут быть инициированы одновременно. Например, в случае, если заполнение данными буфера приемника превышает пороговое значение четыре, формируются как сигнал запроса одноэлементного обмена, так и сигнал запроса блочного обмена данными. В случае, если количество данных в буфере приема меньше порогового значения, формируется только запрос одноэлементного обмена. Это бывает полезно в ситуациях, при которых объем данных меньше размера блока. Пусть, например, нужно принять 19 символов. Тогда контроллер DMA осуществит четыре передачи блоков по четыре символа, а оставшиеся три символа передаст в ходе трех одноэлементных обменов.

Примечание – Для оставшихся трех символов контроллер SSP не инициирует процедуру блочного обмена.

Каждый инициированный приемопередатчиком сигнал запроса DMA остается активным до момента его сброса соответствующим сигналом DMACLR.

После снятия сигнала сброса модуль приемопередатчика вновь получает возможность сформировать запрос на DMA в случае выполнения описанных выше условий. Все запросы DMA снимаются после запрета работы приемопередатчика, а также в случае снятия сигнала разрешения DMA.

В таблице ниже (Таблица 33-1) приведены значения порогов заполнения буферов приемника и передатчика, необходимых для срабатывания запросов блочного обмена DMABREQ.

Таблица 33-1 – Параметры срабатывания запросов блочного обмена данными в режиме DMA

Пороговый уровень	Длина блока обмена данными	
	Буфер передатчика (количество незаполненных ячеек)	Буфер приемника (количество заполненных ячеек)
1/2	4	4

Рисунок 33-16 показывает временные диаграммы одноэлементного и блочного запросов DMA, в том числе действие сигнала DMACLR. Все сигналы должны быть синхронизированы с PCLK.

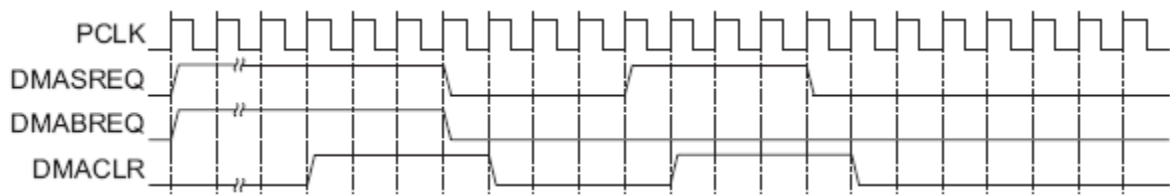


Рисунок 33-16 – Временные диаграммы обмена в режиме DMA

33.7 Программное управление модулем

33.7.1 Общая информация

В микросхеме реализовано два модуля SSP, базовые адреса каждого модуля указаны далее в таблице ниже (Таблица 33-2). Смещение каждого регистра относительно базового адреса постоянно. Следующие адреса являются резервными и не должны использоваться в нормальном режиме функционирования:

- адреса в со смещениями в диапазоне +0x028 ... +0x07C и +0xFD0 ... +0xFDC зарезервированы для перспективных расширений возможностей модуля;
- адреса в со смещениями в диапазоне +0x080 ... +0x088 зарезервированы для тестирования.

33.7.2 Описание регистров контроллера SSP

Данные о регистрах модуля SSP приведены ниже (Таблица 33-2).

Таблица 33-2 – Обобщенные данные о регистрах модуля SSP

Базовый адрес	Наименование	Тип	Значение после сброса	Размер, бит	Описание
0x4004_0000	MDR_SSP1				Контроллер SSP1
0x400A_0000	MDR_SSP2				Контроллер SSP2
0x4000_0000	MDR_SSP3				Контроллер SSP3
0x4000_8000	MDR_SSP4				Контроллер SSP4
Смещение					
0x000	CR0	RW	0x0000	16	Регистр MDR_SSPx->CR0 управления 0
0x004	CR1	RW	0x0	4	Регистр MDR_SSPx->CR1 управления 1
0x008	DR	RW	0x----	16	Буфера FIFO приемника (чтение) Буфер FIFO передатчика (запись) MDR_SSPx->DR
0x00C	SR	RO	0x03	3	Регистр MDR_SSPx->SR состояния
0x010	CPSR	RW	0x00	8	Регистр MDR_SSPx->CPSR делителя тактовой частоты
0x014	IMSC	RW	0x0	4	Регистр MDR_SSPx->IMSC маски прерывания

Базовый адрес	Наименование	Тип	Значение после сброса	Размер, бит	Описание
0x018	RIS	RO	0x8	4	Регистр MDR_SSPx->RIS состояния прерываний без учета маскирования
0x01C	MIS	RO	0x0	4	Регистр MDR_SSPx->MIS состояния прерываний с учетом маскирования
0x020	ICR	WO	0x0	4	Регистр MDR_SSPx->ICR сброса прерывания
0x024	DMACR	RW	0x0	2	Регистр MDR_SSPx->DMACR управления прямым доступом к памяти
Примечание – В поле «тип» указан вид доступа к регистру: RW – чтение и запись, RO – только чтение, WO – только запись					

33.7.2.1 MDR_SSPx->CR0

Регистр управления 0

Регистр CR0 содержит пять битовых полей, предназначенных для управления блоками модуля SSP.

Таблица 33-3 отображает назначение бит регистра CR0.

Таблица 33-3 – Описание бит регистра CR0

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...16	-	Зарезервировано
15...8	SCR	Скорость последовательного обмена. Значение поля SCR используется при формировании тактового сигнала обмена данными. Информационная скорость удовлетворяет соотношению: $F\_SSPCLK / (CPSDVR \cdot (1 + SCR))$, где CPSDVR – четное число в диапазоне от 2 до 254 (см. регистр SSPCPSR), а SCR – число от 0 до 255
7	SPH	Фаза сигнала SSPCLKOUT (используется только в режиме обмена SPI фирмы Motorola). См. пункт 33.6.15 «Формат синхронного обмена SPI фирмы Motorola»
6	SPO	Полярность сигнала SSPCLKOUT (используется только в режиме обмена SPI фирмы Motorola). См. пункт 33.6.15 «Формат синхронного обмена SPI фирмы Motorola»
5, 4	FRF	Формат информационного кадра. 00 – протокол SPI фирмы Motorola; 01 – протокол SSI фирмы Texas Instruments; 10 – протокол Microwire фирмы National Semiconductor; 11 – резерв
3...0	DSS	Размер слова данных: 0000 – резерв; 0001 – резерв; 0010 – резерв; 0011 – 4 бита;

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
		0100 – 5 бит; 0101 – 6 бит; 0110 – 7 бит; 0111 – 8 бит; 1000 – 9 бит; 1001 – 10 бит; 1010 – 11 бит; 1011 – 12 бит; 1100 – 13 бит; 1101 – 14 бит; 1110 – 15 бит; 1111 – 16 бит

33.7.2.2 MDR_SSPx->CR1

Регистр управления 1

Регистр CR1 содержит четыре битовых поля, предназначенных для управления блоками модуля SSP.

Таблица 33-4 отображает назначение бит регистра CR1.

Таблица 33-4 – Описание бит регистра CR1

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15...4		Резерв, при чтении результат не определен. При записи следует устанавливать в 0
3	SOD	Запрет выходных линий в режиме ведомого устройства. Бит используется только в режиме ведомого устройства (MS=1). Это позволяет организовать двусторонний обмен данными в системах, содержащих одно ведущее и несколько ведомых устройств. Бит SOD следует установить в случае, если данный ведомый модуль SSP не должен в настоящее время осуществлять передачу данных в линию SSP_TXD. При этом линии обмена данными ведомых устройств можно соединить параллельно. 0 – управление линией SSP_TXD в ведомом режиме разрешено; 1 – управление линией SSP_TXD в ведомом режиме запрещено
2	MS	Выбор ведущего или ведомого режима работы: 0 – ведущий модуль (устанавливается по умолчанию); 1 – ведомый модуль
1	SSE	Разрешение работы приемопередатчика: 0 – работа запрещена; 1 – работа разрешена
0	LBM	Тестирование по шлейфу: 0 – нормальный режим работы приемопередатчика; 1 – выход регистра сдвига передатчика соединен с входом регистра сдвига приемника

33.7.2.3 MDR_SSPx->DR

Регистр данных

Регистр SSPDR имеет разрядность 16 бит и предназначен для чтения принятых и записи передаваемых данных.

Операция чтения обеспечивает доступ к последней несчитанной ячейке буфера FIFO приемника. Запись данных в этот буфер FIFO осуществляет блок приемника.

Операция записи позволяет занести очередное слово в буфер FIFO передатчика. Извлечение данных из этого буфера осуществляет блок передатчика. При этом извлеченные данные помещаются в регистр сдвига передатчика, откуда последовательно выдаются на линию SSP_TXD с заданной скоростью информационного обмена.

В случае, если выбран размер информационного слова менее 16 бит, перед записью в регистр SSPDR необходимо обеспечить выравнивание данных по правой границе. Блок передатчика игнорирует неиспользуемые биты. Принятые информационные слова автоматически выравниваются по правой границе в блоке приемника.

В режиме обмена данными Microwire фирмы National Semiconductor модуль SSP по умолчанию работает с восьмиразрядными информационными словами (старший значащий байт игнорируется). Размер принимаемых данных задается программно. Буфера FIFO приемника и передатчика автоматически не очищаются даже в случае, если бит SSE установлен в 0. Это позволяет заполнить буфер передатчика необходимой информацией заблаговременно, перед разрешением работы модуля.

Таблица 33-5 отображает назначение бит регистра SSPDR.

Таблица 33-5 – Описание бит регистра DR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15...0	DATA	Принимаемые данные (чтение). Передаваемые данные (запись). В случае, если выбран размер информационного слова менее 16 бит, перед записью в регистр SSPDR необходимо обеспечить выравнивание данных по правой границе. Блок передатчика игнорирует неиспользуемые биты. Принятые информационные слова автоматически выравниваются по правой границе в блоке приемника

33.7.2.4 MDR_SSPx->SR

Регистр состояния

Регистр состояния доступен только для чтения и содержит информацию о состоянии буферов FIFO приемника и передатчика и занятости модуля SSP.

Таблица 33-6 отображает назначение бит регистра SSPSR.

Таблица 33-6 – Описание бит регистра SR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15...5		Резерв, при чтении результат не определен
4	BSY	Флаг активности модуля: 0 – модуль SSP не активен; 1 – модуль SSP в настоящее время передает и/или принимает данные, либо буфер FIFO передатчика не пуст
3	RFF	Буфер FIFO приемника заполнен: 0 – не заполнен; 1 – заполнен
2	RNE	Буфер FIFO приемника не пуст: 0 – пуст; 1 – не пуст
1	TNF	Буфер FIFO передатчика не заполнен: 0 – заполнен; 1 – не заполнен
0	TFE	Буфер FIFO передатчика пуст: 0 – не пуст; 1 – пуст

33.7.2.5 MDR_SSPx->CPSR

Регистр делителя тактовой частоты

Регистр SSPCPSR используется для установки параметров делителя тактовой частоты. Записываемое значение должно быть целым числом в диапазоне от 2 до 254. Младший значащий разряд регистра принудительно устанавливается в ноль. Если записать в регистр SSPCPSR нечетное число, его последующее чтение даст результатом это число, но с установленным в ноль младшим битом.

Таблица 33-7 отображает назначение бит регистра SSPCPSR.

Таблица 33-7 – Описание бит регистра CPSR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...8	-	Резерв. При чтении результат не определен. При записи следует заполнить нулями
7... 0	CPSDVSR	Коэффициент деления тактовой частоты. Записываемое значение должно быть целым числом в диапазоне от 2 до 254. Младший значащий разряд регистра принудительно устанавливается в ноль

33.7.2.6 MDR_SSPx->IMSC

Регистр установки и сброса маски прерывания

При чтении выдается текущее значение маски. При записи производится установка или сброс маски на соответствующее прерывание. При этом запись 1 в разряд разрешает соответствующее прерывание, запись 0 – запрещает.

После сброса все биты регистра маски устанавливаются в нулевое состояние.

Таблица 33-8 отображает назначение бит в регистре IMSC.

Таблица 33-8 – Описание бит регистра IMSC

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...4		Резерв. Не модифицируйте. При чтении выдаются нули
3	TXIM	Маска прерывания по заполнению на 50% и менее буфера FIFO передатчика: 1 – не маскирована; 0 – маскирована
2	RXIM	Маска прерывания по заполнению на 50% и менее буфера FIFO приемника: 1 – не маскирована; 0 – маскирована
1	RTIM	Маска прерывания по таймауту приемника (буфер FIFO приемника не пуст и не было попыток его чтения в течение времени таймаута): 1 – не маскирована; 0 – маскирована
0	RORIM	Маска прерывания по переполнению буфера приемника: 1 – не маскирована; 0 – маскирована

33.7.2.7 MDR_SSPx->RIS

Регистр состояния прерываний

Этот регистр доступен только для чтения и содержит текущее состояние прерываний без учета маскирования. Данные, записываемые в регистр, игнорируются.

Таблица 33-9 отображает назначение бит в регистре RIS.

Таблица 33-9 – Описание бит регистра RIS

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31... 4		Резерв. Не модифицируйте. При чтении выдаются нули
3	TXRIS	Состояние до маскирования прерывания SSPTXINTR
2	RXRIS	Состояние до маскирования прерывания SSPRXINTR
1	RTRIS	Состояние до маскирования прерывания SSPRTINTR
0	RORRIS	Состояние до маскирования прерывания SSPRORINTR

33.7.2.8 MDR_SSPx->MIS

Регистр маскированного состояния прерываний

Этот регистр доступен только для чтения и содержит текущее состояние прерываний с учетом маскирования. Данные, записываемые в регистр, игнорируются.

Таблица 33-10 отображает назначение бит в регистре SSPMIS.

Таблица 33-10 – Описание бит регистра MIS

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...4		Резерв. Не модифицируйте. При чтении выдаются нули
3	TXMIS	Состояние маскированного прерывания SSPTXINTR
2	RXMIS	Состояние маскированного прерывания SSPRXINTR
1	RTMIS	Состояние маскированного прерывания SSPRTINTR
0	RORMIS	Состояние маскированного прерывания SSPRORINTR

33.7.2.9 MDR_SSPx->ICR

Регистр сброса прерываний

Этот регистр доступен только для записи и предназначен для сброса признака прерывания по заданному событию путем записи 1 в соответствующий бит. Запись в любой из разрядов регистра 0 игнорируется.

Таблица 33-11 отображает назначение бит в регистре SSPICR.

Таблица 33-11 – Описание бит регистра ICR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31... 2		Резерв. Не модифицируйте. При чтении выдаются нули
1	RTIC	Сброс прерывания SSPRTINTR
0	RORIC	Сброс прерывания SSPRORINTR

33.7.2.10 MDR_SSPx->DMACR

Регистр управления прямым доступом к памяти

Регистр доступен по чтению и записи. После сброса все биты регистра обнуляются.

Таблица 33-12 отображает назначение бит в регистре DMACR.

Таблица 33-12 – Описание бит регистра DMACR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...2		Резерв. Не модифицируйте. При чтении выдаются нули.
1	TXDMAE	Использование DMA при передаче. Если бит установлен в 1, разрешено формирование запросов DMA для обслуживания буфера FIFO передатчика
0	RXDMAE	Использование DMA при приеме. Если бит установлен в 1, разрешено формирование запросов DMA для обслуживания буфера FIFO приемника

33.8 Прерывания

В модуле предусмотрено пять маскируемых линий запроса на прерывание с выводом на один общий сигнал, представляющий собой комбинацию независимых по схеме ИЛИ.

Сигналы запроса на прерывание:

- SSPRXINTR – запрос на обслуживание буфера FIFO приемника;
- SSPTXINTR – запрос на обслуживание буфера FIFO передатчика;
- SSPRORINTR – переполнение буфера FIFO приемника;
- SSPRTINTR – таймаут приемника;
- SSPINTR – логическое ИЛИ сигналов SSPRXINTR, SSPTXINTR, SSPRTINTR и SSPRORINTR.

Каждый из независимых сигналов запроса на прерывание может быть маскирован путем установки соответствующего бита в регистре маски SSPIMSC. Установка бита в «1» разрешает соответствующее прерывание, а в 0 – запрещает.

Доступность индивидуальных линий и общей линии запроса позволяет организовать обслуживание прерываний в системе как путем применения глобальной процедуры обработки, так и с помощью драйвера устройства, построенного по модульному принципу.

Прерывания от приемника и передатчика SSPRXINTR и SSPTXINTR выведены отдельно от прерываний по изменению состояния устройства. Это позволяет использовать данные сигналы запроса для обеспечения чтения и записи данных согласованно с достижением заданного порога заполнения буферов FIFO приемника и передатчика.

Признаки возникновения каждого из условий прерывания можно считать либо из регистра прерываний SSPRIS, либо из маскированного регистра прерываний SSPMIS.

33.8.1 SSPRXINTR

Прерывание по заполнению буфера FIFO приемника формируется в случае, если буфер приемника содержит четыре или более несчитанных слов данных.

33.8.2 SSPTXINTR

Прерывание по заполнению буфера FIFO передатчика формируется в случае, если буфер передатчика содержит четыре или менее корректных слов данных.

Состояние прерывания не зависит от значения сигнала разрешения работы модуля SSP. Это позволяет организовать взаимодействие программного обеспечения с передатчиком одним из двух способов. Во-первых, можно записать данные в буфер заблаговременно, перед активизацией передатчика и разрешения прерываний. Во-вторых, можно предварительно разрешить работу модуля и формирование прерываний и заполнять буфер передатчика в ходе работы процедуры обслуживания прерываний.

33.8.3 SSPRORINTR

Прерывание по переполнению буфера FIFO приемника формируется в случае, если буфер уже заполнен и блоком приемника осуществлена попытка записать в него еще одно слово. При этом принятое слово данных регистрируется в регистре сдвига приемника, но в буфер приемника не заносится.

33.8.4 SSPRTINTR

Прерывание по таймауту приемника возникает в случае, если буфер FIFO приемника не пуст, и на вход приемника не поступало новых данных в течение времени таймаута, равного 32 тактам частоты SSPCLKOUT (для ведущего и ведомого режимов работы). Данный механизм гарантирует, что пользователь будет знать о наличии в буфере приемника необработанных данных.

Прерывание по таймауту снимается либо после считывания данных из буфера приемника до его опустошения, либо после приема новых слов данных по входной линии SSP_RXD. Кроме того, оно может быть снято путем записи 1 в бит RTIC регистра сброса прерывания SSPTICR.

33.8.5 SSPINTR

Все описанные сигналы запроса на прерывание скомбинированы в общую линию путем объединения по схеме ИЛИ сигналов SSPRXINTR, SSPTXINTR, SSPRTINTR и SSPRORINTR с учетом маскирования. Общий выход может быть подключен к системному контроллеру прерывания, что позволит ввести дополнительное маскирование запросов на уровне периферийных устройств.

34 Контроллер MDR_UART

Модуль универсального асинхронного приемопередатчика (UART – Universal Synchronous Asynchronous Receiver Transmitter) представляет собой периферийное устройство микросхемы.

В состав контроллера включен кодек (ENDEC – ENcoder/DEcoder) последовательного интерфейса инфракрасной (ИК) передачи данных в соответствии с протоколом SIR (SIR – Serial Infra Red) ассоциации Infrared Data Association (IrDA).

34.1 Основные сведения

Основные сведения о модуле представлены в следующих разделах:

- основные характеристики;
- программируемые параметры;
- отличия от приемопередатчика 16C650.

34.1.1 Основные характеристики модуля UART

Может быть запрограммирован для использования, как в качестве универсального асинхронного приемопередатчика, так и для инфракрасного обмена данными (SIR).

Содержит независимые буферы приема (16x12) и передачи (16x8) типа FIFO (First In First Out – первый вошел, первый вышел), что позволяет снизить интенсивность прерываний центрального процессора.

Программное отключение FIFO позволяет ограничить размер буфера одним байтом.

Программное управление скоростью обмена. Обеспечивается возможность деления тактовой частоты опорного генератора в диапазоне (1x16 – 65535x16). Допускается использование нецелых коэффициентов деления частоты, что позволяет использовать любой опорный генератор с частотой более 3,6864 МГц.

Поддержка стандартных элементов асинхронного протокола связи – стартового и стопового бит, а та же бита контроля четности, которые добавляются перед передачей и удаляются после приема.

Независимое маскирование прерываний от буфера FIFO передатчика, буфера FIFO приемника, по таймауту приемника, по изменению линий состояния модема, а также в случае обнаружения ошибки.

Поддержка прямого доступа к памяти.

Обнаружение ложных стартовых бит.

Формирование и обнаружения сигнала разрыва линии.

Полностью программируемый асинхронный последовательный интерфейс с характеристиками:

- данные длиной 5, 6, 7 или 8 бит;
- формирование и контроль четности (проверочный бит выставляется по четности, нечетности, имеет фиксированное значение, либо не передается);
- формирование 1 или 2 стоповых бит;
- скорость передачи данных – от 0 до UARTCLK/16 Бод.

Кодек ИУ обмена данными IrDA SIR обеспечивает:

- программный выбор обмена данными по линиям асинхронного приемопередатчика либо кодека ИК-связи IrDA SIR;
- поддержку функционирования с информационной скоростью до 115200 бит/с в режиме полудуплекса;
- поддержку длительности бит для нормального режима (3/16) и для режима пониженного энергопотребления (1.41 – 2.23 мкс);
- программируемое деление опорной частоты UARTCLK для получения заданной длительности бит в режиме пониженного энергопотребления.

Наличие идентификационного регистра, однозначно идентифицирующего модуль, что позволяет операционной системе выполнять автоматическую конфигурацию.

34.1.2 Программируемые параметры

Следующие ключевые параметры могут быть заданы программно:

- скорость передачи данных – целая и дробная часть числа;
- количество бит данных;
- количество стоповых бит;
- режим контроля четности;
- разрешение или запрет использования буферов FIFO (глубина очереди данных – 32 элемента или один элемент, соответственно);
- порог срабатывания прерывания по заполнению буферов FIFO (1/8, 1/4, 1/2, 3/4 и 7/8);
- частота внутреннего тактового генератора (номинальное значение – 1,8432 МГц) может быть задана в диапазоне 1,42 – 2,12 МГц для обеспечения возможности формирования бит данных с укороченной длительностью в режиме пониженного энергопотребления;
- режим аппаратного управления потоком данных.

34.1.3 Отличия от контроллера UART 16C650

Контроллер отличается от промышленного стандарта асинхронного приемопередатчика 16C650 следующими характеристиками:

- пороги срабатывания прерывания по заполнению буфера FIFO приемника – 1/8, 1/4, 1/2, 3/4 и 7/8;
- пороги срабатывания прерывания по заполнению буфера FIFO передатчика – 1/8, 1/4, 1/2, 3/4 и 7/8;
- отличается распределение адресов внутренних регистров и назначение бит в регистрах;
- недоступны изменения сигналов состояния модема.

Следующие возможности контроллера 16C650 не поддерживаются:

- полуторная длительность стопового бита (поддерживается только 1 или 2 стоповых бита);
- независимое задание тактовой частоты приемника и передатчика.

34.2 Функциональные возможности

Устройство выполняет следующие функции:

- преобразование данных, полученных от периферийного устройства, из последовательной в параллельную форму;
- преобразование данных, передаваемых на периферийное устройство, из параллельной и последовательную форму.

Процессор читает и записывает данные, а также управляющую информацию и информацию о состоянии модуля. Прием и передача данных буферизуются с помощью внутренней памяти FIFO, позволяющей сохранить до 16 байтов независимо для режимов приема и передачи.

Модуль приемопередатчика:

- содержит программируемый генератор, формирующий тактовый сигнал одновременно для передачи и для приема данных на основе внутреннего тактового сигнала UARTCLK;
- обеспечивает возможности, сходные с возможностями промышленного стандарта – контроллера UART 16C650;
- позволяет осуществлять обмен информацией с максимальной скоростью:
 - в режиме UART – в зависимости от частоты (см пункт 34.5.2 «Тактовые сигналы»);
 - в режиме IrDA – до 460800 бит/с;
 - в режиме IrDA с пониженным энергопотреблением – до 115200 бит/с.

Режим работы приемопередатчика и скорость обмена данными контролируются регистром управления линией UARTLCR_N и регистрами делителя скорости передачи данных – целой части (UARTIBRD) и дробной части (UARTFBRD).

Устройство может формировать следующие сигналы:

- независимые маскируемые прерывания от приемника (в том числе по таймауту), передатчика, а также по изменению состояния модема и в случае обнаружения ошибки;
- общее прерывание, возникающее в случае, если возникло одно из независимых немаскированных прерываний;
- сигналы запроса на прямой доступ к памяти (DMA) для совместной работы с контроллером DMA.

В случае возникновения ошибки в структуре сигнала, четности данных, а также разрыва линии соответствующий бит ошибки устанавливается и сохраняется в буфере FIFO. В случае переполнения буфера немедленно устанавливается соответствующий бит в регистре переполнения, а доступ к записи в буфер FIFO блокируется.

Существует возможность программно ограничить размер буфера FIFO одним байтом, что позволяет реализовать общепринятый интерфейс асинхронной последовательной связи с двойной буферизацией.

Блок последовательного интерфейса инфракрасной передачи данных в соответствии с протоколом IrDA SIR реализует протокол обмена данными ENDEC. В случае его активизации обмен информацией осуществляется не с помощью сигналов UARTTXD и UARTRXD, а посредством сигналов nSIROUT и SIRIN.

В этом случае устройство переводит линию UARTTXD в пассивное состояние (высокий уровень), и перестает реагировать на изменение состояния модема, а также сигнала на линии UARTRXD. Протокол SIR ENDEC обеспечивает возможность обмена данными исключительно в режиме полудуплекса, то есть он не может передавать во время приема данных и принимать во время передачи данных.

В соответствии со спецификацией физического уровня протокола IrDA SIR, задержка между передачей и приемом должна составлять не менее 10 мс.

34.3 Описание функционирования блока UART

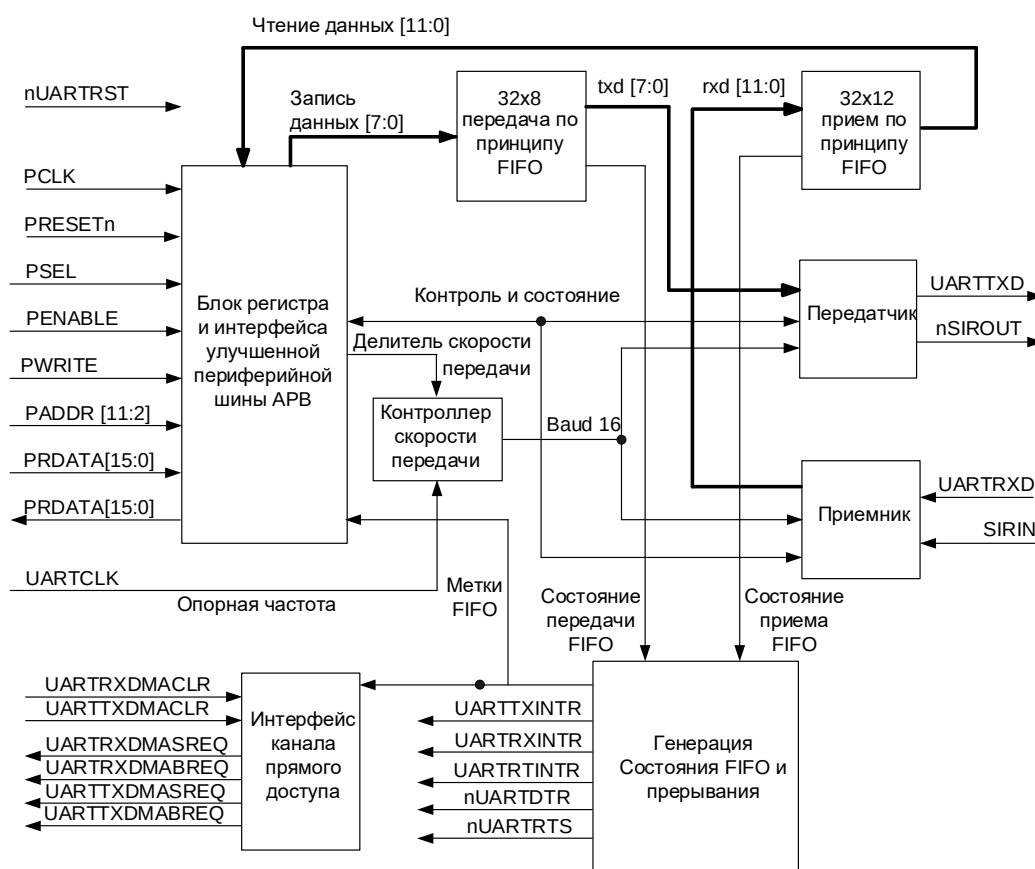


Рисунок 34-1 – Блок-схема универсального асинхронного приемопередатчика (UART)

34.3.1 Генератор тактового сигнала приемопередатчика

Генератор содержит счетчики без цепи сброса, формирующие внутренние тактовые сигналы Baud16 и IrLPBaud16.

Сигнал Baud16 используется для синхронизации схем управления приемником и передатчиком последовательного обмена данными. Он представляет собой последовательность импульсов с шириной, равной одному периоду сигнала UARTCLK и частотой, в 16 раз выше скорости передачи данных.

Сигнал IrLPBaud16 предназначен для синхронизации схемы формирования импульсов с длительностью, требуемой для ИК обмена данными в режиме с пониженным энергопотреблением.

34.3.2 Буфер FIFO передатчика

Буфер передатчика имеет ширину 8 бит, глубину 16 слов, схему организации доступа типа «первый вошел, первый вышел». Данные от центрального процессора, записанные через шину APB, сохраняются в буфере до тех пор, пока не будут считаны логической схемой передачи данных. Существует возможность запретить буфер FIFO передатчика, в этом случае он будет функционировать как однобайтовый буферный регистр.

34.3.3 Буфер FIFO приемника

Буфер приемника имеет ширину 12 бит, глубину 16 слов, схему организации доступа типа «первый вошел, первый вышел». Принятые от периферийного устройства данные и соответствующие кодов ошибки сохраняются логикой приема данных в нем до тех пор, пока не будут считаны центральным процессором через шину APB. Буфер FIFO приемника может быть запрещен, в этом случае он будет действовать как однобайтовый буферный регистр.

34.3.4 Блок передатчика

Логические схемы передатчика осуществляют преобразование данных, считанных из буфера передатчика, из параллельной в последовательную форму. Управляющая логика выдает последовательный поток бит в порядке: стартовый бит, биты данных, начиная с младшего значащего разряда, бит проверки на четность, и, наконец, стоповые биты, в соответствии с конфигурацией, записанной в регистре управления.

34.3.5 Блок приемника

Логические схемы приемника преобразуют данные, полученные от периферийного устройства, из последовательной в параллельную форму после обнаружения корректного стартового импульса. Кроме того, производятся проверки переполнения буфера, проверки на ошибки контроля четности, на ошибки в структуре сигнала, а также на разрыв линии. Признаки обнаружения этих ошибок также сохраняются в выходном буфере.

34.3.6 Блок формирования прерываний

Контроллер генерирует независимые маскируемые прерывания с активным высоким уровнем. Кроме того, формируется комбинированное прерывание путем объединения указанных независимых прерываний по схеме ИЛИ.

Комбинированный сигнал прерывания может быть подан на внешний контроллер прерываний системы, при этом появится дополнительная возможность маскирования устройства в целом, что облегчает построение модульных драйверов устройств.

Другой подход состоит в подаче на системный контроллер прерываний независимых линий запроса на прерывание от приемопередатчика. В этом случае процедура обработки сможет одновременно считать информацию обо всех источниках прерывания. Данный подход привлекателен в случае, если скорость доступа к регистрам периферийных устройств значительно превышает тактовую частоту центрального процессора в системе реального времени.

Для более подробной информации см. подраздел 34.7 «Прерывания».

34.3.7 Интерфейс прямого доступа к памяти

Модуль обеспечивает интерфейс с контроллером DMA согласно схеме взаимодействия приемопередатчика и контроллера DMA.

34.3.8 Блок и регистры синхронизации

Контроллер поддерживает как асинхронный, так и синхронный режимы работы тактовых генераторов CPU_CLK и UARTCLK. Регистры синхронизации и логика квитирования постоянно находятся в активном состоянии. Это практически не отражается на характеристиках устройства и занимаемой площади. Синхронизация сигналов управления осуществляется в обоих направлениях потока данных, то есть как из области действия CPU_CLK в область действия UARTCLK, так и наоборот.

34.4 Описание функционирования ИК-кодека IrDA SIR

Рисунок 34-2 демонстрирует структурную схему кодека.

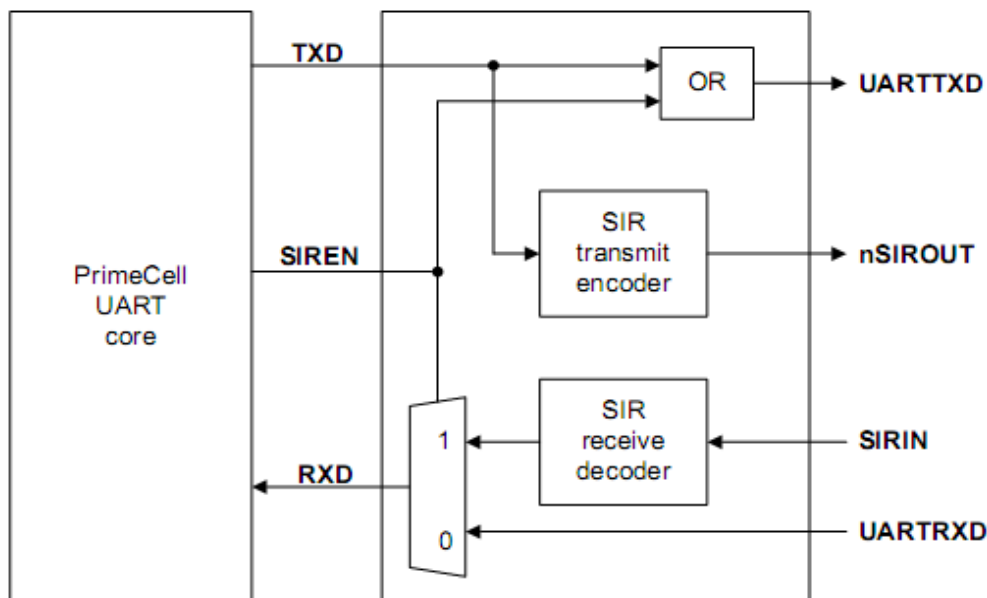


Рисунок 34-2 – Структурная схема кодека IrDA

34.4.1 Кодер ИК-передатчика

Кодер преобразует поток данных с выхода асинхронного передатчика, сформированный по закону модуляции без возврата к нулю (NRZ). Спецификация физического уровня протокола IrDA SIR подразумевает использование модуляции с возвратом к нулю и инверсией (RZI), в соответствии с которой передача логического нуля соответствует излучению одного светового ИК-импульса. Сформированный выходной поток импульсов подается на усилитель и, далее, на ИК-светодиод.

Длительность импульса в режиме IrDA составляет, согласно спецификации, три периода внутреннего тактового генератора с частотой Baud16, то есть 3/16 периода времени, выделенного на передачу одного бита.

В режиме IrDA с пониженным энергопотреблением ширина импульса задана как 3/16 периода, выделенного на передачу бита, при скорости передачи данных 115200 бит/с. Данное требование реализуется за счет формирования трех периодов тактового сигнала IrLPBaud16 с номинальной частотой 1,8432 МГц, в свою очередь, формируемого путем

деления частоты UARTCLK. Значение частоты IrLPBaud16 задается путем записи соответствующего коэффициента деления частоты в регистр UARTILPR.

Выход кодера имеет активное низкое состояние. При передаче логической единицы выход кодера остается в низком состоянии, при передаче логического нуля – формируется импульс, при этом выход кратковременно переводится в высокое состояние.

Как в нормальном режиме, так и в режиме пониженного энергопотребления использование нецелых значений коэффициента деления скорости передачи данных увеличивает джиттер («дребезжание») фронтов импульсов данных. Наличие джиттера в случае использования дробных коэффициентов деления связано с тем, что интервалы между тактовыми импульсами Baud16 будут нерегулярными – период сигнала Baud16 в разное время будет содержать различное количество периодов сигнала UARTCLK. Можно показать, что в наихудшем случае величина джиттера в потоке ИК-импульсов может достигать трех периодов UARTCLK. В соответствии со спецификацией стандарта IrDA SIR, джиттер не должен превышать величины 13 %. В случае, если частота сигнала UARTCLK составляет более 3 6834 МГц, а скорость передачи данных меньше или равна 115200 бит/с, величина джиттера не превышает 9%. Таким образом, требования стандарта выполняются.

34.4.2 Декодер ИК-приемника

Декодер преобразует поток данных, сформированных по закону возврата к нулю, полученного от приемника ИК сигнала, и выдает поток данных без возврата к нулю на вход приемника UART. В неактивном состоянии вход декодера находится нормально в высоком состоянии. Выходной сигнал кодера имеет полярность, противоположную полярности входа декодера.

Обнаружение стартового бита осуществляется при низком уровне сигнала на входе декодера.

Примечание – Для того, чтобы исключить ложные срабатывания UART от импульсных помех, на входе SIRIN игнорируются импульсы с длительностью менее, чем:

- 3/16 длительности Baud16 в режиме IrDA;
- 3/16 длительности IrLPBaud16 в режиме IrDA с пониженным энергопотреблением.

34.5 Описание работы UART

34.5.1 Сброс модуля

Приемопередатчик и кодек могут быть сброшены общим сигналом сброса процессора. Значения регистров после сброса описаны в подразделе 34.8 «Программное управление модулем».

34.5.2 Тактовые сигналы

Частота тактового сигнала UARTCLK должна обеспечивать поддержку требуемого диапазона скоростей передачи данных:

$$F_{\text{UARTCLK}}(\text{min}) \geq 16 \cdot \text{baud\_rate\_max};$$

$$F_{\text{UARTCLK}}(\text{max}) \leq 16 \cdot 65535 \cdot \text{baud\_rate\_min}.$$

Например, для поддержки скорости передачи данных в диапазоне от 110 до 460800 Бод частота UARTCLK должна находиться в интервале от 7 3728 МГц до 115,34 МГц.

Частота UARTCLK, кроме того, должна выбираться с учетом возможности установки скорости передачи данных в рамках заданных требований точности.

34.5.3 Работа универсального асинхронного приемопередатчика

Управляющая информация хранится в регистре управления линией UARTLCR. Этот регистр имеет внутреннюю ширину 30 бит, однако внешний доступ по шине APB к нему осуществляется через следующие регистры:

- UARTLCR_H – определяет:
 - параметры передачи данных;
 - длину слова;
 - режим буферизации;
 - количество передаваемых стоповых бит;
 - режим контроля четности;
 - формирование сигнала разрыва линии;
- UARTIBRD – определяет целую часть коэффициента деления для скорости передачи данных;
- UARTFBRD – определяет дробную часть коэффициента деления для скорости передачи данных.

34.5.4 Коэффициент деления частоты

Коэффициент деления для формирования скорости передачи данных состоит из 22 бит, при этом 16 бит выделено для представления его целой части, а 6 бит – дробной части. Возможность задания нецелых коэффициентов деления позволяет осуществлять обмен данными со стандартными информационными скоростями, при этом используя в качестве UARTCLK тактовый сигнал с произвольной частотой более 3,6864 МГц.

Целая часть коэффициента деления записывается в 16-битный регистр UARTIBRD. Шестиразрядная дробная часть записывается в регистр UARTFBRD. Значение коэффициента деления связано с содержимым указанных регистров следующим образом:

$$\text{Коэффициент деления} = \frac{UARTCLK}{16 \cdot \text{скорость передачи данных}} = BRD\_I + BRD\_F, \quad (7)$$

где BRD_I – целая часть;

BRD_F – дробная часть коэффициента деления.

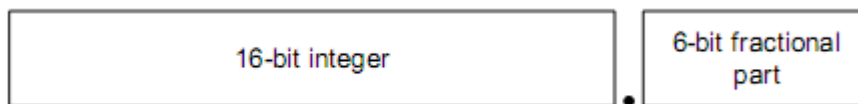


Рисунок 34-3 – Коэффициент деления

Шестибитное значение, записываемое в регистр UARTFBRD, вычисляется путем выделения дробной части требуемого коэффициента деления, умножения ее на 64 (то есть на $2n$, где n – ширина регистра UARTFBRD) и округления до ближайшего целого числа:

$$M = \text{integer}(BRD\_F \cdot 2n + 0,5), \quad (8)$$

где *integer* – операция отсечения дробной части числа, $n = 6$.

В модуле формируется внутренний сигнал Baud16, представляющий собой последовательность импульсов с длительностью, равной периоду сигнала UARTCLK и средней частотой, в 16 раз большей требуемой скорости обмена данными.

34.5.5 Передача и прием данных

Принятые или передаваемые данные заносятся в 16-элементные буферы FIFO, при этом каждый элемент приемного буфера FIFO кроме байта данных хранит также четыре бита информации о состоянии модема.

Для передачи данные заносятся в буфер FIFO передатчика. Если работа приемопередатчика разрешена, начинается передача информационного кадра с параметрами, указанными в регистре управления линией UARTLCR_H. Передача данных продолжается до опустошения буфера FIFO передатчика. После записи элемента в буфер FIFO передатчика сигнал BUSY переходит в высокое состояние. Это состояние сохраняется в течение всего времени передачи данных. В низкое состояние сигнал BUSY переходит только после того, как буфер FIFO передатчика станет пуст, а последний бит данных (включая стоповые биты) будет передан. Сигнал BUSY может находиться в высоком состоянии даже в случае, если приемопередатчик будет переведен из разрешенного состояний в запрещенное.

Для каждого бита данных (в приемной линии) производится три измерения уровня, решение принимается по мажоритарному принципу.

В случае, если приемник находился в неактивном состоянии (на линии входного сигнала UART_RXD постоянно присутствовала единица) и произошел переход входного сигнала из высокого в низкий логический уровень (обнаружен стартовый бит), включается счетчик, тактируемый сигналом Baud16, после чего отсчеты сигнала на входе приемника регистрируются каждые восемь тактов (в режиме асинхронного приемопередатчика) или каждые четыре такта (в режиме ИК-обмена данными) сигнала Baud16. Более частая выборка данных в режиме ИК-обмена связана с необходимостью корректной обработки импульсов данных согласно протоколу SIR IrDA.

Стартовый бит считается достоверным в случае, если сигнал на линии UART_RXD сохраняет низкий логический уровень в течение восьми отсчетов сигнала Baud16 с момента включения счетчика. В противном случае переход в ноль рассматривается как ложный старт и игнорируется.

В случае, если обнаружен достоверный стартовый бит, производится регистрация последовательности данных на входе приемника. Очередной бит данных фиксируются каждые 16 отсчетов тактового сигнала Baud16 (что соответствует длительности одного символа). Производится регистрация всех бит данных (согласно запрограммированным параметрам) и бита четности (если включен режим контроля четности).

Наконец, производится проверка присутствия корректного стопового бита (высокий логический уровень сигнала UART_RXD). В случае, если последнее условие не выполняется, устанавливается признак ошибки формирования кадра. После того, как слово данных принято полностью, оно заносится в буфер FIFO приемника, наряду с четырьмя битами признаков ошибки, связанных с принятым словом (см. Таблица 34-1 – Назначение бит слова данных в FIFO-буфере приемника).

34.5.6 Биты ошибки

Три бита признаков ошибки, ассоциированные с принятым символом данных, заносятся в разряды [10...8] слова данных в буфере FIFO приемника. Также предусмотрен признак ошибки переполнения буфера FIFO в разряде 11 слова данных.

Таблица 34-1 описывает назначение всех бит слова данных в FIFO-буфере приемника.

Таблица 34-1 – Назначение бит слова данных в FIFO-буфере приемника

Бит буфера FIFO	Назначение
11	Признак переполнения буфера
10	Ошибка – «разрыв линии»
9	Ошибка проверки на четность
8	Ошибка формирования кадра
7...0	Принятые данные

34.5.7 Бит переполнения буфера

Бит переполнения непосредственно не связан с конкретным символом в буфере приемника. Признак переполнения фиксируется в случае, если буфер FIFO заполнен к моменту, когда очередной символ данных полностью принят (находится в регистре сдвига). При этом данные из регистра сдвига не попадают в буфер приемника и теряются с началом приема очередного символа. Как только в буфере приемника появляется свободное место, очередной принятый символ данных заносится в буфер FIFO вместе с текущим значением признака переполнения. После успешной записи данных в буфер признак переполнения сбрасывается.

34.5.8 Запрет буфера FIFO

Предусмотрена возможность отключения FIFO буферов приемника и передатчика. В этом случае приемная и передающая сторона контроллера UART располагают лишь однобайтными буферными регистрами. Бит переполнения буфера устанавливается при этом тогда, когда очередной символ данных уже принят, однако предыдущий еще не был считан.

В настоящей реализации модуля буферы FIFO физически не отключаются, необходимая функциональность достигается за счет логических манипуляций с флагами. При этом в случае, если буфер FIFO отключен, а сдвиговый регистр передатчика пуст (не используется), запись байта данных происходит непосредственно в регистр сдвига, минуя буферный регистр.

34.5.8.1 Проверка по шлейфу

Проверка по шлейфу (замыкание выхода передатчика на вход приемника) выполняется путем установки в «1» бита LBE в регистре управления контроллером UARTCR.

34.5.9 Работа кодека ИК-обмена данными IrDA SIR

Кодек обеспечивает сопряжение асинхронного потока данных, сформированного приемопередатчиком, с полудуплексным последовательным интерфейсом IrDA SIR. Какая-либо аналоговая обработка сигнала при этом не выполняется. Назначение кодека – сформировать цифровой поток данных на вход приемника асинхронного сигнала и обработать цифровой поток данных с выхода передатчика.

Предусмотрено два режима работы:

- **В режиме IrDA** уровень логического нуля передается на линию nSIROUT в виде импульса с высоким логическим уровнем и длительностью 3/16 от выбранного периода следования бит данных. Логическая «1» при этом передается в виде постоянного низкого уровня сигнала. Сформированный выходной сигнал далее подается на передатчик ИК-сигнала, обеспечивая излучение светового импульса всякий раз при передаче нулевого бита. На приемной стороне световые импульсы воздействуют на базу фототранзистора ИК-приемника, который в результате формирует низкий логический уровень. Это, в свою очередь, обуславливает низкий уровень на входе SIRIN.

- **В режиме IrDA с пониженным энергопотреблением** длительность передаваемых импульсов ИК-излучения устанавливается в три раза выше длительности импульсов внутреннего опорного сигнала IrLPBaud16 (равной 1,63 мкс при номинальной частоте 1,8432 МГц). Данный режим активизируется путем установки бита SIRLP в регистре управления UARTCR.

Как в нормальном режиме, так и в режиме пониженного энергопотребления:

- кодирование осуществляется на основе бит данных, сформированных асинхронным передатчиком модуля;
- в ходе приема данных декодированные биты далее обрабатываются блоком асинхронного приема.

В соответствии со спецификацией физического уровня протокола IrDA SIR, обмен данными должен осуществляться в режиме полудуплекса, при этом задержка между передачей и приемом данных должна составлять не менее 10 мс. Эта задержка должна формироваться программно. Необходимость ее введения обусловлена тем, что воздействие передающего ИК-светодиода на находящийся рядом ИК-приемник может привести к искажению принимаемого сигнала или даже ввести приемный тракт в состояние насыщения. Задержка между окончанием передачи и началом приема данных именуется латентность, или время установки (готовности) приемника.

Сигнал IrLPBaud16 формируется путем деления частоты сигнала UARTCLK в соответствии с коэффициентом деления, записанным в регистре UARTILPR.

Коэффициент деления вычисляется по формуле

$$\text{Коэффициент деления} = \frac{F\_UARTCLK}{F\_IrLPBaud16'} \quad (9)$$

где номинальное значение IrLPBaud16 составляет 1,8432 МГц. Коэффициент деления должен быть выбран так, чтобы выполнялось соотношение:

$$1,42 \text{ МГц} < F\_IrLPBaud16 < 2,12 \text{ МГц}.$$

34.5.9.1 Проверка по шлейфу

Проверка по шлейфу выполняется после установки в «1» бита LBE регистра управления контроллером UARTCR с одновременной установкой в «1» бита SIRTEST регистра управления тестированием UARTTCR.

В этом режиме данные, передаваемые на выход nSIROUT, должны подаваться на вход SIRIN.

Примечание – Это единственный случай использования тестового регистра в нормальном режиме функционирования модуля.

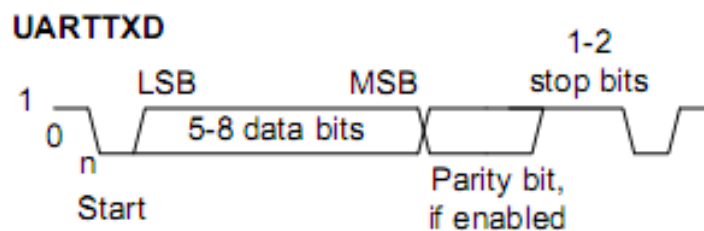


Рисунок 34-4 – Кадр передачи данных

34.5.10 Модуляция данных IrDA

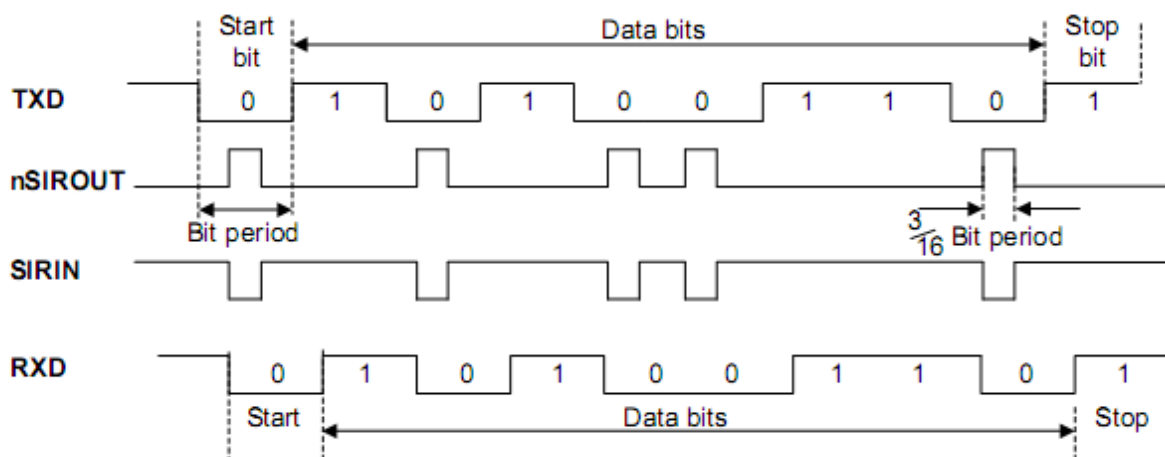


Рисунок 34-5 – Модуляция данных IrDA

34.6 Интерфейс прямого доступа к памяти

Модуль универсального асинхронного приемопередатчика оснащен интерфейсом подключения к контроллеру прямого доступа к памяти. Работа в данном режиме контролируется регистром управления DMA UARTDMACR.

Интерфейс DMA включает в себя следующие сигналы:

– **Для приема:**

– UARTRXDMASREQ – запрос передачи отдельного символа, инициируется контроллером UART. Размер символа в режиме приема данных – до 12 бит. Сигнал переводится в активное состояние в случае, если буфер FIFO приемника содержит по меньшей мере один символ.

– UARTRXDMABREQ – запрос блочного обмена данными, инициируется модулем приемопередатчика. Сигнал переходит в активное состояние в случае, если заполнение буфера FIFO приемника превысило заданный порог. Порог программируется индивидуально для каждого буфера FIFO путем записи значения в регистр UARTIFLS.

– UARTRXDMACLR – сброс запроса на DMA, инициируется модулем приемопередатчика с целью сброса принятого запроса. В случае, если был запрошен блочный обмен данными, сигнал сброса формируется в ходе передачи последнего символа данных в блоке.

– **Для передачи:**

– UARTTXDMASREQ – запрос передачи отдельного символа, инициируется модулем приемопередатчика. Размер символа в режиме передачи данных – до восьми бит. Сигнал переводится в активное состояние в случае, если буфер FIFO передатчика содержит, по меньшей мере, одну свободную ячейку.

– UARTTXDMABREQ – запрос блочного обмена данными, инициируется модулем приемопередатчика. Сигнал переводится в активное состояние в случае, если заполнение буфера FIFO передатчика ниже заданного порога. Порог программируется индивидуально для каждого буфера FIFO путем записи значения в регистр UARTIFLS.

– UARTTXDMACLR – сброс запроса на DMA, инициируется контроллером DMA с целью сброса принятого запроса. В случае, если был запрошен блочный обмен данными, сигнал сброса формируется в ходе передачи последнего символа данных в блоке.

Сигналы блочного и одноэлементного обмена данными не являются взаимно исключающими, они могут быть инициированы одновременно. Например, в случае, если заполнение данными буфера приемника превышает пороговое значение, формируется как сигнал запроса одноэлементного обмена, так и сигнал запроса блочного обмена данными. В случае, если количество данных в буфере приема меньше порогового значения, формируется только запрос одноэлементного обмена. Это бывает полезно в ситуациях, при которых объем данных меньше размера блока. Пусть, например, нужно принять 19 символов, а порог заполнения буфера FIFO установлен равным четырем. Тогда контроллер DMA осуществит четыре передачи блоков по четыре символа, а оставшиеся три символа передаст в ходе трех одноэлементных обменов.

Примечание – Для оставшихся трех символов контроллер UART не может инициировать процедуру блочного обмена.

Каждый инициированный приемопередатчиком сигнал запроса DMA остается активным до момента его сброса соответствующим сигналом DMACLR.

После снятия сигнала сброса модуль приемопередатчика вновь получает возможность сформировать запрос на DMA в случае выполнения описанных выше условий. Все запросы DMA снимаются после запрета работы приемопередатчика, а также в случае установки в ноль бита управления DMA TXDMAE или RXDMAE в регистре управления DMA UARTDMACR.

В случае запрета буферов FIFO устройство способно передавать и принимать только одиночные символы; как следствие, контроллер может инициировать DMA только в одноэлементном режиме. При этом модуль в состоянии формировать только сигналы управления DMA UARTRXDMASREQ и UARTTXDMASREQ. Для информации о запрете буферов FIFO см. описание регистра управления линией UARTLCR_H.

Когда буферы FIFO включены, обмен данными может производиться в ходе как одноэлементных, так и блочных передач данных, в зависимости от установленной величины порога заполнения буферов и их фактического заполнения. Таблица 34-2 показывает значения параметров срабатывания запросов блочного обмена UARTRXDMABREQ и UARTTXDMABREQ в зависимости от порога заполнения буфера.

Таблица 34-2 – Параметры срабатывания запросов блочного обмена данными в режиме DMA

Пороговый уровень	Длина блока обмена данными	
	Буфер передатчика (количество незаполненных ячеек)	Буфер приемника (количество заполненных ячеек)
1/8	28	4
1/4	24	8
1/2	16	16
3/4	8	24
7/8	4	28

В регистре управления DMA UARTDMACR предусмотрен бит DMAONERR, который позволяет запретить DMA от приемника в случае активного состояния линии прерывания по обнаружению ошибки UARTEINTR. При этом соответствующие линии запроса DMA – UARTRXDMASREQ и UARTRXDMABREQ переводятся в неактивное состояние (маскируются) до сброса UARTEINTR. На линии запроса DMA, обслуживающие передатчик, состояние UARTEINTR не влияет.

Ниже показаны временные диаграммы одноэлементного и блочного запросов DMA, в том числе действие сигнала DMACLR (Рисунок 34-6). Все сигналы должны быть синхронизированы с CPU_CLK. В интересах ясности изложения предполагается, что синхронизация сигналов запроса DMA в контроллере DMA не производится.

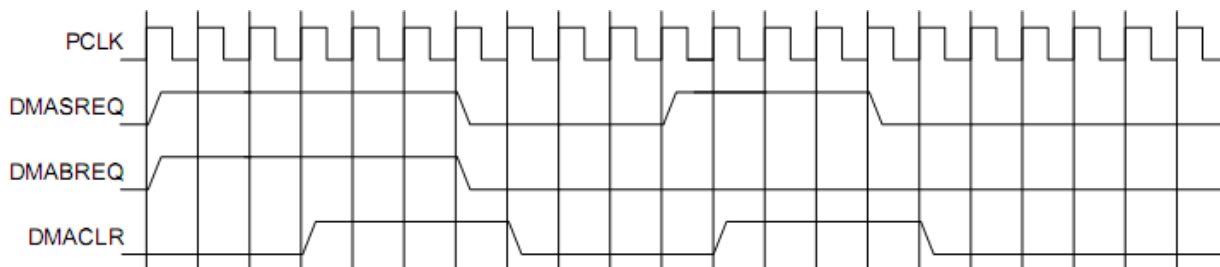


Рисунок 34-6 – Временные диаграммы одноэлементного и блочного запросов DMA

34.7 Прерывания

В модуле предусмотрено 11 маскируемых источников прерывания. В результате формируется один общий сигнал, представляющий собой комбинацию независимых сигналов, объединенных по схеме «ИЛИ».

Сигналы запроса на прерывание:

- UARTRXINTR – прерывание от приемника;
- UARTRXINTR – прерывание от передатчика;
- UARTRTINTR – прерывание по таймауту приемника;
- UARTEINTR – ошибка:
 - UARTOEINTR, переполнение буфера;
 - UARTBEINTR, прерывание приема – разрыв линии;
 - UARTPEINTR, ошибка контроля четности;
 - UARTFEINTR, ошибка в структуре кадра.
- UARTINTR – логическое «ИЛИ» сигналов UARTRXINTR, UARTRXINTR, UARTRTINTR и UARTEINTR.

Каждый из независимых сигналов запроса на прерывание может быть маскирован путем установки соответствующего бита в регистре маски UARTIMSC. Установка бита в «1» разрешает соответствующее прерывание, в 0 – запрещает.

Доступность, как индивидуальных линий, так и общей линии запроса позволяет организовать обслуживание прерываний в системе, как путем применения глобальной процедуры обработки, так и с помощью драйвера устройства, построенного по модульному принципу.

Прерывания от приемника и передатчика UARTRXINTR и UARTRXINTR выведены отдельно от прерываний по изменению состояния устройства. Это позволяет использовать сигналы запроса UARTRXINTR и UARTRXINTR для обеспечения чтения и записи данных согласованно с достижением заданного порога заполнения буферов FIFO приемника и передатчика.

Прерывание по обнаружению ошибке UARTEINTR формируется в случае возникновения той или иной ошибки приема данных. Предусмотрен ряд условий формирования признака ошибки.

Прерывание по состоянию модема представляет собой комбинацию признаков изменения отдельных линий состояния модема.

Признаки возникновения каждого из условий прерывания можно считать либо из регистра прерываний UARTRIS, либо из маскированного регистра прерываний UARTRIS.

34.7.1 UARTRXINTR

Состояние прерывания от приемника может измениться в случае возникновения одного из следующих событий:

- буфер FIFO разрешен и его заполнение достигло заданного порогового значения. В этом случае линия прерывания переходит в высокое состояние. Сигнал прерывания переходит в низкое состояние после чтения данных из буфера приемника до тех пор, пока его заполнение не станет меньше порога, либо после сброса прерывания;
- буфер FIFO запрещен (имеет размер один символ), принят один символ данных. При этом линия прерывания переходит в высокое состояние. Сигнал прерывания переходит в низкое состояние после чтения одного байта данных, либо после сброса прерывания.

34.7.2 UARTTXINTR

Состояние прерывания от передатчика может измениться в случае возникновения одного из следующих событий:

- буфер FIFO разрешен и его заполнение меньше или равно заданному пороговому значению. В этом случае линия прерывания переходит в высокое состояние. Сигнал прерывания переходит в низкое состояние после записи данных в буфера передатчика до тех пор, пока его заполнение не станет больше порога, либо после сброса прерывания;
- буфер FIFO запрещен (имеет размер один символ), данные в буферном регистре передатчика отсутствуют. При этом линия прерывания переходит в высокое состояние. Сигнал прерывания переходит в низкое состояние после записи одного байта данных, либо после сброса прерывания.

Для занесения данных в буфер FIFO передатчика необходимо записать данные в буфер либо перед разрешением работы приемопередатчика и прерываний, либо после разрешения работы приемопередатчика и прерываний.

Примечание – Прерывание передатчика работает по фронту, а не по уровню сигнала. В случае, если модуль и прерывания от него разрешены до осуществления записи данных в буфер FIFO передатчика, прерывание не формируется. Прерывание возникает только при опустошении буфера FIFO.

34.7.3 UARTRTINTR

Прерывание по таймауту приемника возникает в случае, если буфер FIFO приемника не пуст, и на вход приемника не поступало новых данных в течение периода времени, необходимого для передачи 32 бит. Прерывание по таймауту снимается либо после считывания данных из буфера приемника до его опустошения (или считывания одного байта в случае, если буфер FIFO запрещен), либо путем записи 1 в соответствующий бит регистра сброса прерывания UARTICR.

34.7.4 UARTEINTR

Прерывание по обнаружению ошибки возникает в случае ошибки при приеме данных. Оно может быть вызвано рядом факторов:

- ошибка в структуре кадра;
- ошибка контроля четности;
- разрыв линии;
- переполнение буфера.

Причину возникновения прерывания можно определить, прочитав содержимое регистра прерываний UARTRIS, либо содержимое маскированного регистра прерываний UARTRMIS.

Сброс прерывания осуществляется путем записи соответствующих бит в регистр сброса прерывания UARTRICR. За прерываниями по обнаружению ошибки закреплены биты с 7 по 10.

34.7.5 UARTRINTR

Все описанные сигналы запроса на прерывание скомбинированы в общую линию путем объединения по схеме ИЛИ сигналов UARTRXINTR, UARTRTXINTR, UARTRRTINTR и UARTEINTR с учетом маскирования. Общий выход может быть подключен к системному контроллеру прерывания, что позволит ввести дополнительное маскирование запросов на уровне периферийных устройств.

34.8 Программное управление модулем

34.8.1 Общая информация

Следующая информация применима ко всем регистрам контроллера:

- Базовый адрес контроллера не фиксирован и может быть различным в разных системах. Смещение каждого регистра относительно базового адреса постоянно.
- Не следует пытаться получить доступ к зарезервированным или неиспользуемым адресам. Это может привести к непредсказуемому поведению модуля.
- За исключением специально оговоренных в настоящем документе случаев:
 - не следует изменять значения не определенных в документе разрядов регистров;
 - не следует использовать значения не определенных в документе разрядов регистров;
 - все биты регистров (за исключением специально оговоренных случаев, прим. перев) устанавливаются в значение 0 после сброса по включению питания или системного сброса.
- Столбец «Тип» (Таблица 34-3) определяет режим доступа к регистру в соответствии с обозначениями:
 - RW – чтение и запись;
 - RO – только чтение;
 - WO – только запись.

34.8.2 Обобщенные данные о регистрах устройства

Данные о регистрах модуля универсального асинхронного приемопередатчика приведены в таблице ниже (Таблица 34-3).

Таблица 34-3 – Обобщенные данные о регистрах устройства

Смещение	Наименование	Тип	Значение после сброса	Размер, бит	Описание
0x40030000	MDR_UART1				Контроллер UART1
0x40038000	MDR_UART2				Контроллер UART2
0x400D0000	MDR_UART3				Контроллер UART3
0x000	DR	RW	0x---	12/8	MDR_UARTx->DR Регистр данных
0x004	RSR_ECR	RW	0x0	4/0	MDR_UARTx->RSR_ECR Регистр состояния приемника / Сброс ошибки приемника
0x008- 0x014					Зарезервировано
0x018	FR	RO	0b-10010---	9	MDR_UARTx->FR Регистр флагов
0x01C					Зарезервировано
0x020	ILPR	RW	0x00	8	MDR_UARTx->ILPR Регистр управления ИК-обменом в режиме пониженного энерго- потребления
0x024	IBRD	RW	0x0000	16	MDR_UARTx->IBRD Целая часть делителя скорости обмена данными
0x028	FBRD	RW	0x00	6	MDR_UARTx->FBRD Дробная часть делителя скорости обмена данными
0x02C	LCR_H	RW	0x00	8	MDR_UARTx->LCR_H Регистр управления линией
0x030	CR	RW	0x0300	16	MDR_UARTx->CR Регистр управления
0x034	IFLS	RW	0x12	6	MDR_UARTx->IFLS Регистр порога прерывания по заполнению буфера FIFO
0x038	IMSC	RW	0x000	11	MDR_UARTx->IMSC Регистр маски прерывания
0x03C	RIS	RO	0x00-	11	MDR_UARTx->RIS Регистр состояния прерываний
0x040	MIS	RO	0x00-	11	MDR_UARTx->MIS Регистр состояния прерываний с маскированием
0x044	ICR	WO	-	11	MDR_UARTx->ICR Регистр сброса прерывания
0x048	DMACR	RW	0x00	3	MDR_UARTx->DMACR Регистр управления DMA

34.8.2.1 MDR_UARTx->DR

Регистр данных

В ходе передачи данных:

Если буфер FIFO передатчика разрешен, то слово данных, записанное в рассматриваемый регистр, направляется в буфер FIFO передатчика.

В противном случае, записанное слово фиксируется в буферный регистр передатчика (последний элемент буфера FIFO).

Операция записи в регистр инициирует передачу данных. Слово данных предваряется стартовым битом, дополняется битом контроля четности (если режим контроля четности включен) и стоповым битом. Сформированное слово отправляется в линию передачи данных.

В ходе приема данных:

Если буфер FIFO приемника разрешен, байт данных и четыре бита состояния (разрыв, ошибка формирования кадра, четность, переполнение) сохраняются в 12-битном буфере.

В противном случае байт данных и биты состояния записываются в буферный регистр (последний элемент буфера FIFO).

Полученные из линии связи байты данных считываются путем чтения из регистра UARTDR принятых данных совместно с соответствующими битами состояния. Информация о состоянии также может быть получена путем чтения регистра UATRISR/UARTECR (Таблица 34-4).

Таблица 34-4 – Описание бит регистра UARTDR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15...12	-	Зарезервировано
11	OE	Переполнение буфера приемника. Бит устанавливается в 1 в случае, если на вход приемника поступают данные, в то время, как буфер заполнен. Сбрасывается в 0 после того, как в буфере появится свободное место
10	BE	Разрыв линии. Устанавливается в 1 при обнаружении признака разрыва линии, то есть в случае наличия низкого логического уровня на входе приемника в течение времени, большего, чем длительность передачи полного слова данных (включая стартовый, стоповый биты и бит проверки на четность). При включенном FIFO данная ошибка ассоциируется с последним символом, поступившим в буфер. В случае обнаружения разрыва линии в буфер загружается только один нулевой символ, прием данных возобновляется только после перехода линии в логическую 1 и последующего обнаружения корректного стартового бита
9	PE	Ошибка контроля четности. Устанавливается в 1 в случае, если четность принятого символа данных не соответствует установкам бит EPS и SPS в регистре управления линией UARTLCR_H. При включенном FIFO данная ошибка ассоциируется с последним символом, поступившим в буфер

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
8	FE	Ошибка в структуре кадра. Устанавливается в 1 в случае, если в принятом символе не обнаружен корректный стоповый бит (корректный стоповый бит равен 1). При включенном FIFO данная ошибка ассоциируется с последним символом, поступившим в буфер
7...0	DATA	Принимаемые данные (чтение). Передаваемые данные (запись)

Примечание – Необходимо запрещать работу приемопередатчика перед любым перепрограммированием его регистров управления. Если приемопередатчик переводится в отключенное состояние во время передачи или приема символа, то перед остановкой он завершает выполняемую операцию.

34.8.2.2 MDR_UARTx->RSR_ECR

Регистр состояния приемника / сброса ошибки

Состояние приемника также может быть считано из регистра UARTRSR. В этом случае информация о состоянии признаков разрыва линии, ошибки контроля четности и ошибки в структуре кадра относится к последнему символу, считанному из регистра данных UARTDR. С другой стороны, признак переполнения буфера устанавливается немедленно после возникновения этого состояния (и не связан с последним, считанным из регистра UARTDR байтом данных).

Запись в регистр UARTECR приводит к сбросу признаков ошибок переполнения, четности, структуры кадра, разрыва линии. Кроме того, все эти признаки устанавливаются в «0» после сброса устройства.

Таблица 34-5 показывает назначение битов регистра UARTRSR/UARTECR.

Таблица 34-5 – Описание бит регистра UARTRSR/UARTECR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
7...4	-	Резерв, при чтении результат не определен
3	OE	Переполнение буфера приемника. Бит устанавливается в 1 в случае, если на вход приемника поступают данные, в то время как буфер заполнен. Сбрасывается в 0 после записи в регистр UARTECR. Содержимое буфера остается верным, так как перезаписан был только регистр сдвига. Центральный процессор должен считать данные для того, чтобы освободить буфер FIFO
2	BE	Разрыв линии. Устанавливается в 1 при обнаружении признака разрыва линии, то есть в случае наличия низкого логического уровня на входе приемника в течение времени, большего, чем длительность передачи полного слова данных (включая стартовый, стоповый биты и бит проверки на четность). Бит сбрасывается в 0 после записи в регистр UARTECR. При включенном FIFO данная ошибка ассоциируется с символом, находящемся на вершине буфера. В случае обнаружения разрыва линии в буфер загружается только один нулевой символ, прием данных возобновляется только после перехода линии в логическую 1 и последующего обнаружения корректного стартового бита

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
1	PE	Ошибка контроля четности. Устанавливается в 1 в случае, если четность принятого символа данных не соответствует установкам бит EPS и SPS в регистре управления линией UARTLCR_H (стр. 3-12). Бит сбрасывается в 0 после записи в регистр UARTECR. При включенном FIFO данная ошибка ассоциируется с символом, находящимся на вершине буфера
0	FE	Ошибка в структуре кадра. Устанавливается в 1 в случае, если в принятом символе не обнаружен корректный стоповый бит (корректный стоповый бит равен 1). Бит сбрасывается в 0 после записи в регистр UARTECR. При включенном FIFO данная ошибка ассоциируется с символом, находящимся на вершине буфера

Примечания

1 Перед чтением регистра состояния UARTRSR необходимо считать данные, принятые из линии, путем обращения к регистру данных UARTDR. Противоположная последовательность действий не допускается, так как регистр UARTRSR обновляет свое состояние только после чтения регистра UARTDR. Вместе с тем, информация о состоянии приемника может быть получена непосредственно из регистра данных UARTDR.

2 Запись в регистр UARTRSR/UARTECR любого кода сбрасывает признаки ошибок формирования кадра, проверки на четность, разрыва линии и переполнения буфера.

34.8.2.3 MDR_UARTx->FR

Регистр флагов

После сброса биты регистра флагов TXFF, RXFF и BUSY устанавливаются в «0», а биты TXFE и RXFE – в «1». Таблица 34-6 отображает назначение битов регистра UARTFR.

Таблица 34-6 – Описание бит регистра UARTFR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15...9	-	Резерв. Не модифицируйте. При чтении заполняются нулями
8	-	Зарезервировано
7	TXFE	Буфер FIFO передатчика пуст. Значение бита зависит от состояния бита FEN регистра управления линией UARTLCR_H. Если буфер FIFO запрещен, бит устанавливается в 1, когда буферный регистр передатчика пуст. В противном случае он равен 1, если пуст буфер FIFO передатчика. Данный бит не дает никакой информации о наличии данных в регистре сдвига передатчика
6	RXFF	Буфер FIFO приемника заполнен. Значение бита зависит от состояния бита FEN регистра управления линией UARTLCR_H. Если буфер FIFO запрещен, бит устанавливается в 1, когда буферный регистр приемника занят. В противном случае он равен 1, если заполнен буфер FIFO приемника

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
5	TXFF	Буфер FIFO передатчика заполнен. Значение бита зависит от состояния бита FEN регистра управления линией UARTLCR_H. Если буфер FIFO запрещен, бит равен 1, когда буферный регистр передатчика занят. В противном случае он равен 1, если заполнен буфер FIFO передатчика
4	RXFE	Буфер FIFO приемника пуст. Значение бита зависит от состояния бита FEN регистра управления линией UARTLCR_H. Если буфер FIFO запрещен, бит устанавливается в 1, когда буферный регистр приемника пуст. В противном случае он равен 1, если пуст буфер FIFO приемника
3	BUSY	UART занят. Бит равен 1 в случае, если контроллер передает в линию данные. Бит остается установленным до тех пор, пока данные, включая стоповые биты, не будут полностью переданы. Кроме того, бит занятости устанавливается в 1 при наличии данных в буфере FIFO передатчика, вне зависимости от состояния приемопередатчика (даже если он запрещен)
2...0	-	Зарезервировано

34.8.2.4 MDR_UARTx->ILPR

Регистр управления ИК-обменом в режиме пониженного энергопотребления

Этот восьмиразрядный регистр, доступный для чтения и записи, содержит значение коэффициента деления частоты UARTCLK, для формирования тактового сигнала IrLPBaud16. Таблица 34-7 отображает назначение битов регистра UARTILPR.

Требуемое значение коэффициента деления для формирования сигнала IrLPBaud16 вычисляется по формуле

$$ILPDVSR = \frac{F\_UARTCLK}{F\_IrLPBaud16}, \quad (10)$$

где номинальное значение частоты F_IrLPBaud16 составляет 1,8432 МГц.

Коэффициент деления должен быть установлен таким образом, чтобы выполнялось соотношение:

$$1,42 \text{ МГц} < F\_IrLPBaud16 < 2,12 \text{ МГц},$$

что, в свою очередь, гарантирует формирование кодеком импульсов данных с длительностью 1,41 – 2,11 мкс (в три раза длиннее периода сигнала IrLPBaud16).

Таблица 34-7 – Описание бит регистра UARTILPR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
7...0	ILPDVSR	Коэффициент деления частоты UARTCLK, для формирования тактового сигнала IrLPBaud16. После сброса устанавливается в 0. Примечание – Коэффициент 0 – запрещенное значение. В случае его установки импульсы IrLPBaud16 формироваться не будут

Примечание – В интересах подавления помех при работе в режиме IrDA с пониженным энергопотреблением кодек игнорирует поступающие на вход SIRIN импульсы с длительностью, меньшей трех периодов сигнала IrLPBaud16.

34.8.2.5 MDR_UARTx->IBRD

Регистр целой части делителя скорости передачи данных

Таблица 34-8 отображает назначение разрядов регистра UARTBIRD.

Таблица 34-8 – Описание бит регистра UARTBIRD

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15...0	BAUDDIV_INT	Целая часть коэффициента деления частоты для формирования тактового сигнала передачи данных. После сброса устанавливается в 0

34.8.2.6 MDR_UARTx->FBRD

Регистр дробной части делителя скорости передачи данных,

Таблица 34-9 отображает назначение битов регистра UARTBFRD.

Таблица 34-9 – Описание бит регистра UARTBFRD

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
5...0	BAUDDIV_FRAC	Дробная часть коэффициента деления частоты для формирования тактового сигнала передачи данных. После сброса устанавливается в 0

Коэффициент деления вычисляется по формуле

$$\text{BAUDDIV} = \frac{\text{FUARTCLK}}{16 \cdot \text{Baud\_rate}}, \quad (11)$$

где FUARTCLK – тактовая частота контроллера UART;

Baud_rate – требуемая скорость передачи данных.

Коэффициент BAUDDIV состоит из целой и дробной частей – BAUDDIV_INT и BAUDDIV_FRAC, соответственно.

Примечания

1 Изменения содержимого регистров UARTIBRD и UARTFBRD вступают в силу только после завершения передачи и приема текущего символа данных;

2 Минимальный допустимый коэффициент деления – 1, максимальный 65535 ($2^{16} - 1$). Таким образом, значение UARTIBRD, равное 0, является недопустимым, при этом значение регистра UARTFBRD игнорируется;

3 Аналогично, при UARTIBRD равном 65535 (0xFFFF), значение UARTFBRD не может быть больше нуля. Невыполнение этого условия приведет к прерыванию приема или передачи.

Далее приведен пример вычисления коэффициента деления.

Пример: Вычисление коэффициента деления

Пусть требуемая скорость передачи данных составляет 230400 бит/с, частота тактового сигнала UARTCLK равна 4 МГц. Тогда:

$$\text{Коэффициент деления} = (4 \cdot 106) / (16 \cdot 230400) = 1,085$$

$$\text{Таким образом, BRDI} = 1, \text{ BRDF} = 0,085$$

Следовательно, значение, записываемое в регистр UARTBFRD, равно

$$m = \text{integer}((0,085 \cdot 64) + 0.5) = 5$$

$$\text{Реальное значение коэффициента деления} = 1 + 5 / 64 = 1,078$$

$$\text{Реальная скорость передачи данных} = (4 \cdot 106) / (16 \cdot 1,078) = 231911 \text{ бит/с}$$

$$\text{Ошибка установки скорости} = (231911 - 230400) / 230400 \cdot 100 \% = 0,656 \%$$

Максимальная ошибка установки скорости передачи данных с использованием шестизрядного регистра UARTBFRD = $1 / 64 \cdot 100 \% = 1,56 \%$. Такая ошибка возникает в случае $m = 1$, при этом разница накапливается в течение 64 тактовых интервалов.

В следующей таблице (Таблица 34-10) представлены значения коэффициента деления для типичных скоростей передачи данных при частоте UARTCLK = 7,3728 МГц. При таких параметрах дробная часть коэффициента деления не используется, следовательно, в регистр UARTFBRD должен быть записан ноль.

Таблица 34-10 – Коэффициенты деления при частоте UARTCLK = 7,3728 МГц

Коэффициент деления	Скорость передачи данных
0x0001	460800
0x0002	230400
0x0004	115200
0x0006	76800
0x0008	57600
0x000C	38400
0x0018	19200
0x0020	14400
0x0030	9600
0x00C0	2400
0x0180	1200
0x105D	110

В таблице ниже приведены значения коэффициента деления для типичных скоростей передачи данных при частоте UARTCLK = 4 МГц.

Таблица 34-11 – Коэффициенты деления при частоте UARTCLK = 4 МГц

Целая часть	Дробная часть	Требуемая скорость	Реальная скорость	Ошибка, %
0x001	0x05	230400	231911	0,656
0x002	0x0B	115200	115101	0,086
0x003	0x10	76800	76923	0,160
0x006	0x21	38400	38369	0,081
0x011	0x17	14400	14401	0,007
0x068	0x0B	2400	2400	~ 0
0x8E0	0x2F	110	110	~ 0

34.8.2.7 MDR_UARTx->LCR_H

Регистр управления линией

Данный регистр обеспечивает доступ к разрядам с 29 по 22 регистра UARTLCR. При сбросе все биты регистра UARTLCR_H обнуляются.

Таблица 34-12 отображает назначение битов регистра UARTLCR_H.

Таблица 34-12 – Регистр UARTLCR_H

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15...8	-	Резерв. Не модифицируйте. При чтении выдаются нули.
7	SPS	Передача бита четности с фиксированным значением. 0 – запрещена; 1 – на месте бита четности передается инверсное значение бита EPS, оно же проверяется при приеме данных. (При EPS=0 на месте бита четности передается 1, при EPS=1 – передается 0). Значение бита SPS не играет роли в случае, если битом PEN формирование и проверка бита четности запрещен
6...5	WLEN	Длина слова – количество передаваемых или принимаемых информационных бит в кадре: 0b11 – 8 бит; 0b10 – 7 бит; 0b01 – 6 бит; 0b00 – 5 бит;
4	FEN	Разрешение работы буфера FIFO приемника и передатчика. 0 – запрещено; 1 – разрешено
3	STP2	Режим передачи двух стоповых бит. 0 – один стоповый бит; 1 – два стоповых бита. Приемник не проверяет наличие дополнительного стопового бита в кадре
2	EPS	Четность/нечетность. 0 – бит четности дополняет количество единиц в информационной части кадра до нечетного; 1 – до четного числа. Значение бита EPS не играет роли в случае, если битом PEN формирование и проверка бита четности запрещена
1	PEN	Разрешение проверки четности. 0 – кадр не содержит бита четности; 1 – бит четности передается в кадре и проверяется при приеме данных
0	BRK	Разрыв линии. Если этот бит установлен в 1, то по завершении передачи текущего символа на выходе UARTTXD устанавливается низкий уровень сигнала. Для правильного выполнения этой операции программное обеспечение должно обеспечить передачу сигнала разрыва в течение, как минимум, времени передачи двух информационных кадров. В нормальном режиме функционирования бит должен быть установлен в 0

Содержимое регистров UARTLCR_H, UARTIBRD и UARTFBRD совместно образует общий 30-разрядный регистр UARTLCR, который обновляется по стробу, формируемому при записи в UARTLCR_H. Таким образом, для того, чтобы изменение параметров коэффициента деления частоты обмена данными вступило в силу, после изменения значения регистров UARTIBRD и/или UARTFBRD необходимо осуществить запись данных в регистр UARTLCR_H.

Примечания

- 1 Изменение значений трех регистров можно осуществить корректно двумя способами:
 - запись UARTIBRD, запись UARTFBRD, запись UARTLCR_H;
 - запись UARTFBRD, запись UARTIBRD, запись UARTLCR_H.
- 2 Для того, чтобы изменить значение лишь одного из регистров (UARTIBRD или UARTFBRD) необходимо выполнить следующий шаг:
 - запись UARTIBRD (или UARTFBRD), запись UARTLCR_H.

Таблица 34-13 показывает таблицу истинности для бит управления контролем четности PEN, EPS и SPS регистра управления линией UARTLCR_H.

Таблица 34-13 – Управление режимом контроля четности

PEN	EPS	SPS	Бит контроля четности
0	X	X	Не передается, не проверяется
1	1	0	Проверка четности слова данных
1	0	0	Проверка нечетности слова данных
1	0	1	Бит четности постоянно равен 1
1	1	1	Бит четности постоянно равен 0

Примечания

- 1 Регистры UARTLCR_H, UARTIBRD и UARTFBRD не должны изменяться:
 - при разрешенной работе приемопередатчика;
 - во время завершения приема или передачи данных в процессе остановки (перевода в запрещенное состояние) приемопередатчика.
- 2 Целостность данных в буферах FIFO не гарантируется в следующих случаях:
 - после установки бита разрыва линии BRK;
 - если программное обеспечение произвело остановку приемопередатчика при наличии данных в буферах FIFO после его повторного перевода в разрешенное состояние.

34.8.2.8 MDR_UARTx->CR

Регистр управления

После сброса все биты регистра управления, за исключением бит 9 и 8 устанавливаются в нулевое состояние. Биты 9 и 8 устанавливаются в единичное состояние. Таблица 34-14 отображает назначение битов регистра UARTCR.

Таблица 34-14 – Регистр управления UARTCR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15...10	-	Зарезервировано
9	RXE	Прием разрешен. Установка бита в 1 разрешает работу приемника. Прием данных осуществляется либо по интерфейсу асинхронного последовательного обмена, либо по интерфейсу ИК-обмена SIR, в зависимости от значения бита SIREN. В случае перевода приемопередатчика в запрещенное состояние в ходе приема данных, он завершает прием текущего символа перед остановкой
8	TXE	Передача разрешена. Установка бита в 1 разрешает работу передатчика. Передача осуществляется либо по интерфейсу асинхронного последовательного обмена, либо по интерфейсу ИК-обмена SIR, в зависимости от значения бита SIREN. В случае перевода приемопередатчик в запрещенное состояние в ходе передачи данных, он завершает передачу текущего символа: перед остановкой
7	LBE	0 – запрещено; 1 – шлейф разрешен. В режиме разрешенного шлейфа: Если установлены бит SIREN=1 и бит регистра управления тестированием UARTTCR SIRTEST=1, то сигнал с выхода кодека nSIROUT инвертируется и подается на вход кодека SIRIN. Бит SIRTEST устанавливается в 1 для того, чтобы вывести устройство из полудуплексного режима, характерного для интерфейса SIR. После окончания тестирования по шлейфу бит SIRTEST должен быть установлен в 0. Если бит SIRTEST=0, то выходная линия передатчика UARTTXD коммутируется на вход приемника UARTRXD. Как в режиме SIR, так и в режиме UART, выходные линии состояния модема коммутируются на соответствующие входные линии. После сброса бит устанавливается в 0
6...3	-	Резерв. Не модифицируйте. При чтении выдаются нули.
2	SIRLP	Выбор режима ИК-обмена с пониженным энергопотреблением: 0 – длительность импульсов данных равна 3/16 длительности передачи бита; 1 – длительность импульсов данных равна трем тактам сигнала IrLPBaud16 вне зависимости от выбранной скорости передачи данных. Выбор этого режима снижает энергопотребление, однако может привести к уменьшению дальности связи
1	SIREN	Разрешение работы кодека ИК-передачи данных IrDA SIR: 0 – . Сигнал nSIROUT находится в низком состоянии, данные на входе SIRIN не обрабатываются. 1 – разрешено. Данные передаются на выход nSIROUT и принимаются с входа SIRIN. Линия UARTTXD находится в

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
		высоком состоянии. Данные на входе UARTRXD и линиях состояния модема не обрабатываются. В случае, если UARTEN=0 значение бита не играет роли
0	UARTEN	Разрешение работы приемопередатчика: 0 – работа запрещена. Перед остановкой завершается прием и/или передача обрабатываемого в текущий момент символа. 1 – работа разрешена. Производится обмен данными либо по линиям асинхронного обмена, либо по линиям ИК-обмена SIR, в зависимости от состояния бита SIREN
Примечания 1 Для того, чтобы разрешить передачу данных, необходимо установить в логическую «1» биты TXE и UARTEN. Аналогично, для разрешения приема данных необходимо установить в «1» биты RXE и UARTEN. 2 Рекомендуется следующая последовательность действий для программирования регистров управления: – остановите работу приемопередатчика; – дождитесь окончания приема и/или передачи текущего символа данных; – сбросьте буфер передатчика путем установки бита FEN регистра UARTLCR_H в 0; – измените настройки регистра UARTCR; – возобновите работу приемопередатчика		

34.8.2.9 MDR_UARTx->IFLS

Регистр порога прерывания по заполнению буфера FIFO

Данный регистр используется для установки порогового значения заполнения буферов передатчика и приемника, по достижению которых генерируется сигнал прерывания UARTRXINTR или UARTTXINTR, соответственно. Прерывание генерируется в момент перехода величины заполнения буфера через заданное значение.

После сброса в регистре устанавливается порог, соответствующий заполнению половины буфера. Формат регистра UARTIFLS и значения его бит представлены в таблице.

Таблица 34-15 отображает назначение битов регистра UARTIFLS.

Таблица 34-15 – Регистр UARTIFLS

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...6	-	Резерв. Не модифицируйте. При чтении выдаются нули.
5...3	RXIFLSEL	Порог прерывания по заполнению буфера приемника: b000 = буфер заполнен на 1/8; b001 = буфер заполнен на 1/4; b010 = буфер заполнен на 1/2; b011 = буфер заполнен на 3/4; b100 = буфер заполнен на 7/8; b101-b111 = резерв

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
2...0	TXIFLSEL	Порог прерывания по заполнению буфера передатчика: b000 = буфер заполнен на 1/8; b001 = буфер заполнен на 1/4; b010 = буфер заполнен на 1/2; b011 = буфер заполнен на 3/4; b100 = буфер заполнен на 7/8; b101-b111 = резерв. Также стоит помнить, что в случае если сдвиговый регистр передатчика пуст, то слово, записанное в FIFO, будет сразу же переписано в сдвиговый регистр. Следовательно, для генерирования события прерывания от передатчика блока UART необходимо произвести запись в FIFO такого количества слов, которое превысит установленный порог хотя бы на одно слово с учетом описанного случая

34.8.2.10 MDR_UARTx->IMSC

Регистр установки сброса маски прерывания

При чтении выдается текущее значение маски. При записи производится установка или сброс маски на соответствующее прерывание.

После сброса все биты регистра маски устанавливаются в нулевое состояние.

Таблица 34-16 отображает назначение битов регистра UARTIMSC.

Таблица 34-16 – Регистр UARTIMSC

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...11	-	Зарезервировано. Не модифицируйте. При чтении выдаются нули
10	OEIM	Маска прерывания по переполнению буфера UARTOEINTR: 1 – установлена; 0 – сброшена
9	BEIM	Маска прерывания по разрыву линии UARTBEINTR: 1 – установлена; 0 – сброшена
8	PEIM	Маска прерывания по ошибке контроля четности UARTPEINTR: 1 – установлена; 0 – сброшена
7	FEIM	Маска прерывания по ошибке в структуре кадра UARTFEINTR: 1 – установлена; 0 – сброшена
6	RTIM	Маска прерывания по таймауту приема данных UARTRTINTR: 1 – установлена; 0 – сброшена
5	TXIM	Маска прерывания от передатчика UARTTXINTR. 1 – установлена; 0 – сброшена

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
4	RXIM	Маска прерывания от приемника UARTRXINTR. 1 – установлена; 0 – сброшена
3...0	-	Зарезервировано

34.8.2.11 MDR_UARTx->RIS

Регистр состояния прерываний

Этот регистр доступен только для чтения и содержит текущее состояние прерываний без учета маскирования. Данные, записываемые в регистр, игнорируются.

Предупреждение. После сброса все биты регистра, за исключением бит прерывания по состоянию модема (биты с 3 по 0), устанавливаются в 0. Значение бит прерывания по состоянию модема после сброса не определено.

Таблица 34-17 отображает назначение битов регистра UARTRIS.

Таблица 34-17 – Регистр UARTRIS

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...11	-	Зарезервировано. Не модифицируйте. При чтении выдаются нули
10	OERIS	Состояние прерывания по переполнению буфера UARTOEINTR
9	BERIS	Состояние прерывания по разрыву линии UARTBEINTR
8	PERIS	Состояние прерывания по ошибке контроля четности UARTPEINTR
7	FERIS	Состояние прерывания по ошибке в структуре кадра UARTFEINTR
6	RTRIS	Состояние прерывания по таймауту приема данных UARTRTINTR Примечание – Бит RTRIS может быть установлен только при установленной маске прерывания по таймауту приема данных UARTRTINTR в регистре UARTIMSC. Это вызвано тем, что сигнал маски прерывания по таймауту используется в качестве разрешения перехода в режим пониженного энергопотребления. Чтение состояния прерывания по таймауту из регистров UARTRIS и UARTRIS приводит к одинаковым результатам
5	TXRIS	Состояние прерывания от передатчика UARTTXINTR
4	RXRIS	Состояние прерывания от приемника UARTRXINTR
3...0	-	Зарезервировано

34.8.2.12 MDR_UARTx->MIS

Регистр маскированного состояния прерываний

Этот регистр доступен только для чтения и содержит текущее состояние прерываний с учетом маскирования. Данные, записываемые в регистр, игнорируются.

После сброса все биты регистра, за исключением бит прерывания по состоянию модема (биты с 3 по 0), устанавливаются в 0. Значение бит прерывания по состоянию модема после сброса не определено.

Таблица 34-18 отображает назначение битов регистра UARTMIS.

Таблица 34-18 – Регистр UARTMIS

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...11	-	Зарезервировано. Не модифицируйте. При чтении выдаются нули
10	OEMIS	Маскированное состояние прерывания по переполнению буфера UARTOEINTR
9	BEMIS	Маскированное состояние прерывания по разрыву линии UARTBEINTR
8	PEMIS	Маскированное состояние прерывания по ошибке контроля четности UARTPEINTR
7	FEMIS	Маскированное состояние прерывания по ошибке в структуре кадра UARTFEINTR
6	RTMIS	Маскированное состояние прерывания по таймауту приема данных UARTRTINTR
5	TXMIS	Маскированное состояние прерывания от передатчика UARTTXINTR
4	RXMIS	Маскированное состояние прерывания от приемника UARTRXINTR
3...0	-	Зарезервировано

34.8.2.13 MDR_UARTx->ICR

Регистр сброса прерываний

Этот регистр доступен только для записи и предназначен для сброса признака прерывания по заданному событию путем записи 1 в соответствующий бит. Запись нуля в любой из разрядов регистра игнорируется.

Таблица 34-19 отображает назначение битов регистра UARTICR.

Таблица 34-19 – Регистр UARTICR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...11	-	Зарезервировано. Не модифицируйте. При чтении выдаются нули
10	OEIC	Сброс прерывания по переполнению буфера UARTOEINTR
9	BEIC	Сброс прерывания по разрыву линии UARTBEINTR
8	PEIC	Сброс прерывания по ошибке контроля четности UARTPEINTR
7	FEIC	Сброс прерывания по ошибке в структуре кадра UARTFEINTR
6	RTIC	Сброс прерывания по таймауту приема данных UARTRTINTR
5	TXIC	Сброс прерывания от передатчика UARTTXINTR
4	RXIC	Сброс прерывания от приемника UARTRXINTR
3...0	-	Зарезервировано
Примечание – Для организации надежного сброса статусных флагов прерываний необходимо учитывать требования, изложенные в подразделе 36.3		

34.8.2.14 MDR_UARTx->DMACR

Регистр управления прямым доступом к памяти

Регистр доступен по чтению и записи. После сброса все биты регистра обнуляются.

Таблица 34-20 отображает назначение бит регистра DMACR.

Таблица 34-20 – Регистр DMACR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...3	-	Зарезервировано. Не модифицируйте. При чтении выдаются нули
2	DMAONERR	Если бит установлен в 1, то в случае возникновения прерывания по обнаружению ошибки блокируются запросы DMA от приемника UARTRXDMASREQ и UARTRXDMABREQ
1	TXDMAE	Использование DMA при передаче. Если бит установлен в 1, то разрешено формирование запросов DMA для обслуживания буфера FIFO передатчика
0	RXDMAE	Использование DMA при приеме. Если бит установлен в 1, то разрешено формирование запросов DMA для обслуживания буфера FIFO приемника

35 Контроллер SDIO

Контроллер SDIO – специализированный контроллер для работы с картами памяти SD и разрядностью шины данных от 1 до 4 бит. Поддерживается работа контроллера как в режиме ведущего (хоста), так и в режиме ведомого (карты памяти).

35.1 Описание функционирования контроллера

35.1.1 Передача команды (по спецификации SDIO)

Для передачи команды по интерфейсу SDIO следует настроить режим работы контроллера (регистр SD_CR), заполнить регистр команд (SD_CMDDR), обнулить регистр контрольной суммы (SD_CMDCRC), задать количество бит которые надо передать по линии команд(регистр SD_CMDtransfer), а затем выставить бит WORK2 в «1». Формат передаваемой команды по спецификации SDIO представлен ниже (Рисунок 35-1).

При заполнении FIFO буфера команд следует учитывать, что команда передается старшим битом младшего байта вперед. Младший байт должен быть формата 0b01xxxxxx.

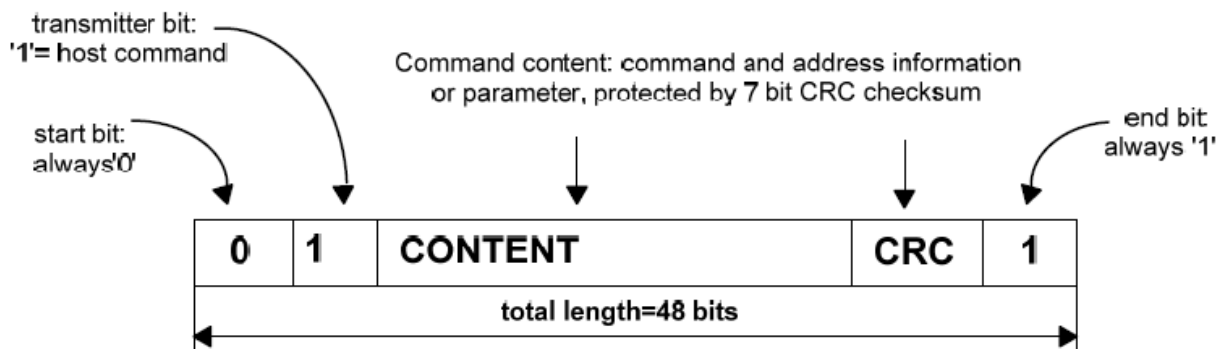


Рисунок 35-1 – Формат передаваемой команды по спецификации SDIO в режиме ведущего

Последовательность действий для передачи команды:

- включить контроллер SDIO (бит CR_SDE);
- если бит WORK2 установлен в «1», то следует либо дождаться окончания передачи прошлой команды, либо вручную сбросить этот бит;
- настроить направление передачи команды (бит DIRCMD) и установить бит ожидания стартового бита в линии CMD (бит SBITCMD);
- заполнить регистр команд ;
- инициализировать нулевым значением регистр контрольной суммы линии команд;
- задать длину передаваемой команды, с учетом стартового бита, CRC и стопового бита (т.е. по спецификации SDIO 48 бит);
- установить бит WORK2 в «1».

Пример инициализации передачи команды в соответствии со спецификацией SDIO:

```
SDIO->SD_CR = SDIO_SD_CR_SDE; /* SDIO ON, Master */
do { /* Wait while CmdActive bit is set */
SDIO->SD_CR &= ~SDIO_SD_CR_WORK2; /* Reset Work2 */
} while (SDIO->SD_CR & SDIO_SD_CR_WORK2);
SDIO->SD_CR |= SDIO_SD_CR_DIRCMD |
SDIO_SD_CR_SBITCMD; /* Command TX & wait start bit */
SDIO->SD_CMDDR = ((idcmd | 0x40) & 0x0000007F) |
(arg>>16 & 0x0000FF00) |
(arg & 0x00FF0000) |
(arg<<16 & 0xFF000000); /* Set the command + argument */
SDIO->SD_CMDDR = arg & 0x000000FF; /* Set the argument */
SDIO->SD_CMDCRC = 0x00000000; /* Clear CRC */
SDIO->SD_CMDtransfer = 48; /* Command length */
SDIO->SD_CR |= SDIO_SD_CR_WORK2; /* Initiate command transaction */
```

35.1.2 Прием команды (по спецификации SDIO)

Для приема команды по интерфейсу SDIO следует настроить режим работы контроллера (регистр SD_CR), обнулить регистр контрольной суммы (SD_CMDCRC), задать количество бит которые надо принять по линии команд(регистр SD_CMDtransfer), а затем выставить бит WORK2 в «1». Формат принимаемой команды по спецификации SDIO представлен ниже (Рисунок 35-2).

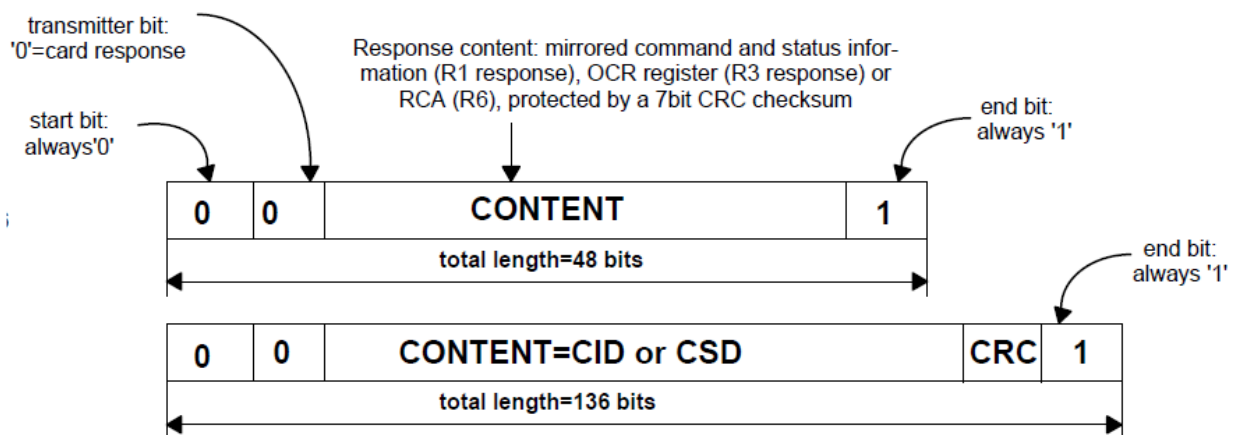


Рисунок 35-2 – Формат принимаемой команды по спецификации SDIO в режиме ведущего

Последовательность действий при приеме команды:

- включить контроллер SDIO (бит CR_SDE);
- если бит WORK2 установлен в «1», то следует либо дождаться окончания передачи прошлой команды, либо вручную сбросить этот бит;
- настроить направление передачи команды (бит DIRCMD) и установить бит ожидания стартового бита в линии CMD (бит SBITCMD);
- инициализировать нулевым значением регистр контрольной суммы линии команд;

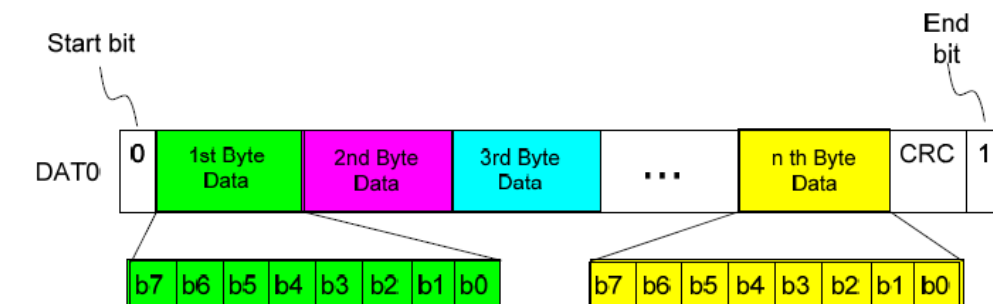
- задать длину ожидаемой команды, с учетом стартового бита, CRC и стопового бита (т.е. по спецификации SDIO 48 бит или 136 бит);
- установить бит WORK2 в «1».

Пример инициализации приема команды в соответствии со спецификацией SDIO:

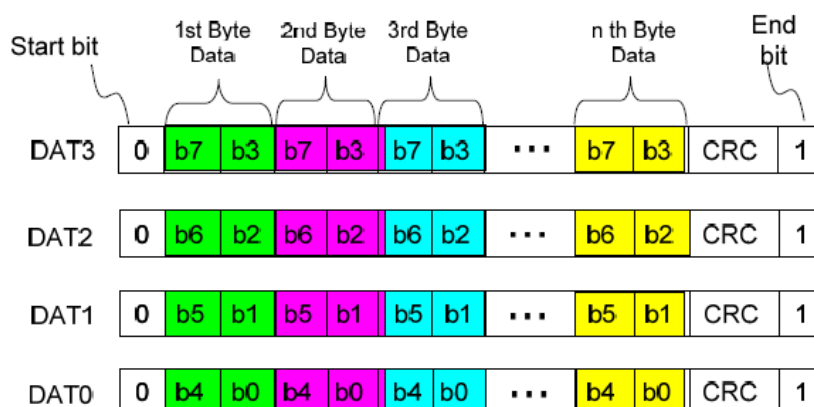
```
SDIO->SD_CR = SDIO_SD_CR_CLKOE |
SDIO_SD_CR_SDE; /* SDIO ON & Master */
do { /* Wait while CmdActive bit is set */
SDIO->SD_CR &= ~SDIO_SD_CR_WORK2; /* Reset Work2 */
} while (SDIO->SD_CR & SDIO_SD_CR_WORK2);
SDIO->SD_CR &= ~SDIO_SD_CR_DIRCMD; /* Command RX */
SDIO->SD_CR |= SDIO_SD_CR_SBITCMD; /* Set wait start bit */
SDIO->SD_CMDCRC = 0x00000000; /* Clear CRC */
SDIO->SD_CMDtransfer = 136; /* Command response length */
SDIO->SD_CR |= SDIO_SD_CR_WORK2; /* Initiate command transaction */
```

35.1.3 Передача данных (по спецификации SDIO)

Для передачи данных по интерфейсу SDIO следует настроить режим работы контроллера (регистр SD_CR), заполнить буфер данных (SD_DATDR), обнулить регистры контрольной суммы (SD_DAT0CRC- SD_DAT3CRC), задать количество бит которые надо передать по линиям данных (регистр SD_DATtransfer), а затем выставить бит WORK1 в «1». Формат передаваемой команды по спецификации SDIO представлен ниже (Рисунок 35-3).



Data Packet Format for Standard Bus (only DAT0 used)



Data Packet Format for Wide Bus (all four lines used)

Рисунок 35-3 – Формат данных по спецификации SDIO в режиме ведущего

Последовательность действий для передачи данных:

- включить контроллер SDIO (бит CR_SDE) и установить режим ведущего (бит CLKOE);
- если бит WORK1 установлен в «1», то следует либо дождаться окончания передачи прошлой команды, либо вручную сбросить этот бит;
- настроить направление передачи команды (бит DIRDAT), разрядность шины данных (бит WIDTHDAT) и установить бит формирования стартового бита в линиях DAT0-DAT3 (бит SBITDAT);
- заполнить буфер данных;
- инициализировать нулевым значением регистры контрольной суммы линий данных;
- задать длину передаваемых данных, с учетом CRC и стопового бита (без учета стартового бита);
- установить бит WORK1 в «1»;
- дождаться появления свободного места в FIFO передатчика (бит FIFODAT_FULL), после чего записать в буфер данных одно фиктивное слово с нулевым значением.

Пример инициализации передачи данных в соответствии со спецификацией SDIO:

```
SDIO->SD_CR = SDIO_SD_CR_CLKOE |
SDIO_SD_CR_SDE; /* SDIO ON & Master */
do { /* Wait while DATAActive bit is set */
SDIO->SD_CR &= ~SDIO_SD_CR_WORK; /* Reset Work1 */
} while (SDIO->SD_CR & SDIO_SD_CR_WORK);
SDIO->SD_CR |= SDIO_SD_CR_DIRDAT; /* DATA TX */
SDIO->SD_CR &= ~SDIO_SD_CR_WIDTHDAT; /* 4 bit width */
SDIO->SD_DATA0CRC = 0x00000000; /* Clear CRC DATA0*/
SDIO->SD_DATA1CRC = 0x00000000; /* Clear CRC DATA1*/
SDIO->SD_DATA2CRC = 0x00000000; /* Clear CRC DATA2*/
SDIO->SD_DATA3CRC = 0x00000000; /* Clear CRC DATA3*/
i = 0;
n_byte = 512;
while (i < n_byte)
{
SDIO->SD_DATDR = buff[i] | /*Fill the FIFO buffer*/
buff[i+1]<<8 |
buff[i+2]<<16 |
buff[i+3]<<24;
i = i + 4;
}
SDIO->SD_DATtransfer = n_byte * 8 + (16 + 1); /* Set length */
SDIO->SD_CR |= SDIO_SD_CR_SBITDAT; /* Set generate start bit */
SDIO->SD_CR |= SDIO_SD_CR_WORK; /* Initiate data transaction */
while(SDIO->SD_SR & SDIO_SD_SR_FIFODAT_FULL){}
SDIO->SD_DATDR = 0x00000000;
```

35.1.4 Прием данных (по спецификации SDIO)

Для приема данных по интерфейсу SDIO следует настроить режим работы контроллера (регистр SD_CR), обнулить регистры контрольной суммы (SD_DATA0CRC-SD_DATA3CRC), задать количество бит которые надо принять по линиям данных (регистр SD_DATtransfer), а затем выставить бит WORK1 в «1».

Последовательность действий для приема данных:

- включить контроллер SDIO (бит CR_SDE) и установить режим ведущего (бит CLKOE);
- если бит WORK1 установлен в «1», то следует либо дождаться окончания передачи прошлой команды, либо вручную сбросить этот бит;
- настроить направление передачи команды (бит DIRDAT), разрядность шины данных (бит WIDTHDAT) и установить бит ожидания стартового бита в линиях DAT0-DAT3 (бит SBITDAT);
- инициализировать нулевым значением регистры контрольной суммы линий данных;
- задать длину ожидаемых данных, с учетом CRC и стопового бита (без учета стартового бита)
- установить бит WORK1 в «1».

Примечание – Для корректного приема команды/данных в режиме ведущего необходимо обеспечить подтяжку линии CMD и DAT[3:0] к питанию.

Пример инициализации приема данных в соответствии со спецификацией SDIO:

```
SDIO->SD_CR = SDIO_SD_CR_CLKOE |
SDIO_SD_CR_SDE; /* SDIO ON & Master */
do { /* Wait while DATAActive bit is set */
SDIO->SD_CR &= ~SDIO_SD_CR_WORK; /* Reset Work1 */
} while (SDIO->SD_CR & SDIO_SD_CR_WORK);
SDIO->SD_CR &= ~SDIO_SD_CR_DIRDAT; /* DATA RX */
SDIO->SD_CR &= ~SDIO_SD_CR_WIDTHDAT; /* 4 bit width */
SDIO->SD_DATA0CRC = 0x00000000; /* Clear CRC DATA0*/
SDIO->SD_DATA1CRC = 0x00000000; /* Clear CRC DATA1*/
SDIO->SD_DATA2CRC = 0x00000000; /* Clear CRC DATA2*/
SDIO->SD_DATA3CRC = 0x00000000; /* Clear CRC DATA3*/
n_byte = 512;
SDIO->SD_DATtransfer = n_byte * 8 + (16 + 1); /* Set length */
SDIO->SD_CR |= SDIO_SD_CR_SBITDAT; /* Set wait start bit */
SDIO->SD_CR |= SDIO_SD_CR_WORK; /* Initiate data transaction */
```

35.2 Прерывания

В контроллере SDIO предусмотрено 6 маскируемых источников прерывания. В результате формируется один общий сигнал, представляющий собой комбинацию независимых сигналов, объединенных по схеме ИЛИ. Общий сигнал прерывания также является сигналом запроса DMA.

Прерывание может генерировать установка одного из шести флагов:

- RX буфер команд полон;
- RX буфер команд не пуст;
- TX буфер команд пуст;
- RX буфер данных полон;
- RX буфер данных не пуст;
- TX буфер данных пуст.

Каждый из независимых сигналов запроса прерывания может быть маскирован битами разрешения прерывания в регистре SD_CR.

35.3 Схемы включения контроллера

35.3.1 Подключение SD card к контроллеру

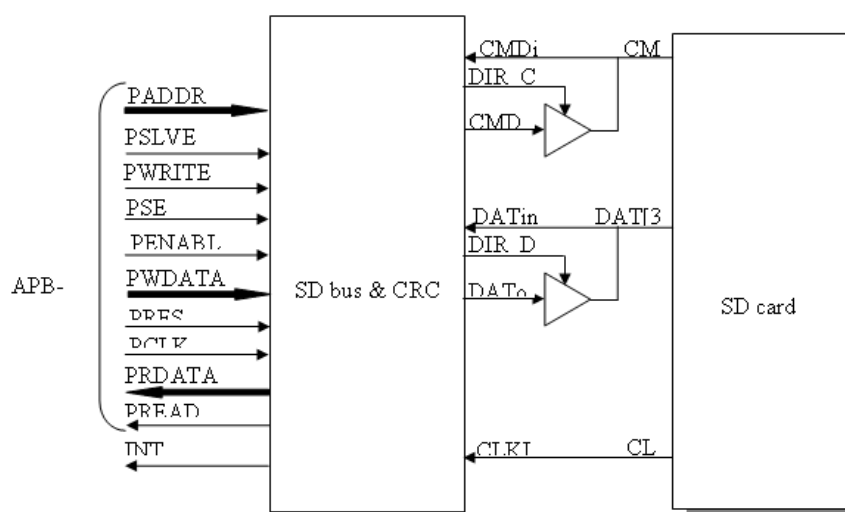


Рисунок 35-4 – Схема подключения хоста к контроллеру SD

35.4 Регистры SD

Таблица 35-1 – Описание регистров контроллера интерфейса SDIO

Базовый Адрес	Название	Описание
0x4004_8000	SDIO	Контроллер интерфейса SDIO
Смещение		
0x00	SD_CR	Регистр управления
0x04	SD_SR	Регистр состояния
0x08	SD_CMDDR	Регистр команд
0x0C	SD_DATDR	Регистр данных
0x10	SD_CMDCRC	Регистр контрольной суммы линии команд
0x14	SD_DAT0CRC	Регистр контрольной суммы линии данных 0
0x18	SD_DAT1CRC	Регистр контрольной суммы линии данных 1
0x1C	SD_DAT2CRC	Регистр контрольной суммы линии данных 2
0x20	SD_DAT3CRC	Регистр контрольной суммы линии данных 3
0x24	SD_CMDtransfer	Регистр-счетчик принимаемых/передаваемых бит линии команд
0x28	SD_DATtransfer	Регистр-счетчик принимаемых/передаваемых бит линий данных

35.4.1 Регистр управления

Таблица 35-2 – Регистр SD_CR

Номер	7	6	5	4	3	2	1	0
Доступ*	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0	0	0
	BR1	BR0	SBITDAT	SBITCMD	WORK	DIRDAT	DIRCMD	SDE

Номер	15	14	13	12	11	10	9	8
Доступ*	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0	0	0
	RXNEIE_ CMD	TXEIE_ CMD	RXFIE_ DAT	RXNEIE_ DAT	TXEIE_ DAT	CRCEN_ CMD	CRCEN_ DAT	BR2

Номер	31...22	21	20	19	18	17	16
Доступ*	U	R/W	R/W	R/W	R/W	R/W	R/W
Сброс		0	0	0	0	0	0
	-	CLKOE	WORK2	ENDBUSY	WRITECMD	WIDTHDAT	RXFIE_CMD

Обозначения здесь и далее по тексту:

R/W – бит доступен на чтение и запись;

RO – бит доступен только на чтение;

U – бит физически не реализован или зарезервирован

Таблица 35-3 – Описание бит регистра SD_CR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31..22		Зарезервировано
21	CLKOE	Сигнал разрешения выдачи сигнала CLK: 1 – ведущий; 0 – ведомый
20	WORK2	Бит начала транзакции команды: 1 – старт транзакции; 0 – останов транзакции. Бит автоматически сбрасывается в нуль, если SD_CMDtransfer равен нулю. Перед установкой этого бита необходимо занести в регистр счетчика передаваемых битов команды ненулевое значение. Если бит установлен, но FIFO для передачи команды пусто или FIFO приема команды полно, то транзакция не начнется до тех пор, пока FIFO не будут подготовлено для приема или передачи команды
19	ENDBUSY	Выставлять BUSY на линию DAT0: 1 – не выставлять BUSY; 0 – выставлять BUSY
18	WRITECMD	Посылать ответ 101b на команду записи:

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
		1 – посылать ответ; 0 – не посылать ответ
17	WIDTHDAT	Разрядность шины данных: 1 – шина одnorазрядная; 0 – шина четырехразрядная
16	RXFIE_CMD	Прерывание RX буфер команд полон разрешено: 1 – прерывание не маскировано. Это разрешает генерировать прерывание, если установлен флаг FIFOCMD_FULL; 0 – прерывание маскировано
15	RXNEIE_CMD	Прерывание RX буфер команд не пуст разрешено: 1 – прерывание не маскировано. Это разрешает генерировать прерывание, если сброшен флаг FIFOCMD_EMPTY; 0 – прерывание маскировано
14	TXEIE_CMD	Прерывание TX буфер команд пуст разрешено: 1 – прерывание не маскировано. Это разрешает генерировать прерывание, если установлен флаг FIFOCMD_EMPTY; 0 – прерывание маскировано
13	RXFIE_DAT	Прерывание RX буфер данных полон разрешено: 1 – прерывание не маскировано. Это разрешает генерировать прерывание, если установлен флаг FIFODAT_FULL; 0 – прерывание маскировано
12	RXNEIE_DAT	Прерывание RX буфер данных не пуст разрешено: 1 – прерывание не маскировано. Это разрешает генерировать прерывание, если сброшен флаг FIFODAT_EMPTY; 0 – прерывание маскировано
11	TXEIE_DAT	Прерывание TX буфер данных пуст разрешено: 1 – прерывание не маскировано. Это разрешает генерировать прерывание, если установлен флаг FIFODAT_EMPTY; 0 – прерывание маскировано
10	CRCEN_CMD	Аппаратное вычисление CRC по линии CMD: 1 – вычисление CRC разрешено; 0 – вычисление CRC запрещено. Этот бит не должен изменяться во время транзакции
9	CRCEN_DAT	Аппаратное вычисление CRC по линиям DAT3-DAT0: 1 – вычисление CRC разрешено; 0 – вычисление CRC запрещено. Этот бит не должен изменяться во время транзакции
8..6	BR2-BR0	Управление скоростью передачи: 001 – PCLK/4; 010 – PCLK/8; 011 – PCLK/16; 100 – PCLK/32; 101 – PCLK/64; 110 – PCLK/128; 111 – PCLK/256. Эти биты не должны изменяться во время транзакции

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
5	SBITDAT	Ожидание или формирование стартового бита в линиях DAT3-DAT0: 1 – ожидание или формирование в линиях DAT3-DAT0 стартового бита в зависимости от бита DIRDAT; 0 – ожидание или формирование стартового бита отключено. После установления сигнала WORK в единицу происходит ожидание (DIRDAT=0) или формирование (DIRDAT=1) стартового бита в линиях DAT3-DAT0. После этого бит автоматически сбрасывается в нуль
4	SBITCMD	Ожидание стартового бита в линии CMD: 1 – ожидание в линии CMD стартового бита; 0 – ожидание стартового бита отключено. После установления сигнала WORK в единицу происходит ожидание стартового бита в линии CMD. После этого бит автоматически сбрасывается в нуль. Передача стартового бита в линию CMD осуществляется записью команды определенного формата (48 или 136 бит) в регистр SD_CMDDR
3	WORK1	Бит начала транзакции данных: 1 – старт транзакции; 0 – останов транзакции. Бит автоматически сбрасывается в нуль, если SD_DATtransfer равен нулю. Перед установкой этого бита необходимо занести в регистр счетчика передаваемых битов данных ненулевое значение. Если бит установлен, но FIFO для передачи данных пусто или FIFO приема данных полно, то транзакция не начнется до тех пор, пока FIFO не будет подготовлено для приема или передачи данных
2	DIRDAT	Бит выбора направления линий DAT3-DAT0: 1 – линии работают на выход; 0 – линии работают на вход. Этот бит не должен изменяться во время транзакции
1	DIRCMD	Бит выбора направления линии CMD: 1 – линия работает на выход; 0 – линия работает на вход. Этот бит не должен изменяться во время транзакции
0	SDE	Разрешение работы SD bus: 1 – периферия включена; 0 – периферия отключена

35.4.2 Регистр состояния

Таблица 35-4 – Регистр SD_SR

Номер	31...4	3	2	1	0
Доступ*	U	R	R	R	R
Сброс		0	0	1	1
		FIFODAT_ FULL	FIFOCMD_ FULL	FIFODAT_ EMPTY	FIFOCMD_EMPTY

Таблица 35-5 – Описание бит регистра SD_SR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...4		Зарезервировано
3	FIFODAT_FULL	Состояние FIFO приема/передачи данных: 0 – FIFO не полно; 1 – FIFO полно
2	FIFOCMD_FULL	Состояние FIFO приема/передачи команд: 0 – FIFO не полно; 1 – FIFO полно
1	FIFODAT_EMPTY	Состояние FIFO приема/передачи данных: 0 – FIFO не пусто; 1 – FIFO пусто
0	FIFOCMD_EMPTY	Состояние FIFO приема/передачи команд: 0 – FIFO не пусто; 1 – FIFO пусто
Примечание – выбор буфера FIFO приема или передачи команд/данных, состояние которого отображается в регистре SD_SR, определяется битом DIRCMD/DIRDAT регистра SD_CR		

35.4.3 Регистр команд

Таблица 35-6 – Регистр SD_CMDDR

Номер	31...0
Доступ*	R/W
Сброс	0
	RX_FIFO/TX_FIFO

Таблица 35-7 – Описание бит регистра SD_CMDDR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31..0	RX_FIFO/TX_FIFO	8×32 бита FIFO приема и передачи

35.4.4 Регистр данных

Таблица 35-8 – Регистр SD_DATDR

Номер	31...0
Доступ*	R/W
Сброс	0
	RX_FIFO/TX_FIFO

Таблица 35-9 – Описание бит регистра SD_DATDR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31..0	RX_FIFO/TX_FIFO	128×32 бита FIFO приема и передачи

35.4.5 Регистр контрольной суммы линии команд

Таблица 35-10 – Регистр SD_CMDCRC

Номер	31..7	6..0
Доступ*	U	R/W
Сброс		0
		SD_CMDCRC

Таблица 35-11 – Описание бит регистра SD_CMDCRC

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31..7		Зарезервировано
6..0	SD_CMDCRC	Регистр CRC линии команд. Если разрешено вычисление контрольной суммы при приеме/передаче битом CRCEN_CMD, то регистр содержит подсчитанное значение CRC последовательно принятой/переданной команды. Перед началом подсчета регистр необходимо инициализировать нулевым значением. CRC рассчитывается последовательно с использованием полинома CRC7 (x^7+x^3+1). Чтение регистра в момент незавершенной транзакции приведет к ошибочному результату

35.4.6 Регистры контрольных сумм линий данных

Таблица 35-12 – Регистр SD_DATCRC

Номер	31..16	15..0
Доступ*	U	R/W
Сброс		0
		SD_DATCRC

Таблица 35-13 – Описание бит регистра SD_DATCRC

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31..16		Зарезервировано
15..0	SD_DAT0CRC SD_DAT1CRC SD_DAT2CRC SD_DAT3CRC	Регистры CRC линий данных. Если разрешено вычисление контрольной суммы при приеме/передаче битом CRC_DATEN, то регистр содержит подсчитанное значение CRC последовательно принятых/переданных данных. Перед началом подсчета регистр необходимо инициализировать нулевым значением. CRC рассчитывается последовательно с использованием полинома CRC16 ($x^{16}+x^{12}+x^5+1$). Чтение регистра в момент незавершенной транзакции приведет к ошибочному результату

35.4.7 Регистр-счетчик принимаемых/ передаваемых бит линии команд

Таблица 35-14 – Регистр SD_CMDtransfer

Номер	31..16	15..0
Доступ*	U	R/W
Сброс		0
		SD_CMDtransfer

Таблица 35-15 – Описание бит регистра SD_CMDtransfer

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31..16		Зарезервировано
15..0	SD_CMDtransfer	Содержит количество бит, которое необходимо передать/принять в линии команд. По мере приема/передачи счетчик уменьшается на единицу при каждом принятом/переданном бите, пока не достигнет нулевого значения. Максимальное количество доступных бит для приема/передачи 65536. Количество бит не обязательно кратно 32

35.4.8 Регистр-счетчик принимаемых/ передаваемых бит линий данных

Таблица 35-16 – Регистр SD_DATtransfer

Номер	31..16	15..0
Доступ*	U	R/W
Сброс		0
		SD_DATtransfer

Таблица 35-17 – Описание бит регистра SD_DATtransfer

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...16		Зарезервировано
15...0	SD_DATtransfer	Содержит количество бит, которое необходимо передать/принять в линиях данных. По мере приема/передачи счетчик уменьшается на четыре при каждом принятом/переданном полубайте, пока не достигнет нулевого значения. Поэтому значение счетчика должно быть кратно четырем. Максимальное количество доступных бит для приема/передачи 65536. Количество бит не обязательно кратно 32

36 Прерывания и исключения RISC

Состояние исключений:

Inactive – исключение не находится в стадии Active или Pending

Pending – исключение находится в состоянии ожидания обработки процессором.

Запрос прерывания от периферийных блоков или программы может изменить состояние соответствующего прерывания на состояние Pending

Active – исключение начало обрабатываться процессором, но еще не закончено. Обработчик исключения может быть прерван другим обработчиком исключения. В этом случае оба исключения находятся в состоянии Active

Active и **Pending** – исключение начало обрабатываться процессором, но появилось новое исключение в состоянии pending от того же источника.

36.1 Типы исключений

Исключения бывают следующих типов:

- RESET;
- NON MASKABLE INTERRUPT (NMI);
- Hard Fault;
- Memory Management fault;
- Bus Fault;
- Usage Fault;
- Supervisor Call (SVCALL);
- PendSV;
- SysTick.

36.1.1 RESET

RESET вызывается при включении питания и горячем сбросе. Модель исключений трактует RESET как специальную форму исключения. Когда выставляется RESET, работа процессора останавливается потенциально в любой точке инструкций. Когда RESET убирается, выполнение перезапускается с адреса, заданного в таблице векторов для сброса. Выполнение перезапускается с уровнем privileged в режиме thread.

36.1.2 NON MASKABLE INTERRUPT (NMI)

Немаскируемое прерывание (NMI) может быть вызвано периферией или установлено программой. Это самое высокоприоритетное исключение после сброса. Всегда разрешено и имеет фиксированный приоритет – 2.

Примечание – Данное прерывание не реализовано.

NMI не может быть:

- замаскировано или предотвращено от активации из другого исключения;
- прерывает любые исключения, кроме RESET.

36.1.3 Hard Fault

Исключение Hard Fault возникает при ошибке при обработке исключений или потому, что исключение не может быть обработано каким-либо другим механизмом. Hard fault имеет фиксированный приоритет – 1, означающий, что оно имеет больший приоритет, чем любое из исключений с конфигурируемым приоритетом.

36.1.4 Memory Management fault

Исключение Memory Management fault возникает при срабатывании по защите памяти. Блок MPU или фиксированные защитные настройки определяют это исключение, как для данных, так и для инструкций. Исключение используется для прерывания доступа за инструкцией в область EXECUTE NEVER (XN), если блок MPU не используется.

36.1.5 Bus Fault

Исключение возникает при ошибке памяти при выполнении выборки инструкций или обращения за данными. Это может быть при возникновении ошибки на шинах доступа к памяти, например, обращение в несуществующую память.

36.1.6 Usage Fault

Исключение USAGE FAULT возникает при сбоях при выполнении инструкции.

Например:

- выполнение неизвестной инструкции;
- обращение к некорректно выровненным данным;
- некорректное состояние при выполнении инструкции;
- ошибка при возвращении из обработчика.

Данное исключение может быть сконфигурировано и используется для обработки следующих ситуаций:

- невыровненный адрес при обращении за полусловами halfword и словами word;
- деление на ноль.

36.1.7 SVCcall

Исключение Supervisor Call (SVCALL) возникает при выполнении инструкции SVC. В приложениях с использованием Операционных Сред инструкция SVC может использоваться для доступа к функциям ОС и драйверам устройств.

36.1.8 PendSV

Исключение PendSV является прерыванием запросом сервисов системного уровня. В приложениях с использованием ОС PendSV используется для переключения контекстов, когда нет других активных исключений.

36.1.9 SysTick

Исключение SysTick генерируется системным таймером, когда он обнуляется. Программное обеспечение также может генерировать исключение SysTick. В приложениях с использованием ОС процессор может использовать это исключение для подсчета системных циклов.

36.2 Прерывания (IRQ)

Прерывания или IRQ – это исключения, вызываемые периферийными устройствами или программными запросами. Все прерывания асинхронны по отношению к выполняемым инструкциям. В системе прерывания используются для коммуникации периферии и процессора

Таблица 36-1 – Различные типы исключений

Номер исключения	Номер IRQ	Тип	Приоритет	Адрес вектора обработчика (смещение)	Активация
1	-	RESET	-3, наивысший	0x0000_0004	Асинхронный
2	-14	NMI	-2	0x0000_0008	Асинхронный
3	-13	Hard Fault	-1	0x0000_000C	-
4	-12	Memory Management Fault	Конфигурируемый	0x0000_0010	Синхронный
5	-11	Bus Fault	Конфигурируемый	0x0000_0014	Синхронный/ Асинхронный
6	-10	Usage Fault	Конфигурируемый	0x0000_0018	Синхронный
7-10	-	-	-	Зарезервировано	-
11	-5	SVCall	Конфигурируемый	0x0000_002C	Синхронный
12-13	-	-	-	Зарезервировано	-
14	-2	PendSV	Конфигурируемый	0x0000_0038	Асинхронный
15	-1	SysTick	Конфигурируемый	0x0000_003C	Асинхронный
16 и выше	0 и выше	IRQ	Конфигурируемый	0x0000_0040 и выше	Асинхронный

Для асинхронных исключений, кроме RESET, процессор может выполнить другие инструкции между возникновением сигнала исключения и входом в обработчик.

Программа в Privileged режиме может запретить прерывания, имеющие конфигурируемый приоритет.

36.3 Обработчики исключений

Для обработки исключений используются:

– Процедуры обработки прерываний (Interrupt Service Routines – ISRs)

Прерывания с IRQ0 по IRQ31 обрабатываются процедурами ISR.

Если программный код, сбрасывающий флаг прерывания на уровне периферийного блока, находится в конце функции обработчика прерывания, ядро может не успеть выполнить очистку флага в момент выхода из обработчика, из-за чего прерывание будет вызвано повторно. Чтобы этого избежать, необходимо после сброса флага прерывания сделать фиктивное чтение, например, регистра статусных флагов периферийного блока в отдельную переменную.

– **Обработчики ошибок (Fault Handlers)**

Обрабатывают исключения Hard fault, memory management fault, usage fault и bus fault.

– **Системные обработчики (System handlers)**

Обрабатывают исключения NMI, PendSV, SVCcall и SysTick.

36.4 Таблица векторов

Таблица векторов содержит указатель стека, вектор входа по RESET и стартовые адреса обработчиков, также называемых векторами. Ниже представлена последовательность векторов в таблице (Рисунок 36-1). Младший бит всех векторов должен быть равен 1, указывая на то, что обработчик выполняется в режиме Thumb.

Exception number	IRQ number	Offset	Vector
45	29	0x00B4	IRQ29
.	.	.	.
.	.	.	.
.	.	.	.
18	2	0x004C	IRQ2
17	1	0x0048	IRQ1
16	0	0x0044	IRQ0
15	-1	0x0040	Systick
14	-2	0x003C	PendSV
13	.	0x0038	Reserved
12	.	.	Reserved for Debug
11	-5	0x002C	SVCcall
10	.	.	Reserved
9	.	.	
8	.	.	
7	.	.	
6	-10	0x0018	Usage fault
5	-11	0x0014	Bus fault
4	-12	0x0010	Memory management fault
3	-13	0x000C	Hard fault
2	-14	0x0008	Reserved
1	.	0x0004	Reset
.	.	0x0000	Initial SP value

Рисунок 36-1 – Таблица векторов исключений и прерываний

При системном сбросе таблица векторов располагается по фиксированному адресу 0x00000000. Программное обеспечение в privileged режиме может перенести таблицу в другое место памяти через регистр VTOR. Таблица может располагаться в адресах от 0x00000080 до 0x3ffff80. Подробнее в описании регистра VTOR.

36.5 Приоритеты исключений

- Более малое значение приоритета означает больший приоритет.
- Конфигурируемы все приоритеты, кроме RESET и Hard Fault.
- Если программное обеспечение не задает приоритетов, то все они имеют приоритет 0.

– Конфигурируемый приоритет может быть в диапазоне от 0 до 15. Это означает что RESET, Hard Fault и NMI, имеющие отрицательное значение приоритета, всегда имеют больший приоритет.

– Если имеется несколько исключений с одинаковым приоритетом, то больший приоритет имеет исключение с меньшим порядковым номером.

– Если процессор выполняет обработчик исключения и происходит исключение с большим приоритетом, то происходит переход на обработчик исключения с большим приоритетом.

– Если при выполнении обработчика произошло исключение с таким же приоритетом, то это исключение будет выполнено по завершению текущего обработчика, несмотря на порядковый номер исключения.

36.5.1 Группировка приоритетов прерываний

Для увеличения управляемости приоритетов в системах с прерываниями контроллер прерываний NVIC поддерживает группировку приоритетов. Это достигается за счет разбиения регистра приоритета прерывания на две части:

- верхняя часть определяет группу приоритетов;
- нижняя часть задает подприоритет в группе.

Только приоритет группы определяет последовательность обработки прерываний. Когда процессор выполняет обработку прерывания, другое прерывание с таким же приоритетом группы не прервет обработку первоначального обработчика. При возникновении нескольких прерываний, имеющих одинаковый приоритет группы, подприоритеты определяют последовательность их обработки. При возникновении нескольких прерываний с одинаковым приоритетом группы и подприоритетом первым обрабатывается прерывание с меньшим номером.

36.6 Вход в обработчик и выход из обработчика

36.6.1 Приоритетное прерывание

Выполнение процессором процедуры обработки исключительной ситуации (далее по тексту – исключения) может быть прервано в случае возникновения исключения с приоритетом выше, чем у обрабатываемого. Подробнее данный вопрос рассмотрен в пункте 36.5.1 «Группировка приоритетов прерываний». В случае, если внутри обработчика исключения возникает прерывание более высокого приоритета, возникает ситуация, называемая вложенным исключением. Подробнее данный вопрос рассмотрен в пункте 36.6.5 «Вход в процедуру обработки исключения».

36.6.2 Возврат

Возврат из программы-обработчика осуществляется по завершении обработки исключительной ситуации, с одновременным выполнением следующих условий:

- в системе отсутствуют необработанные исключения с достаточным приоритетом;
- завершённый обработчик не обрабатывал запоздавшее исключение (late-arriving exception).

Процессор обращается к стеку и восстанавливает состояние, имевшее место до вызова обработчика. Более подробная информация дана в пункте 36.6.6 «Возврат из обработчика исключения».

36.6.3 Передача управления без восстановления контекста (tail-chaining)

Данный механизм ускоряет процесс обработки исключений. По завершении выполнения обработчика осуществляется проверка наличия необработанных исключений и в случае, если исключения, требующие вызова обработчика, присутствуют, восстановление состояния процессора из стека не производится, а управление передается непосредственно на новый обработчик.

36.6.4 Запоздавшее исключение (late-arriving exception)

В случае, если во время сохранения состояния при входе в обработчик возникла исключительная ситуация с более высоким приоритетом, процессор передает управление непосредственно высокоприоритетному обработчику.

Подобный способ обработки высокоприоритетного исключения возможен до момента начала выполнения первой инструкции процедуры обработки исключительной ситуации. После возврата из обработчика запоздавшего исключения осуществляется передача управления на прерванный низкоприоритетный обработчик без восстановления контекста.

36.6.5 Вход в процедуру обработки исключения

Вызов процедуры обработки исключения происходит в случае наличия необработанных исключительных ситуаций с достаточным приоритетом и при выполнении одного из следующих условий:

- процессор находится в режиме приложения (thread mode);
- новая исключительная ситуация имеет приоритет выше, чем обрабатываемая в текущий момент времени, что приводит к приоритетному прерыванию выполнения текущего обработчика. В этом случае возникает вложение одного исключения в другое.

При необходимости вызова обработчика, за исключением случаев обработки запоздавшего исключения и передачи управления на обработчик без восстановления контекста, процессор заносит в текущий стек восемь слов данных, называемые далее стековым фреймом. Этот фрейм включает в себя следующие значения:

- регистры R0-R3, R12;
- адрес возврата;
- регистр PSR;
- регистр LR.

Указанная операция далее будет называться сохранением контекста. Непосредственно после ее выполнения указатель стека равен младшему адресу стекового фрейма.

В случае, если бит STKALIGN в регистре управления конфигурацией (CCR) установлен в 1, во время сохранения контекста производится выравнивание адреса стека по границе двойного слова.

Стековый фрейм содержит адрес возврата, указывающий на ближайшую невыполненную инструкцию прерванной программы. По завершении процедуры обработки исключения значение адреса возврата заносится в счетчик команд, после чего выполнение программы возобновляется с прерванной точки.

Одновременно с сохранением контекста процессор осуществляет выборку адреса точки входа в процедуру обработки исключения из таблицы векторов исключений. По завершении операции сохранения контекста процессор передает управление на полученный из таблицы адрес.

Одновременно в регистр LR записывается значение EXC_RETURN, позволяющее определить, какой из двух указателей стека соответствует данному стековому фрейму, и в каком режиме находился процессор перед входом в обработчик.

Если во время передачи управления не возникло исключения с более высоким приоритетом, процессор начинает выполнение вызванной процедуры обработки и автоматически изменяет состояние текущего прерывания с ожидающего обработки на активное.

В противном случае процессор передает управление обработчику высокоприоритетной исключительной ситуации без изменения состояния отложенного прерывания в соответствии с правилами, изложенными в пункте 36.6.4 «Запоздавшее исключение (late-arriving exception)».

36.6.6 Возврат из обработчика исключения

Возврат из обработчика исключения осуществляется в случае, если процессор находится в режиме обработчика (handler mode) и выполняет одну из следующих инструкций, позволяющих загрузить значение EXC_RETURN в регистр PC:

- инструкцию POP с аргументом PC;
- инструкцию BX с любым регистром;
- инструкции LDR или LDM с регистром PC в качестве приемника.

Значение EXC_RETURN загружается в регистр LR по входу в обработчик исключения. Механизм обработки исключений использует это значение для того, чтобы определить, завершил ли процессор выполнение процедуры обработки исключительной ситуации. Младшие четыре бита EXC_RETURN содержат информацию о состоянии стека и режиме работы процессора. Информация о назначении разрядов EXC_RETURN[3:0] и особенности процесса возврата из обработчика исключения представлены в следующей таблице (Таблица 36-1).

Процессор устанавливает биты EXC_RETURN [31:4] в 0xFFFFFFFF. Загрузка данного значения в PC указывает на завершение процедуры обработки исключения и заставляет процессор выполнить необходимые действия для возврата из обработчика.

Таблица 36-2 – Возврат из обработчика исключения

EXC_RETURN[3:0]	Описание
bXXX0	Зарезервирован

EXC_RETURN[3:0]	Описание
b0001	Возврат в режим обработчика. Восстановление контекста осуществляется из стека MSP. Дальнейшая работа осуществляется со стеком MSP
b0011	Зарезервирован
b01X1	Зарезервирован
b1001	Возврат в режим приложения. Восстановление контекста осуществляется из стека MSP. Дальнейшая работа осуществляется со стеком MSP
b1101	Возврат в режим приложения. Восстановление контекста осуществляется из стека PSP. Дальнейшая работа осуществляется со стеком PSP
b1X11	Зарезервирован

36.7 Обработка отказов

Отказы являются частным случаем исключений. Отказы могут возникать по следующим причинам:

- ошибка шины в ходе:
- чтения инструкции или вектора обработчика;
- доступа к данным.
- ошибка, обнаруженная процессором, например, неопределенная инструкция или попытка изменить состояние процессора с помощью команды BX;
- попытка выполнить инструкцию, расположенную в области памяти, помеченной как неисполняемая (Non-Executable – XN);
- отказ блока защиты памяти MPU вследствие нарушения прав доступа или вследствие попытки доступа к неподдерживаемой области адресного пространства.

36.7.1 Типы отказов

Таблица 36-3 демонстрирует типы отказов, обработчики, вызываемые при их возникновении, соответствующие данному типу отказа регистры состояния, и биты регистра, указывающие на конкретный отказ. Более подробная информация представлена в пункте 38.1.10 «SCB->CFSR».

Таблица 36-3 – Отказы

Отказ	Обработчик	Наименование бита регистра	Регистр отказа
Ошибка доступа к шине при чтении вектора	Аппаратный отказ	VECTTBL	«Регистр состояния аппаратных отказов»
Эскалация отказа		FORCED	
Ошибка доступа к памяти:	Отказ доступа к памяти	-	«Регистр состояния отказов доступа к памяти», «Регистр адреса отказа доступа к памяти»
– при чтении команды		IACCVIOL	
– при доступе к данным		DACCVIOL	
– при сохранении контекста		MSTKERR	
– при восстановлении контекста		MUNSKERR	

Отказ	Обработчик	Наименование бита регистра	Регистр отказа
Ошибка шины:	Отказ доступа к шине	-	«Регистр состояния отказа доступа к шине», «Регистр адреса отказа доступа к шине»
– при сохранении контекста		STKERR	
– при восстановлении контекста		UNSTKERR	
– при загрузке инструкции		IBUSERR	
локализованная ошибка шины данных		PRECISERR	
нелокализованная ошибка шины данных		IMPRECISERR	
Попытка доступа к сопроцессору	Отказ, вызванный ошибками программирования	NOCP	«Регистр состояния отказов, вызванных ошибками программирования»
Неизвестная инструкция		UNDEFINSTR	
Попытка выбора неверного набора инструкций*		INVSTATE	
Неверное значение EXC_RETURN		INVPC	
Запись или чтение по неверно выровненному адресу		UNALIGNED	
Деление на 0		DIVBYZERO	
* Попытка выбора набора инструкций, неподдерживаемых процессором			

36.7.2 Эскалация отказов и аппаратные отказы

Всем типам исключительных ситуаций по отказу, за исключением аппаратных отказов (hard fault) можно задать приоритет обработки, см. пункт 38.1.8 «SCB->SHP[x]». Выполнение данных обработчиков можно программно запретить, см. пункт 38.1.9 «SCB->SHCSR».

Как правило, приоритет обработки исключения, наряду со значениями регистров маскирования исключений, определяет, будет ли вызываться данный обработчик отказа, а также – сможет ли он прервать выполнение другого обработчика.

В некоторых ситуациях отказ с конфигурируемым уровнем приоритета рассматривается системой как аппаратный. Такая ситуация именуется эскалацией отказа (escalation). Это возможно в следующих случаях:

- обработчик отказа во время своего выполнения вызвал отказ того же типа. Этот тип эскалации обусловлен тем фактом, что обработчик не может прервать собственное выполнение, так как его приоритет равен текущему;
- обработчик отказа вызвал отказ другого типа с приоритетом, меньшим или равным собственному. В этом случае новый обработчик также не может быть активизирован вследствие недостаточного уровня приоритета;
- обработчик исключительной ситуации вызвал отказ с приоритетом обработки, меньшим или равным текущему;
- возник отказ, обработчик которого не разрешен.

Если отказ обращения к шине возник во время загрузки данных в стек при передаче управления на обработчик отказа доступа к шине – эскалации не происходит. Таким образом, в случае, если отказ возник вследствие разрушения стека, передача

управления на обработчик отказа выполняется, несмотря на то, что сохранение контекста не было осуществлено.

Обработка аппаратных отказов имеет фиксированный приоритет. Она может быть прервана только по сигналу сброса Reset или немаскируемого прерывания NMI. Сам обработчик способен прерывать обработку любых исключительных ситуаций, кроме ситуаций сброса Reset, NMI, а также другого аппаратного отказа.

36.7.3 Регистры состояния и адреса отказа

Регистры состояния отказа содержат информацию о причине отказа. Для обработки отказов шины и доступа к памяти предусмотрены регистры адреса отказа, содержащие адрес, по которому произошло обращение, вызвавшее отказ. Подробная информация приведена в таблице ниже (Таблица 36-4).

Таблица 36-4 – Регистры состояния и адреса отказа

Обработчик	Регистр состояния	Регистр адреса	Описание регистров
Аппаратный отказ	HFSR	-	«Регистр состояния аппаратных отказов»
Отказ доступа к памяти	MMFSR	MMFAR	«Регистр состояния отказов доступа к памяти» «Регистр адреса отказа доступа к памяти»
Отказ доступа к шине	BFSR	BFAR	«Регистр состояния отказов доступа к шине» «Регистр адреса отказа доступа к шине»
Отказ, вызванный ошибками программирования	UFSR	-	«Регистр состояния отказов, вызванных ошибками программирования»

36.7.4 Блокировка

Процессор переходит в состояние блокировки в случае, если аппаратный отказ возник во время выполнения программы-обработчика аппаратного отказа.

После перехода в состояние блокировки процессор перестает выполнять какие-либо команды. В этом состоянии он будет находиться до момента сброса.

36.8 Управление электропитанием

В процессоре RISC предусмотрены следующие режимы ожидания (пониженного энергопотребления):

- Deep Sleep;
- Sleep;
- Standby.

Выбор процессором конкретного режима ожидания определяется значением бита SLEEPDEEP регистра SCR (см. «Регистр управления системой»).

Далее в разделе описаны механизмы перехода в режим пониженного энергопотребления и условия выхода из этого режима.

36.8.1 Переход в режим пониженного энергопотребления

Система может формировать ложные сигналы событий, выводящие процессор из ожидания. Например, эти сигналы возникают при работе отладчика. Следовательно, программное обеспечение должно быть способным перевести процессор обратно в указанный режим ожидания. Для этого можно, например, организовать в программе пустой цикл.

36.8.2 Ожидание прерывания

Инструкция ожидания прерывания WFI (wait for interrupt) после своего выполнения немедленно переводит процессор в режим пониженного энергопотребления.

36.8.3 Ожидание события

Инструкция ожидания сигнала события WFE (wait for event) переводит или не переводит процессор в режим пониженного энергопотребления в зависимости от результата проверки одноразрядного регистра события. При этом процессор проверяет значение регистра события, и в случае, если он равен 0, приостанавливает дальнейшее выполнение команд и переходит в состояние ожидания. В случае, если он равен 1, процессор записывает в регистр события 0 и продолжает нормальную работы без перехода в режим ожидания.

36.8.4 Переход в режим ожидания по выходу из обработчика исключения (режим Sleep)

В случае, если бит SLEEPONEXIT регистра SCR установлен в 1, по завершении выполнения обработчика исключения процессор возвращается в режим приложения, после чего немедленно переходит в состояние пониженного энергопотребления.

Данный механизм рекомендуется использовать в задачах, в которых процессор используется только для обработки исключений.

36.8.5 Выход из состояния ожидания

Условия выхода процессора из режима ожидания зависят от причины, по которой он был переведен в этот режим.

36.8.5.1 Выход из ожидания по команде WFI и в режиме Sleep

Как правило, процессор выходит из режима ожидания только в случае возникновения исключительной ситуации с приоритетом, достаточным для активизации соответствующего обработчика.

В некоторых приложениях может возникнуть необходимость выполнения процедур восстановления системы после выхода процессора из режима пониженного энергопотребления, однако до того, как он начнет выполнять обслуживание прерываний. Для того, чтобы добиться этого, достаточно установить бит PRIMASK в «1», а бит FAULTMASK – в «0». В случае возникновения в системе разрешенного прерывания с приоритетом выше текущего приоритета, процессор будет выведен из ожидания, однако не сможет передать управление обработчику прерывания до тех пор, пока бит PRIMASK не будет установлен в «0».

36.8.5.2 Выход из ожидания по команде WFE

Процессор выходит из режима ожидания в случае обнаружения исключительной ситуации с приоритетом, достаточным для активизации обработчика.

Кроме того, в случае установки бита SEVONPEND регистра SCR в 1, любое новое необслуженное прерывание формирует сигнал события и выводит процессор из ожидания, даже если это прерывание запрещено или имеет приоритет, недостаточно высокий для запуска обработчика.

Более подробная информация о регистре SCR представлена в разделе «Регистр управления системой».

36.8.6 Рекомендации по программированию режима энергопотребления

В стандарте ANSI языка C отсутствует возможность непосредственной генерации инструкций WFI и WFE. В CMSIS предусмотрены встроенные функции, предназначенные для включения в код этих инструкций:

```
void __WFE(void) // Wait for Event
void __WFI(void) // Wait for Interrupt
```

Периферийные блоки формируют прерывания с IRQ0 до IRQ31

Таблица 36-5 – Формирование прерывания с IRQ0 до IRQ31

Прерывания	Смещение	Блок	Принцип формирования
IRQ0	0x0040	SSP3	Сигнал SSPINTR
IRQ1	0x0044	SSP4	Сигнал SSPINTR
IRQ2	0x0048	USB	Прерывания от USB Host при наличии соответствующих флагов разрешения . HostSOFSent или HostConnEvent или HostResume или HostTransDone. Прерывания от USB Slave при наличии соответствующих флагов разрешения SlaveNAKSent или SlaveSOFRXed или SlaveResetEvent или SlaveResume или SlaveTransDone
IRQ3	0x004C	McBSP1(DSP)	Прерывания от McBSP (DSP)
IRQ4	0x0050	McBSP2(DSP)	Прерывания от McBSP (DSP)
IRQ5	0x0054	DMA	Прерывания от DMA dma_err или dma_done. Обработка прерываний от DMA в соответствии с разделом Error signaling технического описания DMA
IRQ6	0x0058	UART1	Сигнал UARTINTR
IRQ7	0x005C	UART2	Сигнал UARTINTR
IRQ8	0x0060	SSP1	Сигнал SSPINTR
IRQ9	0x0064	McBSP3(DSP)	Прерывания от McBSP (DSP)
IRQ10	0x0068	I2C	Сигнал INT при EN_INT
IRQ11	0x006C	POWER	Сигнал прерывания от POWER Detecor

Прерывания	Смещение	Блок	Принцип формирования
IRQ12	0x0070	WWDG	Сигнал прерывания от WWDG
IRQ13	0x0074	DMA(DSP)	Прерывания DMA (DSP). Возникает в случае завершения обработки полного цикла одного из каналов.
IRQ14	0x0078	Timer 1	Сигнал прерывания от Таймера. TIM_STATUS и TIM_IE
IRQ15	0x007C	Timer 2	Аналогично
IRQ16	0x0080	Timer 3	Аналогично
IRQ17	0x0084	ADC	Сигналы прерываний от АЦП. EOCIF_1 или AWOIF_1 или EOCIF_2 или AWOIF_2
IRQ18	0x0088	SDIO	Прерывание от контроллера SDIO
IRQ19	0x008C	COMPARATOR	Сигнал Rslt_Sy1
IRQ20	0x0090	SSP2	Сигнал SSPINTR
IRQ21	0x0094	AudioCodec(DSP)	Прерывания от модуля Аудиокодека (DSP)
IRQ22	0x0098	Crypto(DSP)	Прерывания от Криптомодуля (DSP)
IRQ23	0x009C	Timer(DSP)	Прерывания от Таймера (DSP)
IRQ24	0x00A0	DSP Core	Программные прерывания от DSP (регистр AIRQ)
IRQ25	0x00A4	DSP State	Прерывания, возникающие при переходе DSP в одно из состояний IDLE (Поле IDLE регистра DSP Control не равно 0).
IRQ26	0x00A8	UART3	Сигнал UARTINTR
IRQ27	0x00AC	BACKUP	Прерывание от ВКР и часов реального времени
IRQ28	0x00B0	Внешнее прерывание 1	Сигнал EXT_INT1. Вывод PA[15] в переопределенном режиме и PD[15] в основном режиме
IRQ29	0x00B4	Внешнее прерывание 2	Сигнал EXT_INT2. Вывод PB[11] в переопределенном режиме
IRQ30	0x00B8	Внешнее прерывание 3	Сигнал EXT_INT3. Вывод PE[6] в переопределенном режиме
IRQ31	0x00BC	Внешнее прерывание 4	Сигнал EXT_INT4. Вывод PF[11] в переопределенном режиме

37 Контроллер прерываний NVIC (RISC)

В разделе описан векторный контроллер прерываний с возможностью вложения (NVIC – Nested Vectored Interrupt Controller) и используемые им регистры.

Контроллер обеспечивает следующие возможности:

- программное задание уровня приоритета в диапазоне от 0 до 15 независимо каждому прерыванию. Более высокое значение уровня соответствует меньшему приоритету, таким образом, уровень 0 отвечает наивысшему приоритету прерывания;
- срабатывание сигнала прерывания по импульсу и по уровню;
- динамическое изменение приоритета прерываний;
- разделение исключений по группам с одинаковым приоритетом и по подгруппам внутри одной группы;
- передача управления из одного обработчика исключения в другой без восстановления контекста.

Процессор автоматически сохраняет в стеке свое состояние (контекст) по входу в обработчик прерывания и восстанавливает его по завершению обработчика, без необходимости непосредственного программирования этих операций. Это обеспечивает обработку исключительных ситуаций с малой задержкой.

Назначение регистров контроллера прерываний представлено ниже (Таблица 37-1).

Таблица 37-1 – Обобщенная информация о регистрах контроллера NVIC

Адрес	Название	Тип	Доступ	Значение после сброса	Описание
0xE000E100	NVIC				Контроллер прерываний NVIC
0x000	ISER[0]	RW	Привилегированный	0x00000000	Регистр разрешения прерываний ISER
...					
0x01C	ISER[7]				
...					
0x080	ICER[0]	RW	Привилегированный	0x00000000	Регистр запрета прерываний ICER
...					
0x09C	ICER[7]				
...					
0x100	ISPR[0]	RW	Привилегированный	0x00000000	Регистр установки состояния ожидания для прерывания ISPR
...					
0x11C	ISPR[7]				
...					
0x180	ICPR[0]	RW	Привилегированный	0x00000000	Регистр сброса состояния ожидания для прерывания ICPR
...					
0x19C	ICPR[7]				
...					
0x200	IABR[0]	RO	Привилегированный	0x00000000	Регистр активных прерываний IABR
...					
0x21C	IABR[7]				

Адрес	Название	Тип	Доступ	Значение после сброса	Описание
...					
0x300	IP[3],IP[2],IP[1],IP[0]	RW	Привилегированный	0x00000000	Регистр приоритета прерываний IP
...					
0x3F0	IP[239], IP[238], IP[237], IP[236]				
...					
0xE00	STIR	WO	В зависимости от конфигурации*	0x00000000	Регистр программного формирования прерываний STIR
* Более подробную информацию см. в описании регистра					

37.1 Логика работы прерываний контроллера NVIC.

В данном разделе описывается функционирование контроллера NVIC при поступлении на его вход запросов прерываний IRQ от различных модулей периферии микросхемы.

Первоначальным условием работы прерывания является его разрешение в модуле NVIC. За это отвечают регистры:

ISER – за разрешение прерываний,

ICER – за запрет прерываний.

В случае, когда соответствующий запрос разрешен (при данном условии рассмотрены все диаграммы в разделе), и приходит сигнал активации прерывания – запрос IRQ request, то возникает признак отложенного прерывания IRQ pending. Данный признак переводит прерывание в состояние ожидания его обработки ядром.

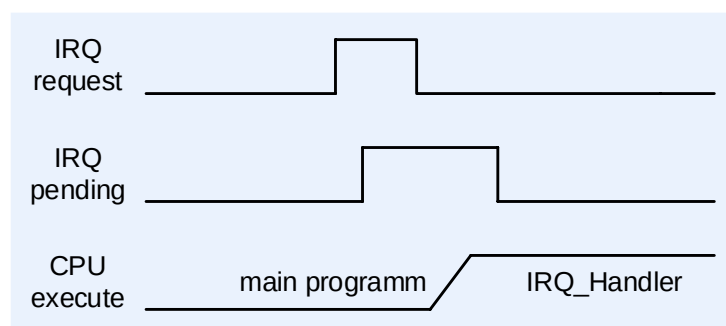


Рисунок 37-1 – Выставление отложенного запроса на прерывание и последующая его обработка

Pending биты выставляются в регистрах ISPR/ICPR, которые в свою очередь позволяют программно управлять признаком отложенного прерывания. ISPR – для установки pending бит, ICPR – для сброса соответственно. Если после прихода запроса на прерывание IRQ request, сбросить pending бит в регистре ICPR до того, как ядро приступит к его обработке, то прерывание будет проигнорировано (Рисунок 37-2).

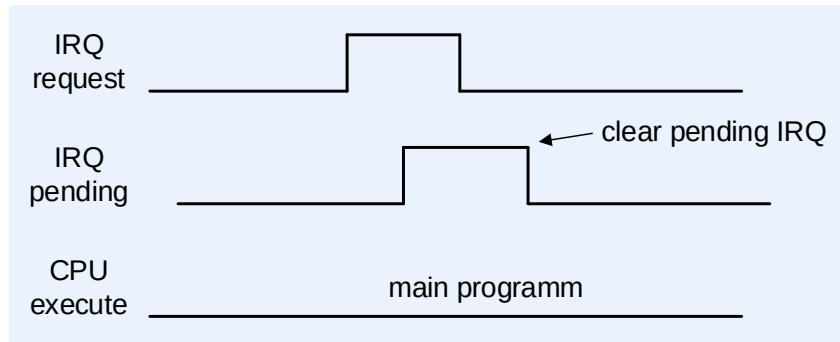


Рисунок 37-2 – Сброс признака отложенного прерывания, до обработки ядром

Если произойдет снятие запроса IRQ request от источника, «защелкивание» признака отложенного прерывания гарантирует отработку его ядром в соответствии с приоритетом (Рисунок 37-3). Сам IRQ pending признак снимается автоматически, когда прерывание становится активным, о чем сигнализирует признак IRQ active. Информация об активности соответствующего прерывания содержится в регистре IABR[x].

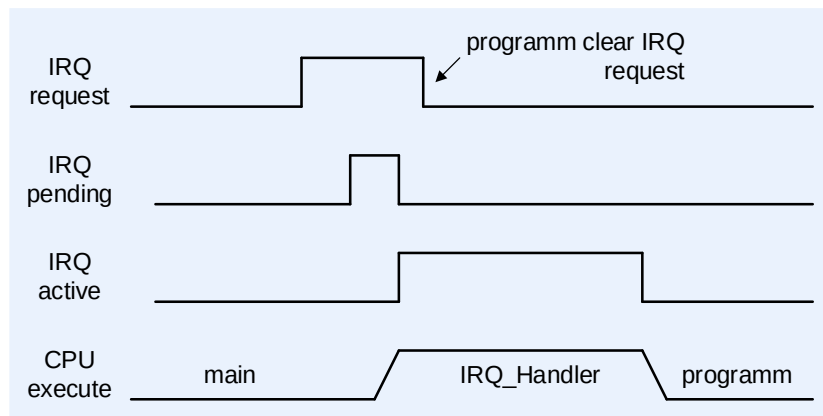


Рисунок 37-3 – Сброс признака отложенного прерывания, до обработки ядром

После того как прерывание стало активным, повторно запустить обработчик того же прерывания будет невозможно до тех пор, пока не будет завершена процедура обработки прерывания командой выхода из исключения. После выполнения команды выхода происходит сброс признака активности IRQ active.

При удержании источником на входе NVIC запроса на обработку IRQ request, по окончании обработки прерывания и снятия признака активного прерывания IRQ active, происходит повторное выставление признака отложенного прерывания IRQ pending – «защелкивание» pending бита, сброс которого в дальнейшем инициирует повторную активность и обработку того же исключения (Рисунок 37-4).

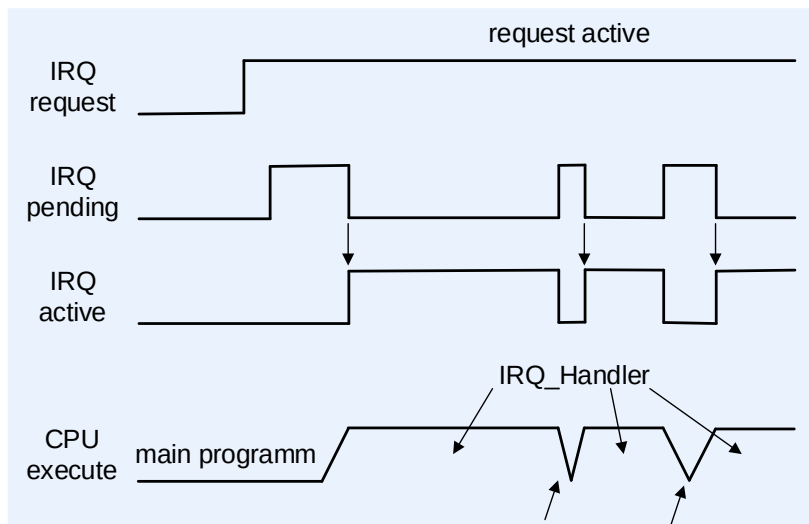


Рисунок 37-4 – Повторная обработка прерываний при удержании запроса от источника

Необходимо учитывать, что если источник прерываний выдает многократную установку и снятие запроса IRQ request на входе контроллера NVIC, то в таком случае только первый запрос выставляет признак отложенного прерывания IRQ pending, а остальные запросы до начала процедуры обработки прерывания (в момент активного признака отложенного прерывания) будут проигнорированы ядром (Рисунок 37-5).

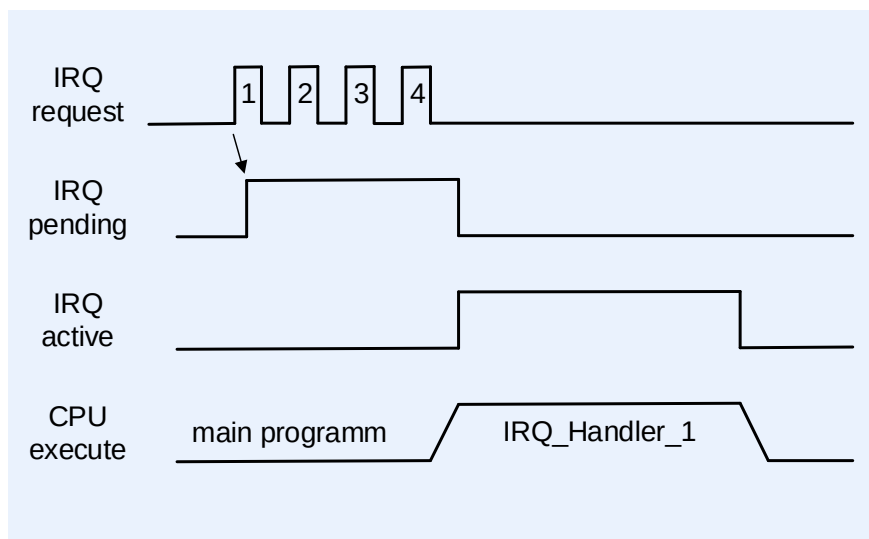


Рисунок 37-5 – Многократная установка снятие запроса IRQ request

Если запрос на прерывание пришел в момент активного прерывания, то в такой ситуации уже будут отработаны оба запроса на прерывание. В отличие от случая, изображенного на рисунке выше (Рисунок 37-5), запрос приходит тогда, когда признак отложенного прерывания IRQ pending уже сброшен, и новый запрос как раз его выставляет, что в дальнейшем позволяет провести повторную обработку прерывания (Рисунок 37-6).

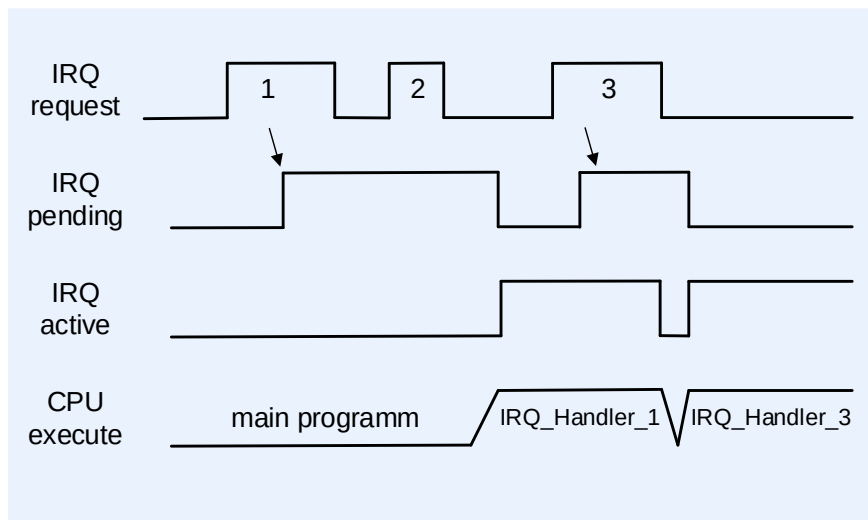


Рисунок 37-6 – Повторная установка запроса на прерывание в момент выполнения обработчика исключения

Выставление признака отложенного прерывания возможно даже в тех случаях, когда соответствующее прерывание запрещено. Все отложенные прерывания будут отражены в ISPR/ICPR, и в случае разрешения таких прерываний регистром ISPR, ядро тут же приступит к их обработке. Рекомендуется перед разрешением соответствующего прерывания убедиться в отсутствии признака отложенного запроса и, при необходимости, сбросить его.

37.2 Упрощенный доступ к регистрам контроллера прерываний

В целях повышения эффективности разработки программного обеспечения в CMSIS предусмотрен упрощенный доступ к регистрам контроллера прерываний NVIC из среды разработки программного обеспечения:

- регистры разрешения, запрета, установки и сброса состояния ожидания прерываний, а также регистр активных прерываний отображаются на массивы 32-разрядных целых чисел, а именно:

- массив ISER[0] соответствует регистру ISER0;
- массив ICER[0] соответствует регистру ICER0;
- массив ISPR[0] соответствует регистру ISPR0;
- массив ICPR[0] соответствует регистру ICPR0;
- массив IABR[0] соответствует регистру IABR0.

- четырехбитные поля регистра приоритета прерываний отображаются на массив четырехразрядных целых чисел, а именно:

массив IP[0]...IP[29] соответствует регистрам IPR0-IPR7, причем элемент массива IP[n] соответствует приоритету прерывания с номером n.

CMSIS генерирует код, гарантированно обеспечивающий в условиях многозадачности корректный непрерываемый (atomic) доступ к регистрам приоритета. Более подробная информация изложена в описании функции NVIC_SetPriority в разделе «Рекомендации по программированию контроллера прерываний NVIC».

Таблица 37-2 показывает отображение прерываний (номеров запросов IRQ) на регистры прерываний и соответствующие переменные CMSIS, для которых предусмотрено по одному биту на прерывание.

Таблица 37-2 – Распределение прерываний в переменных прерывания

Номер прерывания	Элементы массивов CMSIS [*]				
	Разрешение	Запрет	Установка режима ожидания	Сброс режима ожидания	Признак активности
0 – 29	ISER[0]	ICER[0]	ISPR[0]	ICPR[0]	IABR[0]
* Каждый элемент массива соответствует одному регистру контроллера прерываний NVIC, например элемент ICER[1] соответствует регистру ICER1					

37.2.1 NVIC->ISER[x]

Регистр ISER0 предназначен для разрешения прерываний (запись) и определения, какие из прерываний разрешены (чтение).

Таблица 37-3 – Регистр разрешения прерываний

Номер	31...0
Доступ	R/W
Сброс	0
	SETENA

Назначение бит **SETENA**:

запись:

0 – не влияет;

1 – разрешение прерывания.

чтение:

0 – прерывание запрещено;

1 – прерывание разрешено.

При разрешении прерывания, находящегося в состоянии ожидания обработки, контроллер NVIC активизирует его в зависимости от приоритета. Запрос запрещенного прерывания переводит его в состояние ожидания обработки, однако контроллер NVIC не активизирует его вне зависимости от приоритета.

37.2.2 NVIC->ICER[x]

Регистр запрета прерываний

Регистр ICER0 предназначен для запрета прерываний (запись) и определения, какие из прерываний разрешены (чтение) (Таблица 37-4).

Таблица 37-4 – Регистр запрета прерываний

Номер	31... 0
Доступ	R/W
Сброс	0
	CLRENA

Назначение бит **CLRENA**:

запись:

0 – не влияет;

1 – запрет прерывания.

чтение:

0 – прерывание запрещено;

1 – прерывание разрешено.

37.2.3 NVIC->ISPR[x]

Регистр установки состояния ожидания для прерывания

Регистр ISPR0 предназначен для принудительного перевода прерываний в состояние ожидания обслуживания (запись) и определения, какие из прерываний находятся в этом состоянии (чтение) (Таблица 37-5).

Таблица 37-5 – Регистр установки состояния ожидания для прерывания

Номер	31...0
Доступ	R/W
Сброс	0
	SETPEND

Назначение бит **SETPEND**:

запись:

0 – не влияет;

1 – перевод прерывания в состояние ожидания.

чтение:

0 – прерывание не ожидает обслуживания;

1 – прерывание ожидает обслуживания.

Запись 1 в бит регистра ISPR, соответствующий:

- прерыванию, уже ожидающему обслуживания – не влияет на работу системы;
- запрещенному прерыванию – переводит его в состояние ожидания.

37.2.4 NVIC->ICPR[x]

Регистр сброса состояния ожидания для прерывания

Регистр ICPR0 предназначен для принудительного сброса состояния ожидания обслуживания прерывания (запись) и определения, какие из прерываний находятся в состоянии ожидания (чтение).

Таблица 37-6 – Регистр сброса состояния ожидания для прерывания

Номер	31...0
Доступ	R/W
Сброс	0
	CLRPEND

Назначение бит **CLRPEND**:

запись:

0 – не влияет;

1 – сброс состояния ожидания.

чтение:

0 – прерывание не ожидает обслуживания;

1 – прерывание ожидает обслуживания.

Запись «1» в разряд регистра ICPR, соответствующий прерыванию в активном состоянии, не влияет на работу системы.

37.2.5 NVIC->IABR[x]

Регистр активных прерываний

Регистр ICPR0 показывает, какие из прерываний находятся в активном состоянии. Этот регистр доступен только для чтения (Таблица 37-7).

Таблица 37-7 – Регистр активных прерываний

Номер	31...0
Доступ	RO
Сброс	0
	ACTIVE

Назначение бит **ACTIVE**:

чтение:

0 – прерывание не активно;

1 – прерывание активно и обслуживается, либо активно и ожидает обслуживания.

37.2.6 NVIC->IP[x]

Регистры приоритета прерываний

Регистры IPR0-IPR7 представляют собой набор четырехбитных полей, каждое из которых соответствует одному прерыванию. Регистры доступны побайтно.

Каждый из регистров содержит четыре поля приоритета, которые отображаются на четыре элемента массива IP[0]...IP[29] CMSIS, как показано в таблице ниже (Таблица 37-8).

Таблица 37-8 – Регистры приоритета прерываний

IP

Номер	31...16	15...8	7...0
Доступ	U	R/W	R/W
Сброс	0	0	0
	-	IP[29]	IP[28]

IP

Номер	31...24	23...16	15...8	7...0
Доступ	R/W	R/W	R/W	R/W
Сброс	0	0	0	0
	IP[4m+3]	IP[4m+2]	IP[4m+1]	IP[4m]

IP

Номер	31...24	23...16	15...8	7...0
Доступ	R/W	R/W	R/W	R/W
Сброс	0	0	0	0
	IP[3]	IP[2]	IP[1]	IP[0]

Каждое поле содержит значение приоритета в диапазоне от 0 до 15, причем меньшие значения соответствуют более высокому приоритету соответствующего прерывания. Процессор обеспечивает доступ только к битам [7:5] приоритета, биты [4:0] при чтении всегда равны нулю, а при записи игнорируются.

Номер регистра IPR и смещение данных в регистре для заданного номера прерывания N определяются следующими соотношениями:

- номер M соответствующего регистра приоритета равен $M = N \text{ DIV } 4$;
- смещение данных в регистре в зависимости от значения $N \text{ MOD } 4$ равно:
 - 0 – биты регистра [7:0];
 - 1 – биты регистра [15:8];
 - 2 – биты регистра [23:16];
 - 3 – биты регистра [31:24].

37.2.7 NVIC->STIR

Регистр программного формирования прерывания

Запись в регистр STIR приводит к формированию в системе программного прерывания (SGI – Software Generated Interrupt).

В случае, если бит USERSETMPEND в регистре SCR установлен в 1, возможен доступ к регистру STIR из непривилегированных приложений (см. «Регистр управления системой»). Установка этого бита возможна только из привилегированного режима работы процессора.

Таблица 37-9 – Регистр программного формирования прерывания

Номер	31...9	8...0
Доступ	U	WO
Сброс	0	0
	-	INTID

INTID – идентификатор формируемого прерывания в диапазоне 0 – 239.

Например: значение b000000011 соответствует прерыванию IRQ3.

37.3 Прерывания, срабатывающие по уровню сигнала

Процессор способен обрабатывать прерывания, сформированные по уровню сигнала. Формирование запроса на прерывание по уровню происходит при условии удержания сигнала не менее двух тактов процессорного ядра.

Прерывание такого типа считается активным до тех пор, пока периферийное устройство не снимет активный уровень сигнала запроса. Как правило, это происходит после соответствующего обращения процедуры обработки прерывания к периферийному устройству.

После того, как процессор передал управление на обработчик, он автоматически снимает признак ожидания обслуживания прерывания (см. подраздел 37.3 «Аппаратное и программное управление прерываниями»). Если прерывание формируется по уровню сигнала, а сигнал запроса не снят до возврата из обработчика, процессор вновь переведет прерывание в состояние ожидания обслуживания, что, в свою очередь, приведет к повторному вызову его обработчика. Таким образом, периферийное устройство может поддерживать сигнал запроса прерывания в активном состоянии до тех пор, пока не перестанет нуждаться в обслуживании.

37.4 Аппаратное и программное управление прерываниями

Процессор RISC регистрирует все поступающие прерывания. Перевод прерывания, сформированного периферийным устройством, в состояние ожидания обслуживания осуществляется в одном из следующих случаев:

- контроллер прерываний NVIC обнаруживает, что сигнал запроса имеет высокий логический уровень, а прерывание неактивно;
- контроллер прерываний NVIC обнаруживает передний фронт сигнала запроса прерывания;
- программное обеспечение осуществляет запись в соответствующий разряд регистра ISPR0 (см. пункт 37.2.3 «NVIC->ISPR[x]») или соответствующего значения в регистр STIR (см. пункт 37.2.7 «NVIC->STIR»).

Прерывание находится в состоянии ожидания до тех пор, пока не произойдет одно из следующих событий:

- процессор передаст управление процедуре обработки прерывания. В этом случае прерывание переходит в активное состояние, после чего:
 - по завершении обработки прерывания, срабатывающего по уровню, контроллер NVIC проверяет состояние сигнала запроса на прерывание. Если этот сигнал активен, прерывание вновь переводится в состояние ожидания обслуживания, что приводит к немедленной повторной передаче управления на обработчик. В противном случае прерывание переводится в неактивное состояние;
 - если в период выполнения процедуры обработки прерывания, настроенного на срабатывание по фронту, не было зафиксировано импульсов на линии запроса, прерывание переводится в неактивное состояние.
- программное обеспечение осуществляет запись в соответствующий разряд регистра сброса состояния ожидания прерывания.

37.5 Рекомендации по работе с контроллером прерываний

Доступ к регистрам контроллера из программного обеспечения должен осуществляться по корректно выровненным адресам. Процессор не поддерживает возможность доступа к контроллеру по невыровненным адресам. Требования по выравниванию приведены в описании регистров.

Прерывание может быть переведено в состояние ожидания обслуживания даже в случае, если оно запрещено.

Перед установкой нового адреса таблицы векторов прерывания необходимо убедиться, что элементы новой таблицы корректно проинициализированы адресами обработчиков отказов и всех разрешенных исключений, в частности, прерываний. Более подробная информация представлена в пункте 38.1.4 «SCB->VTOR».

Программное разрешение или запрещение прерываний может осуществляться с помощью инструкций CPSIE I и CPSID I. В CMSIS предусмотрены следующие встроенные функции, генерирующие эти инструкции:

```
void __disable_irq(void) // Disable Interrupts
void __enable_irq(void) // Enable Interrupts
```

Кроме того, в CMSIS имеется ряд дополнительных функций, обеспечивающих управление контроллером прерываний NVIC:

Таблица 37-10 – Функции CMSIS для управления контроллером прерываний

Функция	Описание
void NVIC_SetPriorityGrouping (uint32_t priority_grouping)	Установить группировку приоритетов
void NVIC_EnableIRQ(IRQn_t IRQn)	Разрешить IRQn
void NVIC_DisableIRQ(IRQn_t IRQn)	Запретить IRQn
uint32_t NVIC_GetPendingIRQ (IRQn_t IRQn)	Вернуть TRUE, если прерывание IRQn ожидает обслуживания, FALSE – в противном случае
void NVIC_SetPendingIRQ (IRQn_t IRQn)	Перевести IRQn в состояние ожидания обслуживания
void NVIC_ClearPendingIRQ (IRQn_t IRQn)	Сбросить состояние ожидания обслуживания для IRQn
uint32_t NVIC_GetActive (IRQn_t IRQn)	Вернуть номер IRQ текущего активного прерывания
void NVIC_SetPriority (IRQn_t IRQn, uint32_t priority)	Установить приоритет для IRQn
uint32_t NVIC_GetPriority (IRQn_t IRQn)	Считать приоритет IRQn
void NVIC_SystemReset (void)	Сбросить систему

Более подробная информация отражена в документации по CMSIS.

38 Блок управления системой RISC

Блок управления системой SCB обеспечивает доступ к информации о конфигурации и управление работой системы. Регистры блока управления системой представлены в таблице (Таблица 38-1).

Таблица 38-1 – Обобщенная информация о регистрах блока управления системой

Адрес	Имя	Тип	Доступ	Значение после сброса	Описание
0xE000E000	InterruptType				
0x008	ACTLR	RW	Привилегированный	0x00000000	Дополнительный регистр управления
0xE000ED00	SCB				Блок управления системой
0x000	CPUID	RO	Привилегированный	0x412FC230	Регистр идентификации процессора
0x004	ICSR	RW	Привилегированный	0x00000000	Регистр управления прерываниями
0x008	VTOR	RW	Привилегированный	0x00000000	Регистр смещения таблицы векторов прерываний
0x00C	AIRCR	RW	Привилегированный	0xFA050000	Регистр управления прерываниями и программного сброса
0x010	SCR	RW	Привилегированный	0x00000000	Регистр управления системой
0x014	CCR	RW	Привилегированный	0x00000200	Регистр конфигурации и управления
0x018	SHPR1	RW	Привилегированный	0x00000000	Регистр №1 приоритета системных обработчиков
0x01C	SHPR2	RW	Привилегированный	0x00000000	Регистр №2 приоритета системных обработчиков
0x020	SHPR3	RW	Привилегированный	0x00000000	Регистр №3 приоритета системных обработчиков
0x024	SHCRS	RW	Привилегированный	0x00000000	Регистр управления и состояния системных обработчиков
0x028	CFSR	RW	Привилегированный	0x00000000	Регистр состояния отказов с конфигурируемым приоритетом
0x028	MMSR	RW	Привилегированный	0x00	Регистр состояния отказов доступа к памяти
0x029	BFSR	RW	Привилегированный	0x00	Регистр состояния отказов доступа к шине

Адрес	Имя	Тип	Доступ	Значение после сброса	Описание
0x02A	UFSR	RW	Привилегированный	0x0000	Регистр состояния отказов, вызванных ошибками программирования
0x02C	HFSR	RW	Привилегированный	0x00000000	Регистр состояния аппаратного отказа
0x034	MMAR	RW	Привилегированный	Не определено	Регистр адреса отказа доступа к памяти
0x038	BFAR	RW	Привилегированный	Не определено	Регистр адреса отказа доступа к шине

38.1 Упрощенный доступ к регистрам блока управления системой

В целях повышения эффективности в CMSIS предусмотрен упрощенный доступ к регистрам SCB из среды разработки программного обеспечения, а именно, регистры SHPR1-SHPR3 в CMSIS отображаются на массив байтов SHP[0]...SHP[12].

38.1.1 InterruptType->ACTLR

Дополнительный регистр управления

Регистр ACTLR позволяет разрешить или запретить следующие возможности процессора:

- вложение условных инструкций (IT folding);
- использование буферизации записи в режиме отображения памяти по умолчанию (default memory map);
- прерывание многоэлементных инструкций чтения и записи регистров.

Обобщенные данные о регистре ACTLR представлены далее.

Таблица 38-2 – Дополнительный регистр управления

Номер	31...3	2	1	0
Доступ	U	R/W	R/W	R/W
Сброс	0	0	0	0
	-	DISFOLD	DISDEFWBUFF	DISMCYCINT

DISFOLD – установка разряда в «1» запрещает вложение условных инструкций (IT folding) (см. ниже «О вложении условных инструкций»).

DISDEFWBUFF – установка в «1» запрещает использование буфера записи при работе в режиме отображения памяти по умолчанию (default memory map). Это обеспечивает возможность локализовать любые отказы шины, однако приводит к снижению производительности системы, так как все операции записи данных в память должны быть завершены до того, как процессор перейдет к выполнению следующей инструкции. Данный бит влияет исключительно на функционирование буферов записи, реализованных в процессоре RISC.

DISMCYCINT – установка бита в «1» запрещает прерывание многоэлементных инструкций чтения и записи регистров (LDM и STM). Это приводит к увеличению задержки обработки прерываний, вследствие необходимости завершения выполнения инструкций LDM или STM перед началом сохранения контекста и передачи управления обработчику прерывания.

О вложении условных инструкций

В некоторых случаях процессор может начать выполнение первой инструкции в блоке IT, все еще выполняя инструкцию IT. Эта возможность, называемая далее вложением условных инструкций (IT folding), позволяет увеличить производительность системы, однако может привести к непостоянству времени выполнения тела цикла программы («джиттеру»). В случае, если в разрабатываемом приложении это нежелательно, следует установить бит DISFOLD в «1».

38.1.2 SCB->CPUID

Регистр идентификации процессора

Регистр CPUID содержит информацию о модели процессора, версии и варианте его реализации. Таблица 38-3 предоставляет более подробную информацию.

Таблица 38-3 – Регистр идентификации процессора

Номер	31...24	23...20	19...16	15...4	3...0
Доступ	RO	RO	RO	RO	RO
Сброс	0x41	0x2	0xF	0xC23	0x0
	Implementer	Variant	Constant	PartNo	Revision

Implementer – код разработчика 0x41 = MILANDR.

Variant – значение r в номере версии rnpn изделия: 0x2 = r2p0;

Constant – постоянное значение 0xF;

PartNo – номер модели процессора: 0xC23 = RISCore;

Revision – значение p в номере версии rnpn изделия: 0x0 = r2p0.

38.1.3 SCB->ICSR

Регистр управления прерываниями

Регистр ICSR обеспечивает возможность установки и сброса состояния ожидания обслуживания для исключений PendSV и SysTick, а также доступ к следующей информации:

- номер текущего обрабатываемого исключения;
- наличие активных исключений, обработка которых была прервана;
- номер исключения, ожидающего обслуживания, с наивысшим приоритетом;
- наличие прерываний, ожидающих обслуживания.

Таблица 38-4 – Регистр управления прерываниями

Номер	31...29	28	27	26	25	24	23	22	21...12	11	10, 9	8...0
Доступ	U	R/W	WO	R/W	WO	U	R/W	R/W	R/W	R/W	U	R/W
Сброс	0	0	0	0	0	0	0	0	0	0	0	0
	-	PENDSVSET	PENDSVCLR	PENDSTSET	PENDSTCLR	-	Reserved for Debug	ISR_PENDING	VECT_PENDING	RETTOBASE	-	VECTACTIVE

PENDSVSET (RW) – бит установки состояния ожидания обслуживания для исключения PendSV.

Запись:

0 – не влияет на работу системы;

1 – переводит исключение PendSV в состояние ожидания обслуживания.

Чтение:

0 – исключение PendSV не ожидает обслуживания;

1 – исключение PendSV ожидает обслуживания.

Запись 1 – это единственно возможный способ перевода исключения PendSV в состояние ожидания обслуживания.

PENDSVCLR (WO) – бит сброса состояния ожидания обслуживания для исключения PendSV.

Запись:

0 – не влияет на работу системы;

1 – сбрасывает состояние ожидания обслуживания для исключения PendSV.

PENDSTSET (RW) – бит установки состояния ожидания обслуживания для исключения SysTick.

Запись:

0 – не влияет на работу системы.

1 – переводит исключение SysTick в состояние ожидания обслуживания.

Чтение:

0 – исключение SysTick не ожидает обслуживания. 1 – ожидает.

PENDSTCLR (WO) – бит сброса состояния ожидания обслуживания для исключения SysTick.

Запись:

0 – не влияет на работу системы.

1 – сбрасывает состояние ожидания обслуживания для исключения SysTick.

Данный бит доступен только для записи, при чтении результат не определен.

Reserved for Debug use (RO) – этот бит зарезервирован для целей отладки, при чтении вне режима отладки возвращает значение 0.

ISR_PENDING (RO) – флаг наличия в системе прерываний (за исключением отказов), ожидающих обслуживания:

- 0 – ожидающие обслуживания прерывания отсутствуют;
- 1 – присутствуют.

VECT_PENDING (RO) – содержит номер ожидающего обслуживания исключения с наивысшим приоритетом, обработка которого в системе разрешена:

- 0 – необслуженных исключений нет;
- другое число – номер ожидающего обслуживания исключения.

Значение данного поля формируется с учетом полей BASEPRI и FAULTMASK, однако не учитывает влияние поля PRIMASK.

RETTOBASE (RO) – показывает наличие в системе активных исключений, обслуживание которых было прервано:

- 0 – присутствуют;
- 1 – отсутствуют.

VECT_ACTIVE (RO) – содержит номер активного исключения:

- 0 – режим приложения;
- другое число – номер текущего обслуживаемого исключения. Для получения номера запроса прерывания (IRQ) из значения VECT_ACTIVE необходимо вычесть 16.

Запись в регистр ICSR может привести к непредсказуемым результатам в случае:

- одновременной установки в 1 бит PENDSVSET и PENDSVCLR;
- одновременной установки в 1 бит PENDSTSET и PENDSTCLR.

38.1.4 SCB->VTOR

Регистр смещения таблицы векторов прерываний

Регистр VTOR содержит смещение базового адреса таблицы векторов прерываний относительно адреса 0x00000000.

Таблица 38-5 – Регистр смещения таблицы векторов прерываний

Номер	31, 30	29...7	6...0
Доступ	U	R/W	R/W
Сброс	0	0	0
	-	TBLOFF	Reserved

TBLOFF – смещение базового адреса таблицы векторов относительно нижней границы карты распределения памяти. Собственно смещение хранится в битах [28:7]. Бит [29] определяет, размещена ли таблица в области кода или в области памяти SRAM:

- 0 – область кода;
- 1 – SRAM. Бит [29] может также обозначаться как TBLBASE.

При установке значения TBLOFF требуется обеспечить выравнивание базового адреса таблицы векторов. Минимальный размер выравнивания – по границе блока из 32 слов, достаточен для хранения 16 векторов прерываний. Для поддержки большего количества прерываний необходимо увеличить размер выравнивания до ближайшей степени двойки, большей или равной размеру таблицы. Например, для хранения 21 вектора прерываний таблицу следует выровнять по границе блока из 64 слов, так как ее объем составляет 37 слов, а ближайшая степень двойки, большая или равная 37, равна 64.

Учитывая описанные выше требования по выравниванию, разряды [6...0] смещения всегда равны нулю.

38.1.5 SCB->AIRCRCR

Регистр управления прерываниями и программного сброса

Регистр AIRCRCR позволяет группировать исключения по приоритетам, задавать порядок следования байтов в слове (endian) при доступе к данным, а также управлять процессом сброса системы.

Для записи данных в регистр необходимо установить его поле VECTKEY в значение 0x05FA, в противном случае попытка записи будет проигнорирована процессором.

Таблица 38-6 – Регистр управления прерываниями и программного сброса

Номер	31...16	15	14...11	10...8	7...3	2	1	0
Доступ	R/W	RO	U	R/W	U	WO	WO	WO
Сброс	0	0	0	0	0	0	0	0
	On Read: VECTKEYSTAT, On Write: VECTKEY	ENDIANESS	.	PRIGROUP	.	SYSRESETRQ	VECTCLRACTIVE	VECTRESET

VECTKEYSTAT – ключ доступа к регистру. При чтении возвращает 0xFA05.

VECTKEY – ключ доступа к регистру. При записи должен быть равен 0x05FA, в противном случае попытка записи в регистр будет проигнорирована процессором.

ENDIANESS (RO) – порядок следования значащих разрядов при доступе к данным:

0 – младший байт идет первым (little-endian);

1 – старший байт идет первым (big-endian). Значение поля устанавливается, исходя из уровня конфигурационного сигнала BIGEND в момент сброса системы.

PRIGROUP (RW) – группировка приоритетов исключений. Значение данного поля определяет положение двоичной точки, разделяющей поле приоритета на поля номера группы и подгруппы приоритетов.

PRIGROUP – определяет позицию двоичной точки, разделяющей поля PRI_n регистров приоритета прерываний на два подполя – номер группы и номер подгруппы. Таблица 38-7 показывает зависимость этого разбиения от значения PRIGROUP.

Таблица 38-7 – Группировка приоритетов прерываний

PRIGROUP	Значение приоритета в поле PRI_N[7:0]			Общее количество	
	Положение двоичной точки	Биты номера группы	Биты номера подгруппы	групп	подгрупп
0b000, 0b001, 0b010, 0b011	bxxxx.0000	[7:4]	None	16	1
b100	bxxx.y0000	[7:5]	[4]	8	2
b101	bxx.yy0000	[7:6]	[5:4]	4	4
b110	bx.yyy0000	[7]	[6:4]	2	8
b111	b.yyyy0000	None	[7:4]	1	16

SYSRESETREQ (WO) – запрос сброса системы:

0 – не влияет на работу;

1 – инициирует сигнал сброса процессора. При чтении возвращает 0.

VECTCLRACTIVE (WO) – зарезервировано для целей отладки. При чтении возвращает 0. При записи данных в регистр значение поля должно быть равно 0, в противном случае результат непредсказуем.

VECTRESET (WO) – зарезервировано для целей отладки. При чтении возвращает 0. При записи данных в регистр значение поля должно быть равно 0, в противном случае результат непредсказуем.

38.1.6 SCB->SCR

Регистр управления системой

Регистр SCR позволяет определить требования к переходу в режим и выходу из режима пониженного энергопотребления.

Таблица 38-8 – Регистр управления системой

Номер	31...5	4	3	2	1	0
Доступ	U	R/W	U	R/W	R/W	U
Сброс	0	0	0	0	0	0
	-	SEVONPEND	-	SLEEPDEEP	SLEEONEXIT	-

SEVONPEND – разрешает или запрещает формирование сигнала события при переводе исключения в состояние ожидания обработки. 0 – выход из режима пониженного энергопотребления по прерыванию могут инициировать только разрешенные прерывания или события; 1 – выход может инициироваться разрешенными событиями и любыми, в том числе запрещенными, прерываниями.

Перевод прерывания в состояние ожидания обслуживания формирует событие, что в свою очередь приводит к выходу процессора из режима пониженного потребления, инициированного инструкцией WFE, либо к регистрации факта события, если эта инструкция еще не выполнялась.

Кроме того, процессор может быть выведен из режима пониженного энергопотребления при поступлении внешнего события, а также после выполнения инструкции SEV.

SLEEPDEEP – определяет режим пониженного энергопотребления процессора:

0 – спящий режим (Sleep);

1 – режим глубокого сна (Deep Sleep).

SLEEPONEXIT – разрешает или запрещает перевод процессора в режим пониженного энергопотребления при выходе из обработчика события в режим выполнения прикладной программы:

0 – не переводить;

1 – переводить.

38.1.7 SCB->CCR

Регистр конфигурации и управления

Регистр CCR управляет процессом перехода процессора в режим приложения, а также позволяет запретить или разрешить:

- игнорирование отказов доступа к шине в обработчиках аппаратных отказов и при эскалации отказа по FAULTMASK;
- генерацию исключений при делении на ноль и при доступе по невыровненному адресу;
- доступ к регистру STIR из непривилегированного приложения (см. пункт 37.2.7 «NVIC->STIR»).

Таблица 38-9 – Регистр конфигурации и управления

Номер	31...10	9	8	7...5	4	3	2	1	0
Доступ	U	R/W	R/W	U	R/W	R/W	U	R/W	R/W
Сброс	0	0	0	0	0	0	0	0	0
	-	STKALIGN	BHFNMIGN	-	DIV_O_TRP	UNALIGN_TRP	-	USERSETMPEND	NONBASETHRDENA

STKALIGN определяет режим выравнивания адреса стека при обработке исключений:

0 – выравнивание по границе 4 байт;

1 – выравнивание по границе 8 байт. При передаче управления на обработчик исключения процессор анализирует бит [9] сохраненного в стеке слова состояния PSR и определяет по нему режим выравнивания стека. При возврате из обработчика процессор использует сохраненный в стеке бит этого слова для восстановления требуемого режима выравнивания.

BFHFNMIIGN разрешает обработчикам с уровнем приоритета -1 и -2 игнорировать отказы доступа к шине, вызванные инструкциями чтения и записи. Бит влияет на функционирование обработчиков аппаратных отказов и при эскалации отказов по FAULTMASK:

0 – отказы доступа к шине данных, вызванные инструкциями чтения или записи, приводят к блокировке процессора;

1 – обработчики с уровнем приоритета -1 и -2 игнорируют указанные отказы доступа к шине данных. Данный бит следует устанавливать лишь в том случае, если обработчик и используемые им данные размещены в абсолютно безопасной области памяти. Как правило, данный бит используется для локализации и исправления проблем доступа к системным устройствам и мостам ввода-вывода.

DIV_0_TRP разрешает процессору формировать отказ или останавливаться в случае деления на ноль при выполнении инструкций SDIV или UDIV:

0 – не обрабатывать деление на 0;

1 – обрабатывать. В случае, если бит установлен в 0, при делении на ноль процессор устанавливает частное в 0.

UNALIGN_TRP разрешает процессору формировать отказ при невыровненном доступе к данным:

0 – не обрабатывать невыровненный доступ к словам или полусловам данных;

1 – обрабатывать. Если бит равен 1, то невыровненный доступ приводит к отказу, вызванному ошибкой программирования (usage fault).

В случае невыровненного доступа по инструкциям LDM, STM, LDRD или STRD отказ формируется всегда, вне зависимости от значения бита UNALIGN_TRP.

USERSETMPEND разрешает доступ к регистру STIR (см. пункт 37.2.7 «NVIC->STIR») из непривилегированного приложения:

0 – доступ запрещен;

1 – разрешен.

NONEBASETHRDENA определяет процедуру перехода процессора в режим приложения (Thread mode): 0 – процессор может перейти в режим приложения только в случае отсутствия активных исключений, 1 – процессор может перейти в режим приложения из обработчика любого уровня, в соответствии со значением слова EXC_RETURN (см. пункт 36.6.6 «Возврат из обработчика исключения»).

38.1.8 SCB->SHP[x]

Регистры приоритета системных обработчиков

Регистры приоритета системных обработчиков SHPR1-SHPR3 позволяют установить уровень приоритета обработки исключений.

Доступ к регистрам осуществляется побайтно.

Поля PRI_N регистров имеют ширину 8 бит, однако в процессоре реализована поддержка доступа только к старшему полубайту [7...4], при чтении данных из младшего полубайта [3...0] процессор возвращает нули, запись в этот полубайт игнорируется.

Таблица 38-10 – Поля приоритета обработчиков системных отказов

Обработчик отказа	Поле	Описание регистра
Отказ доступа к памяти	SHP[4]	Регистр №1 приоритета системных обработчиков
Отказ доступа к шине	SHP[5]	
Ошибка программирования (usage fault)	SHP[6]	
Вызов SVCall	SHP[11]	Регистр №2 приоритета системных обработчиков
Вызов PendSV	SHP[14]	Регистр №3 приоритета системных обработчиков
Вызов SysTick	SHP[15]	

Регистр №1 приоритета системных обработчиков

Таблица 38-11 – Регистр №1 приоритета системных обработчиков

Номер	31...24	23...16	15...8	7...0
Доступ	R/W	R/W	R/W	R/W
Сброс	0	0	0	0
	PRI_7: Резерв	PRI_6	PRI_5	PRI_4

PRI_7 Резерв.

PRI_6 Приоритет системного обработчика 6, ошибка программирования.

PRI_5 Приоритет системного обработчика 5, отказ доступа к шине.

PRI_4 Приоритет системного обработчика 4, отказ доступа к памяти.

Регистр №2 приоритета системных обработчиков

Таблица 38-12 – Регистр №2 приоритета системных обработчиков

Номер	31...24	23...0
Доступ	R/W	U
Сброс	0	0
	PRI_11	-

PRI_11 Приоритет системного обработчика 11, вызов SVCall.

Регистр №3 приоритета системных обработчиков

Таблица 38-13 – Регистр №3 приоритета системных обработчиков

Номер	31...24	23...16	15... 0
Доступ	R/W	R/W	U
Сброс	0	0	0
	PRI_15	PRI_14	.

PRI_15 Приоритет системного обработчика 15, вызов SysTick.

PRI_14 Приоритет системного обработчика 14, вызов PendSV.

38.1.9 SCB->SHCSR

Регистр управления и состояния системных обработчиков.

Регистр SHCSR позволяет разрешить или запретить работу системных обработчиков, а также содержит сведения:

- о наличии ожидающих обработки отказов доступа к шине, управления памятью, а также вызов SVCall;
- об активных системных обработчиках.

Таблица 38-14 – Регистр управления и состояния системных обработчиков

Номер	31...19	18	17	16	15	14	13	12	11	10	9	8	7	6...4	3	2	1	0
Доступ	U	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	U	R/W	R/W	U	R/W	U	R/W	R/W
Сброс	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	-	USGFAULTENA	BUSFAULTENA	MEMFAULTENA	SVCALLPENDE	BUSFAULTPENDE	MEMFAULTPENDE	USGFAULTPENDE	SYSTICKACT	PENDSVACT	-	MONITORACT	CVCALLAVCT	-	USGFAULTACT	-	BUSFAULTACT	MEMFAULTACT

USGFAULTENA разрешение обработки отказов, вызванных ошибками программирования: 1 – разрешено, 0 – запрещено.

BUSFAULTENA разрешение обработки отказа доступа к шине: 1 – разрешено, 0 – запрещено.

MEMFAULTENA разрешение обработки отказа доступа к памяти: 1 – разрешено, 0 – запрещено.

SVCALLPENDED признак ожидания обработки вызова SVC: возвращает 1, если вызов ожидает обработки.

BUSFAULTPENDED признак ожидания обработки отказа доступа к шине: возвращает 1, если отказ ожидает обработки.

MEMFAULTPENDED признак ожидания обработки отказа доступа к памяти: возвращает 1, если отказ ожидает обработки.

USGFAULTPENDED признак ожидания обработки отказа, вызванного ошибками программирования: возвращает 1, если отказ ожидает обработки.

SYSTICKACT признак активности обработчика исключения SysTick: возвращает 1, если обработчик активен.

PENDSVACT признак активности обработчика исключения PendSV: возвращает 1, если обработчик активен.

MONITORACT признак активности монитора отладчика: возвращает 1, если монитор отладчика активен.

SVCALLACT признак активности обработчика вызова SVC: возвращает 1, если обработчик активен.

USGFAULTACT признак активности обработчика отказа, вызванного ошибкой программирования: возвращает 1, если обработчик активен.

BUSFAULTACT признак активности обработчика отказа доступа к шине, возвращает 1, если обработчик активен.

MEMFAULTACT признак активности обработчика отказа доступа к памяти: возвращает 1, если обработчик активен.

Примечания

1 Установка бита разрешения в 1 разрешает обработку исключения, установка в 0 – запрещает;

2 Чтение 1 из бита-признака активности свидетельствует об активности исключения, 0 – о его неактивности. Существует возможность записи значения в данный бит для принудительного перевода исключения в активное состояние, однако при этом следует предпринять меры предосторожности, описанные далее в разделе;

3 Чтение 1 из бита-признака ожидания свидетельствует о том, что исключение находится в состоянии ожидания обработки. Существует возможность принудительного перевода исключения в состояние ожидания путем записи 1 в данный бит.

Если в системе возникло исключение (отказ), обработчик которого запрещен, процессор формирует запрос на обработку аппаратного отказа.

Существует возможность принудительного перевода того или иного системного исключения в состояние ожидания обработки или активное состояние путем записи в соответствующий разряд регистра SHCSR.

Например, ядро операционной системы может осуществлять запись в биты – признаки активности для того, чтобы осуществить переключение контекста со сменой типа обрабатываемого исключения.

Программа, меняющая значение бит – признаков активности исключения, должна обеспечить необходимую корректировку содержимого стека, в противном случае процессор может сгенерировать отказ. Необходимо убедиться, что программа сохраняет и впоследствии корректно восстанавливает текущее значение признаков активности исключений.

После разрешения системных обработчиков все дальнейшие манипуляции с битами регистра необходимо производить, последовательно выполняя операции чтения, модификации и обратной записи, гарантирующие изменение только необходимых разрядов регистра.

38.1.10 SCB->CFSR

Регистр состояния отказов с конфигурируемым уровнем приоритета.

Регистр CFSR содержит информацию о причине возникновения отказов управления памятью, отказов доступа к шине и ошибок программирования (usage fault).

Таблица 38-15 – Регистр состояния отказов с конфигурируемым уровнем приоритета

Номер	31...16	15...8	7...0
Доступ	R/W	R/W	R/W
Сброс	0	0	0
	Usage Fault Status Register: UFSR	Bus Fault Status Register: BFSR	Memory Management Fault Status Register: MMFSR

Регистр CFSR доступен побайтно. Возможны следующие варианты доступа к регистру CFSR и его отдельным элементам:

- слово по адресу 0xE000ED28 – полный регистр CFSR;
- байт по адресу 0xE000ED28 – регистр MMFSR;
- полуслово по адресу 0xE000ED28 – регистры MMFSR и BFSR;
- байт по адресу 0xE000ED29 – регистр BFSR;
- полуслово по адресу 0xE000ED2A – регистр UFSR.

В последующих подразделах подробно описаны элементы, составляющие регистр CFSR:

- регистр состояния отказов доступа к памяти;
- регистр состояния отказов доступа к шине;
- регистр состояния отказов, вызванных ошибками программирования.

38.1.10.1 Поле MMFSR

Регистр состояния отказов доступа к памяти

Регистр MMFSR содержит набор флагов, указывающих на различные причины отказа доступа к памяти.

Таблица 38-16 – Регистр состояния отказов доступа к памяти

Номер	7	6, 5	4	3	2	1	0
Доступ	R/W	U	R/W	R/W	U	R/W	R/W
Сброс	0	0	0	0	0	0	0
	MMARVALID	-	MSTKERR	MUNSTKERR	-	DACCVIOL	IACCVIOL

MMARVALID признак корректности значения в регистре адреса отказа доступа к памяти (MMAR):

0 – значение в MMAR не содержит корректный адрес отказа;

1 – содержит.

В случае, если произошла эскалация отказа доступа к памяти, обработчик аппаратного отказа должен установить этот бит в 0. В противном случае, после возврата в обработчик отказа доступа к памяти, возможна его некорректная работа, так как значение регистра MMAR будет изменено.

MSTKERR признак отказа на этапе сохранения в стеке контекста при передаче управления на обработчик исключения:

0 – отсутствует;

1 – попытка сохранения в стеке контекста при вызове обработчика исключения вызвала одно или несколько нарушений доступа к памяти. В случае, если бит равен 1, значение указателя стека SP по прежнему корректно, однако содержимое стека может быть неверным. Адрес отказа в регистр MMAR не записывается.

MUNSTKERR признак отказа на этапе восстановления контекста из стека при выходе из обработчика исключения:

0 – отсутствует;

1 – попытка восстановления контекста из стека вызвала одно или несколько нарушений доступа к памяти.

Передача управления на обработчик данного отказа осуществляется без сохранения контекста. Таким образом, в случае, если данный бит равен 1, состояние стека сохраняется, значение указателя стека не меняется, контекст не сохраняется.

Адрес отказа в регистр MMAR не записывается.

DACCVIOL признак нарушения доступа к памяти данных:

0 – отсутствует;

1 – процессор попытался прочесть или записать данные в области, для которой не разрешен такой тип доступа. Если бит равен 1, значение счетчика команд PC, сохраненное в стеке, указывает на инструкцию, вызвавшую отказ. В регистре MMAR содержится адрес, по которому была осуществлена попытка доступа к памяти.

IACCVIOL признак нарушения доступа к памяти команд:

0 – отсутствует;

1 – процессор попытался считать очередную команду из области памяти, для которой не разрешено выполнение. Этот отказ возникает всякий раз при доступе к области, помеченной как неразрешенная для выполнения (XN), даже в случае, если блок защиты памяти MPU не активен (disabled) или отсутствует. Если бит равен 1, значение счетчика команд PC, сохраненное в стеке, указывает на инструкцию, вызвавшую отказ. Адрес отказа в регистр MMAR не записывается.

38.1.10.2 Поле BFSR

Регистр состояния отказов доступа к шине

Регистр BFSR содержит набор флагов, указывающих на различные причины отказа доступа к шине:

Таблица 38-17 – Регистр состояния отказов доступа к шине

Номер	7	6	5	4	3	2	1	0
Доступ	R/W	U	U	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0	0	0
	BFRVALID			STKERR	UNSTKERR	IMPRECISER	PRECISER	IBUSER
	D			R	R	R	R	R

BFRVALID признак корректности значения в регистре адреса отказа доступа к шине (BFAR):

0 – значение в BFAR не содержит корректный адрес отказа;

1 – содержит.

Процессор устанавливает этот бит в 1 в случае, если известен адрес, при доступе по которому произошел отказ. Возникновение впоследствии других отказов, например отказов управления памятью, может сбросить этот бит в 0.

В случае, если возникла эскалация отказа, обработчик аппаратного отказа должен установить этот бит в 0. В противном случае, после возврата в обработчик отказа доступа к шине возможна его некорректная работа, так как значение регистра MMAR будет изменено.

STKERR признак отказа на этапе сохранения в стеке контекста при передаче управления на обработчик исключения: 0 – отсутствует, 1 – попытка сохранения в стеке контекста при вызове обработчика исключения вызвала одно или несколько нарушений доступа к шине. В случае, если бит равен 1, значение указателя стека SP по прежнему корректно, однако содержимое стека может быть неверным. Адрес отказа в регистр BFAR не записывается.

UNSTKERR признак отказа на этапе восстановления контекста из стека при выходе из обработчика исключения:

0 – отсутствует;

1 – попытка восстановления контекста из стека вызвала одно или несколько нарушений доступа к шине.

Передача управления на обработчик данного отказа осуществляется без сохранения контекста. Таким образом, в случае, если данный бит равен 1, состояние стека сохраняется, значение указателя стека не меняется, контекст не сохраняется.

Адрес отказа в регистр BFAR не записывается.

IMPRECISERR признак нелокализованной ошибки доступа к шине данных:

0 – отсутствует;

1 – произошла ошибка доступа к шине данных, однако адрес возврата в стековом фрейме не указывает на инструкцию, вызвавшую ошибку. В случае, если процессор установил этот бит в 1, адрес отказа в регистр BFAR не записывается. Данный отказ является асинхронным, таким образом, если он возник внутри процесса, приоритет которого выше, чем приоритет обработки отказа шины, процессор переводит его в состояние ожидания обслуживания до завершения более приоритетных процессов. В случае, если до передачи управления на обработчик возникла также локализованная ошибка доступа к шине, процессор устанавливает оба соответствующих флага.

PRECISERR признак локализованной ошибки доступа к шине данных:

0 – отсутствует;

1 – произошла ошибка доступа к шине данных, при этом адрес возврата в стековом фрейме указывает на инструкцию, вызвавшую ошибку. В случае, если процессор установил этот бит в 1, он также записывает адрес отказа в регистр BFAR.

IBUSERR признак ошибки доступа к шине инструкций:

0 – отсутствует;

1 – произошла ошибка доступа к шине инструкций. Процессор обнаруживает факт ошибки доступа к шине инструкций на этапе выборки очередной команды, однако признак IBUSERR устанавливается только после попытки выполнения этой инструкции. В случае, если процессор установил этот бит в 1, адрес отказа в регистр BFAR не записывается.

38.1.10.3 Поле UFSR

Регистр состояния отказов, вызванных ошибками программирования

Регистр UFSR содержит набор флагов, указывающих на различные причины отказа.

Таблица 38-18 – Регистр состояния отказов, вызванных ошибками программирования

Номер	15...10	9	8	7...4	3	2	1	0
Доступ	U	R/W	R/W	U	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0	0	0
	-	DIVBYZERO	UNALIGNED	-	NOCP	INVPC	INVSTATE	UNDEFINSTR

DIVBYZERO признак деления на ноль:

0 – деления на ноль не было, либо обработка данного типа ошибки запрещена;

1 – процессор выполнил инструкцию SDIV или UDIV с делителем равным 0. Если бит равен 1, значение счетчика команд PC, сохраненное в стеке, указывает на инструкцию, вызвавшую отказ. Разрешить либо запретить обработку деления на ноль можно путем установки в 1 бита DIV_0_TRP регистра CCR (см. пункт 38.1.7 «SCB->CCR»).

UNALIGNED признак доступа к памяти по невыровненному адресу:

0 – не было, либо обработка данного типа ошибки запрещена;

1 – процессор попытался обратиться к памяти по невыровненному адресу. Разрешить либо запретить обработку этой ошибки можно путем установки в 1 бита UNALIGN_TRP регистра CCR (см. пункт 38.1.7 «SCB->CCR»). Инструкции LDM, STM, LDRD, и STRD, пытающиеся обратиться по невыровненному адресу, вызывают исключение всегда, вне зависимости от значения бита UNALIGN_TRP.

NOCSP попытка обращения к сопроцессору. Процессор не поддерживает инструкции, требующие наличия сопроцессора.

0 – попытки не было;

1 – попытка была.

INVPC загрузка неверного значения в счетчик команд PC.

0 – не было;

1 – процессор попытался загрузить в счетчик команд PC неверное значение EXC_RETURN, вследствие неправильного восстановления контекста, либо неверного значения EXC_RETURN. Если бит равен 1, значение счетчика команд PC, сохраненное в стеке, указывает на инструкцию, пытавшуюся загрузить неверное значение в PC.

INVSTATE неверное состояние:

0 – не было;

1 – процессор попытался выполнить инструкцию, связанную с неверным использованием регистра EPSR. Если бит равен 1, значение счетчика команд PC, сохраненное в стеке, указывает на инструкцию, попытавшуюся некорректно использовать регистр EPSR.

UNDEFINSTR попытка выполнения неверной инструкции:

0 – попытки не было;

1 – процессор попытался выполнить неверную инструкцию. Если бит равен 1, значение счетчика команд PC, сохраненное в стеке, указывает на инструкцию, вызвавшую отказ. Под неверной понимается инструкция, которую процессор не смог декодировать.

После установки в 1 биты регистра UFSR сохраняют это значение до тех пор, пока не будут принудительно сброшены путем записи в них 1, либо до сброса системы.

38.1.11 SCB->HFSR

Регистр состояния аппаратного отказа

Регистр HFSR содержит сведения о причинах вызова обработчика аппаратного отказа. Особенностью данного регистра является то, что для сброса в 0 его разрядов необходимо записать в них значение 1.

Таблица 38-19 – Регистр состояния аппаратного отказа

Номер	31	30	29...2	1	0
Доступ	R/W	R/W	U	R/W	R/W
Сброс	0	0	0	0	0
	DEBUGEVT	FORCED	-	VECTTBL	Reserved

DEBUGEVT бит зарезервирован для отладки. При записи в регистр данный бит должен быть равен 0, в противном случае поведение процессора непредсказуемо.

FORCED признак аппаратного отказа, возникшего вследствие эскалации отказа с конфигурируемым уровнем приоритета, который не может быть обработан (запрещен или имеет недостаточно высокий приоритет):

0 – нет;

1 – да.

Если этот бит равен 1, то для определения причины отказа обработчику следует прочитать значения остальных разрядов регистров HFSR.

VECTTBL признак возникновения отказа шины при попытке доступа к таблице векторов исключений:

0 – не было;

1 – было. Эта ошибка всегда вызывает передачу управления на обработчик аппаратного отказа. Если бит равен 1, значение счетчика команд РС, сохраненное в стеке, указывает на инструкцию, выполнение которой было прервано для обработки исключения.

После установки в 1 биты регистра HFSR сохраняют это значение до тех пор, пока не будут принудительно сброшены путем записи в них 1, либо до сброса системы.

38.1.12 SCB->MMFAR

Регистр адреса отказа доступа к памяти

Регистр MMFAR содержит адрес, при обращении по которому возникла ошибка управления памятью.

Таблица 38-20 – Регистр адреса отказа доступа к памяти

Номер	31...0
Доступ	R/W
Сброс	0
	ADDRESS

ADDRESS если бит MMARVALID регистра MMFSR равен 1, это поле содержит адрес, при обращении по которому возникла ошибка управления памятью. В случае ошибки доступа по невыровненному адресу поле содержит фактическое значение адреса, вызвавшего отказ.

Учитывая, что одна единственная операция чтения или записи может быть разбита процессором на несколько операций доступа по выровненному адресу, в регистре адреса отказа может находиться любое значение в диапазоне адресов, по которым осуществлялась попытка доступа.

Флаги регистра MMFSR содержат информацию о причине отказа, а также сообщают, является ли значение MMFAR корректным. Подробнее см. пункт 36.7.3 «Регистры состояния и адреса отказа».

38.1.13 SCB->BFAR

Регистр адреса отказа доступа к шине

Регистр BFAR содержит адрес, при обращении по которому возникла ошибка доступа к шине.

Таблица 38-21 – Регистр адреса отказа доступа к шине

Номер	31...0
Доступ	R/W
Сброс	0
	ADDRESS

ADDRESS если бит BFARVALID регистра BFSR равен 1, это поле содержит адрес, при обращении по которому возникла ошибка доступа к шине. В случае ошибки доступа по невыровненному адресу поле содержит значение адреса, запрошенного командой процессора, даже если оно не совпадает с адресом, вызвавшим отказ.

Флаги регистра BFSR содержат информацию о причине отказа, а также сообщают, является ли значение BFAR корректным. Подробнее см. пункт 36.7.3 «Регистры состояния и адреса отказа».

Рекомендации по программированию блока управления системой

Необходимо убедиться, что программа использует для обращения к регистрам блока управления системой доступ по корректно выровненным адресам. Обращение ко всем регистрам, за исключением CFSR и SHPR1-SHPR3, должно быть выровнено по границе слова. Регистры CFSR и SHPR1-SHPR3 допускают как побайтный доступ, так и доступ по адресам, выровненным по границе слова или полуслова.

Для того, чтобы определить истинный адрес, вызвавший отказ, в обработчике необходимо выполнить следующие действия:

- считать и сохранить значения регистров MMFAR или BFAR;
- проверить значение бита MMARVALID регистра MMFSR, либо бита BFARVALID регистра BFSR. Значения MMFAR или BFAR корректны только в случае, если соответствующие биты равны 1.

Рекомендуется именно такая последовательность операций, так как возникновение исключения с более высоким приоритетом может изменить значения в регистрах MMFAR и BFAR, например, в случае возникновения сбоя в обработчике более высокоприоритетного исключения.

39 Сторожевые таймеры

39.1 Описание регистров блока сторожевых таймеров

Таблица 39-1 – Описание регистров блока сторожевого таймера IWDG

Базовый Адрес	Название	Описание
0x4006_8000	IWDG	Сторожевой таймер IWDG
Смещение		
0x00	IWDG_KR[15:0]	Регистр ключа
0x04	IWDG_PR[2:0]	Делитель частоты сторожевого таймера
0x08	IWDG_RLR[11:0]	Регистр основания счета сторожевого таймера
0x0C	IWDG_SR[1:0]	Регистр статуса сторожевого таймера

Таблица 39-2 – Описание регистров блока сторожевого таймера WWDG

Базовый Адрес	Название	Описание
0x4006_0000	WWDG	Сторожевой таймер WWDG
Смещение		
0x00	WWDG_CR[7:0]	Регистр управления
0x04	WWDG_CFR[9:0]	Регистр конфигурации
0x08	WWDG_SR[0]	Регистр статуса

39.1.1 Блок сторожевого таймера IWDG

Блок сторожевого таймера IWDG – независимый 12-разрядный сторожевой таймер, который считает вниз от основания счета (значения перезагрузки, настраиваемого начального значения) до нуля. При достижении счетчиком нуля сторожевой таймер формирует сигнал сброса, который перезагружает систему. Для предотвращения формирования сброса от IWDG необходимо заблаговременно перезагрузить таймер записью ключевого значения 0xAAAA в регистр IWDG_KR – после чего таймер снова начнет обратный отсчет от установленного значения. Тем самым и обеспечивается контроль над зависанием программы в определённых точках, где такое возможно (не перезагрузив IWDG вовремя, система сбросится). Сброс системы от независимого сторожевого таймера сбрасывает IWDG. Любой другой системный сброс (например, внешний сброс) не влияет на настройки и работу таймера IWDG.

Блок-схема независимого сторожевого таймера приведена ниже (см. рисунок 39-1).

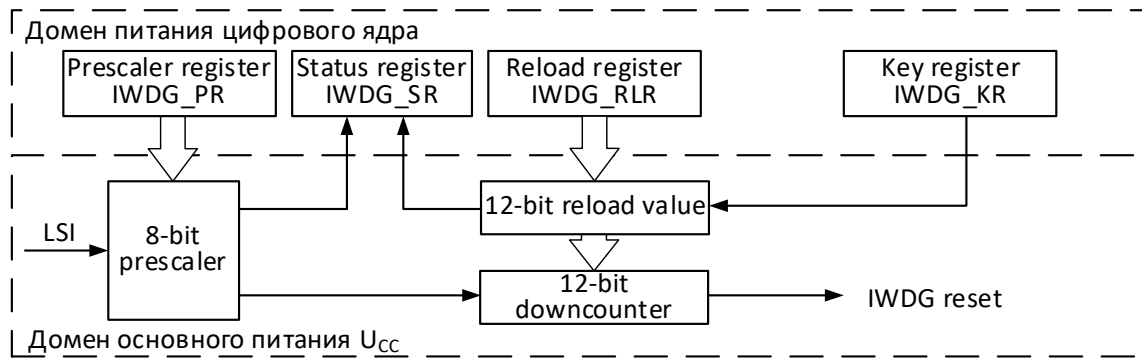


Рисунок 39-1– Блок-схема независимого сторожевого таймера

Структурно IWDG состоит из двух частей. Часть с регистрами управления расположена в домене питания цифрового ядра, доступна по APB, тактируется частотой PCLK и сбрасывается по всем типам сброса. Часть непосредственно с независимым сторожевым таймером расположена в домене основного питания микросхемы U_{CC} и тактируется от генератора LSI – поэтому для фактической записи в регистры IWDG в домене основного питания U_{CC} генератор LSI должен находиться в рабочем режиме. Часть сторожевого таймера в домене основного питания U_{CC} сбрасывается только от сброса, формируемого IWDG – после сброса от IWDG сторожевой таймер останавливает свою работу.

Период счета сторожевого таймера от разрешения работы до формирования сброса является функцией от основания счета (значения перезагрузки) и поделенной на значение предделителя частоты LSI.

Обновление значений предделителя IWDG в домене основного питания осуществляется записью регистра IWDG_PR соответственно. Обновление основания счета (значение, от которого таймер при старте начинает обратный отсчет) IWDG в домене основного питания осуществляется значением регистра IWDG_RLR записью ключевого значения 0xAAAA в регистр ключа. Для возможности записи в регистры IWDG_PR и IWDG_RLR предварительно в регистр ключа должно быть записано ключевое значение 0x5555 для разрешения доступа по записи в них. При записи в регистр IWDG_PR значения предделителя и ключа 0xAAAA в регистр IWDG_KR формируются запросы на обновление соответствующих параметров (предделителя и основания счета), которые передаются IWDG в домене основного питания. Запросы на обновления не захватываются (не сохраняются) частью в домене основного питания, то есть при любом типе сброса запросы на обновление от IWDG в домене питания ядра сбрасываются. Наличие запроса обновления от IWDG в домене питания ядра и/или информация о процессе фактического обновления на стороне IWDG в домене основного питания транслируются флагами PVU (для предделителя) и RVU (для основания счета). Фактическое обновление значений (со сбросом соответствующих флагов после обновления) осуществляется только при разрешенной работе IWDG и наличии частоты LSI. При фактическом обновлении предделителя и основания счета недоступна запись в регистры IWDG_PR и IWDG_RLR соответственно. После фактического обновления снимаются запросы обновления со стороны IWDG в домене питания ядра U_{CC}.

Пример работы с блоком IWDG:

1. Включение генератора LSI и ожидание его выхода в рабочий режим.
2. Разрешение доступа к регистрам IWDG_PR и IWDG_RLR записью значения 0x5555 в регистр IWDG_KR.
3. Запись значения предделителя в регистр IWDG_PR.
4. Разрешение работы IWDG записью значения 0xCCCC в регистр IWDG_KR.
5. Ожидание обновления значения предделителя (сброса флага PVU в 0).
6. Разрешение доступа к регистрам IWDG_PR и IWDG_RLR записью значения 0x5555 в регистр IWDG_KR.
7. Проверка флага RVU = 0 перед записью в IWDG_RLR.
8. Запись значения перезагрузки в регистр IWDG_RLR.
9. Периодическая перезагрузка сторожевого таймера в процессе выполнения программы значением регистра IWDG_RLR с помощью записи ключевого значения 0xAAAA в регистр IWDG_KR.

39.1.1.1 Регистр ключа IWDG_KR

Таблица 39-3 – Регистр IWDG_KR

Номер	31...16	15...0
Доступ	U	W
Сброс	0	0
	-	KEY[15:0]

Таблица 39-4 – Описание бит регистра IWDG_KR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...16	-	Зарезервировано
15...0	KEY[15:0]	Значение ключа (только запись, читается 0x0000). При записи: – 0xAAAA – перезагрузка значения таймера значением регистра IWDG_RLR. Значение должно периодически программно записываться при разрешенной работе IWDG, в противном случае сторожевой таймер генерирует сброс, если таймер достиг значения нуля. – 0x5555 – разрешение доступа по записи к регистрам IWDG_PR и IWDG_RLR. Если после разрешения доступа в регистр ключа записывается другое ключевое значение, доступ к регистрам IWDG_PR и IWDG_RLR запрещается. – 0xCCCC – разрешение работы сторожевого таймера (если работа сторожевого таймера не разрешена; работа таймера останавливается аппаратно сбросом от IWDG)

39.1.1.2 Делитель частоты сторожевого таймера IWDG_PR

Таблица 39-5 – Регистр IWDG_PR

Номер	31...3	2...0
Доступ	U	R/W
Сброс	0	0
	-	PR[2:0]

Таблица 39-6 – Описание бит регистра IWDG_PR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...3	-	Зарезервировано
2...0	PR[2:0]	Делитель частоты сторожевого таймера: 000 – делитель на 4: LSI/4 001 – делитель на 8: LSI/8 010 – делитель на 16: LSI/16 011 – делитель на 32: LSI/32 100 – делитель на 64: LSI/64 101 – делитель на 128: LSI/128 110 – делитель на 256: LSI/256 111 – делитель на 256: LSI/256 Для возможности записи в регистр должен быть разрешен доступ по записи с помощью ключевого значения 0x5555 в регистре IWDG_KR. При записи в регистр формируется запрос на обновление значения делителя частоты в часть IWDG в домене основного питания (см. рисунок 39-1). Запись в регистр IWDG_PR недоступна при фактической записи предделителя в часть IWDG в домене основного питания U_{CC} (см. рисунок 39-1). При чтении возвращает записанное в регистр значение, а не значение предделителя IWDG в домене основного питания

39.1.1.3 Регистр основания счета сторожевого таймера IWDG_RLR

Таблица 39-7 – Регистр IWDG_RLR

Номер	31...12	11...0
Доступ	U	R/W
Сброс	0	0xFFFF
	-	RLR[11:0]

Таблица 39-8 – Описание бит регистра IWDG_RLR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...12	-	Зарезервировано

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
11...0	RLR[11:0]	Значение перезагрузки (основания счета) сторожевого таймера. Сторожевой таймер декрементируется, начиная с этого значения. Определяет значение, загружаемое в сторожевой таймер при записи значения 0хАААА в регистр IWDG_KR. Для возможности записи в регистр должен быть разрешен доступ по записи с помощью ключевого значения 0х5555 в регистре IWDG_KR. Запись в регистр IWDG_RLR недоступна при фактической записи основания счета в часть IWDG в домене основного питания U_{CC} (см. рисунок 39-1). При чтении возвращает записанное в регистр значение, а не значение перезагрузки IWDG в домене основного питания

39.1.1.4 Регистр статуса сторожевого таймера IWDG_SR

Таблица 39-9 – Регистр IWDG_SR

Номер	31...2	1	0
Доступ	U	R	R
Сброс	0	0	0
	-	RVU	PVU

Таблица 39-10 – Описание бит регистра IWDG_SR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...2	-	Зарезервировано
1	RVU	Флаг обновления значения сторожевого таймера Устанавливается аппаратно и служит признаком того, что обновляется значение сторожевого таймера IWDG в домене основного питания U_{CC} из регистра перезагрузки IWDG_RLR (см. рисунок 39-1). Работа IWDG разрешена и LSI работает: Отображает запрос от регистров управления на обновление значения сторожевого таймера значением регистра IWDG_RLR и/или фактическое обновление значения сторожевого таймера в домене основного питания U_{CC}. Сбрасывается при завершении фактического обновления значения перезагрузки. Значение регистра перезагрузки IWDG_RLR может быть обновлено только, если этот бит равен нулю. Работа IWDG не разрешена или LSI не работает: Отображает запрос от регистров управления на обновление значения сторожевого таймера в домене основного питания U_{CC} значением регистра IWDG_RLR. Сбрасывается любым типом сброса. Значение регистра перезагрузки IWDG_RLR может быть

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
		обновлено независимо от состояния этого бита – после запуска IWDG будет обновлен последним записанным в IWDG_RLR значением
0	PVU	Флаг обновления делителя частоты сторожевого таймера Устанавливается аппаратно и служит признаком того, что обновляется значение делителя частоты IWDG в домене основного питания U_{CC} из регистра предделителя IWDG_PR (см. рисунок 39-1). Работа IWDG разрешена и LSI работает: Отображает запрос от регистров управления на обновление делителя частоты сторожевого таймера значением регистра IWDG_PR и/или фактическое обновление значения делителя в домене основного питания U_{CC}. Сбрасывается при завершении фактического обновления делителя. Значение регистра предделителя IWDG_PR может быть обновлено только, если этот бит равен нулю. Работа IWDG не разрешена или LSI не работает: Отображает запрос от регистров управления на обновление значения сторожевого таймера в домене основного питания U_{CC} значением регистра IWDG_PR. Сбрасывается любым типом сброса. Значение регистра перезагрузки IWDG_PR может быть обновлено независимо от состояния этого бита – после запуска IWDG будет обновлен последним записанным в IWDG_PR значением

39.1.2 Блок сторожевого таймера WWDG

39.1.2.1 Регистр управления WWDG_CR

Таблица 39-11 – Регистр WWDG_CR

Номер	31...8	7	6...0
Доступ	U	R/W	R/W
Сброс	0	0	1
	-	WDGA	T[6:0]

Таблица 39-12 – Описание бит регистра WWDG_CR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...8	-	Зарезервировано
7	WDGA	Бит активации. Этот бит устанавливается программно и очищается только аппаратно при сбросе. Когда WDGA=1, сторожевой таймер может генерировать сброс. 0 – сторожевой таймер отключен; 1 – сторожевой таймер включен

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
6...0	T[6:0]	Значение семиразрядного счетчика (от старших разрядов к младшим) Эти биты содержат значение сторожевого таймера, который декрементируется каждые 4096×2^{WDGTB} циклов частоты PCLK периферийной шины APB

39.1.2.2 Регистр конфигурации WWDG_CFR

Таблица 39-13 – Регистр WWDG_CFR

Номер	31...10	9	8, 7	6...0
Доступ	U	R/S	R/W	R/W
Сброс	0	0	0	1
	-	EWI	WDGTB[1:0]	W[6:0]

Таблица 39-14 – Описание бит регистра WWDG_CFR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...10	-	Зарезервировано
9	EWI	Раннее предупреждающее прерывание. Если бит установлен, то разрешается генерация прерывания при достижении сторожевым таймером значения 40h. Прерывание запрещается только аппаратным сбросом
8, 7	WDGTB[1:0]	Делитель частоты сторожевого таймера: 00 – частота таймера (PCLK / 4096) /1; 01 – частота таймера (PCLK / 4096) /2; 10 – частота таймера (PCLK / 4096) /4; 11 – частота таймера (PCLK / 4096) /8
6...0	W[6:0]	Значение окна Эти биты содержат значение окна, в пределах которого возможна инициализация битов T[6:0] значением в пределах 40h-7Fh. Если происходит инициализация битов в момент T>W, то формируется сброс на выходе RESET. Если таймер достигнет значения T=3Fh, то также формируется сброс

39.1.2.3 Регистр статуса WWDG_SR

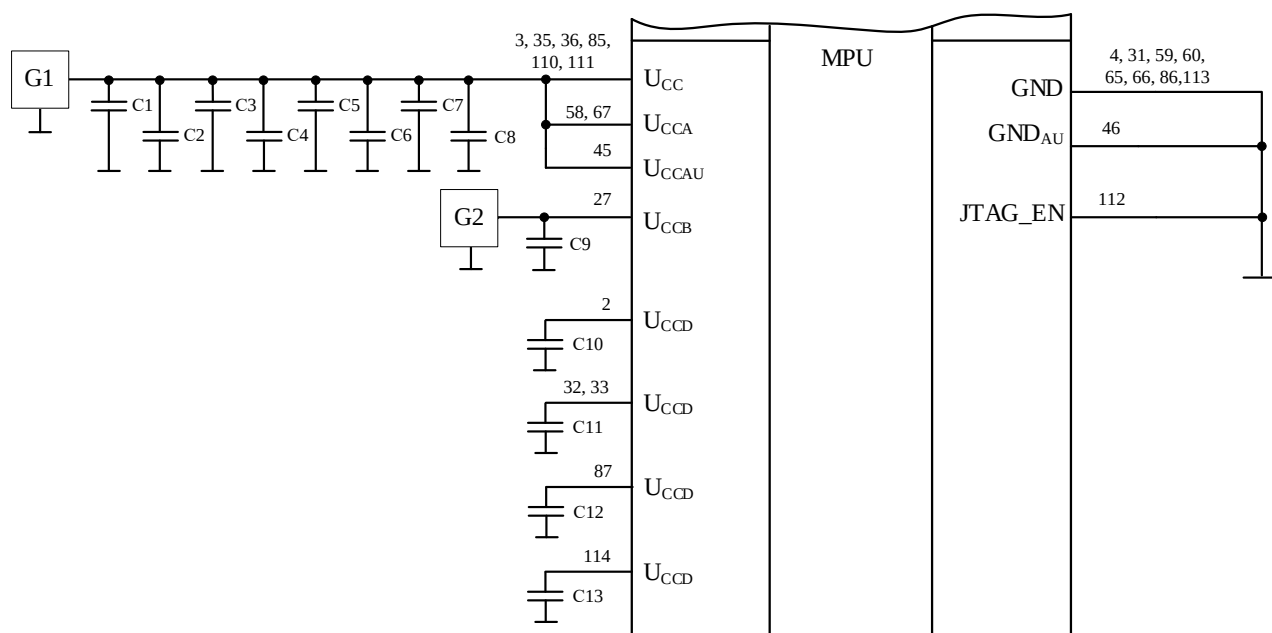
Таблица 39-15 – Регистр WWDG_SR

Номер	31...1	0
Доступ	U	R/C
Сброс	0	0
	-	EWIF

Таблица 39-16 – Описание бит регистра WWDG_SR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...1	-	Зарезервировано
0	EWIF	Флаг раннего предупреждающего прерывания Этот бит устанавливается аппаратно, когда сторожевой таймер достигает значения 40h. Бит очищается программно записью нуля. Запись единицы не влияет. Этот бит также устанавливается, если прерывание запрещено EWI=0

40 Типовая схема включения



C1 – конденсатор емкостью $22 \text{ мкФ} \pm 5 \%$, 16 В;

C2 – C13 – конденсаторы емкостью $0,1 \text{ мкФ} \pm 5 \%$, 16 В;

G1 – источник напряжения питания 3,0 – 3,6 В;

G2 – источник напряжения питания 1,8 – 3,6 В;

MPU – включаемая микросхема.

Рисунок 40–1 – Типовая схема включения микросхемы

41 Типовые зависимости

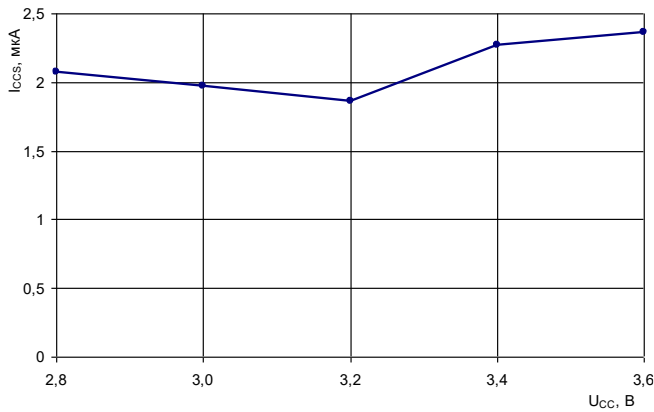


Рисунок 41-1 – Зависимость статического тока потребления в режиме покоя (регулятор напряжения выключен) от напряжения источника питания, при $T=25\text{ °C}$

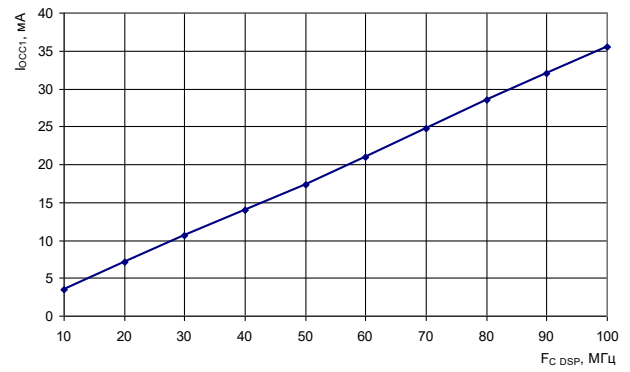


Рисунок 41-2 – Зависимость динамического тока потребления модуля ПЦОС от частоты следования импульсов тактовых сигналов, при $T=25\text{ °C}$, $U_{CC}=3,6\text{ В}$, $f_{C\_RISC}=70\text{ МГц}$

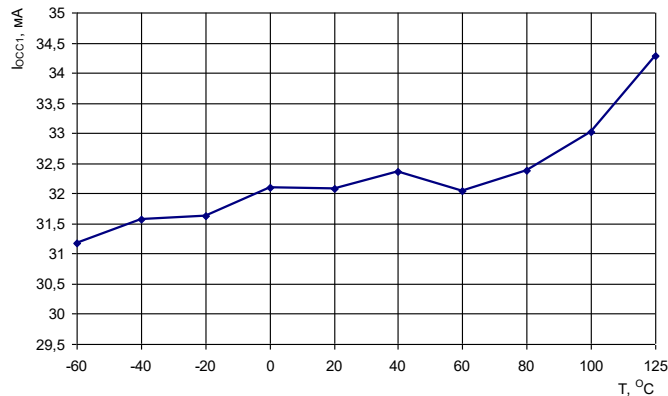


Рисунок 41-3 – Зависимость динамического тока потребления модуля ПЦОС от температуры, при $U_{CC}=3,6\text{ В}$, $f_{C\_DSP}=100\text{ МГц}$, $f_{C\_RISC}=0\text{ МГц}$

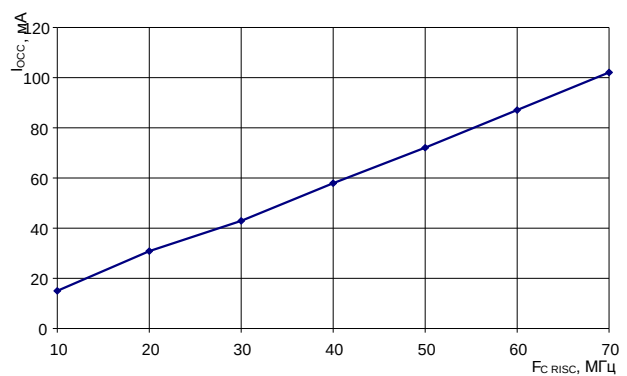


Рисунок 41-4 – Зависимость динамического тока потребления процессора RISC от частоты следования импульсов тактовых сигналов, при $T=25\text{ °C}$, $U_{CC}=3,6\text{ В}$, $f_{C\_DSP}=100\text{ МГц}$

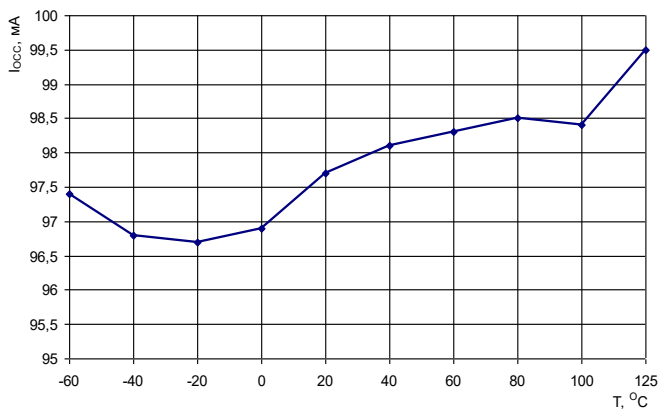


Рисунок 41-5 – Зависимость динамического тока потребления от температуры, при $U_{CC}=3,6\text{ В}$, $f_{C\_DSP}=0\text{ МГц}$, $f_{C\_RISC}=70\text{ МГц}$

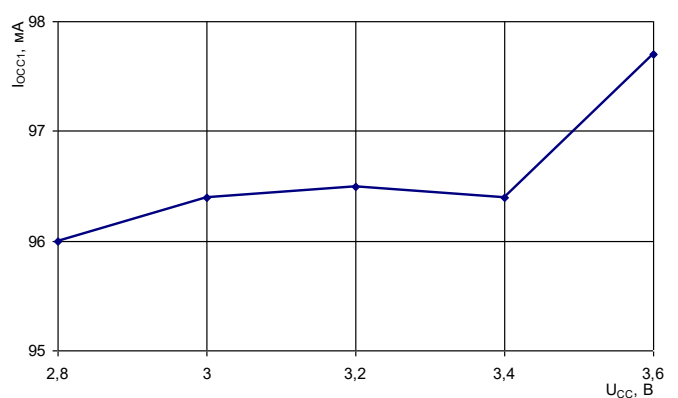


Рисунок 41-6 – Зависимость динамического тока потребления модуля ПЦОС от напряжения источника питания, при $T=25\text{ °C}$, $f_{C\_DSP}=100\text{ МГц}$, $f_{C\_RISC}=0\text{ МГц}$

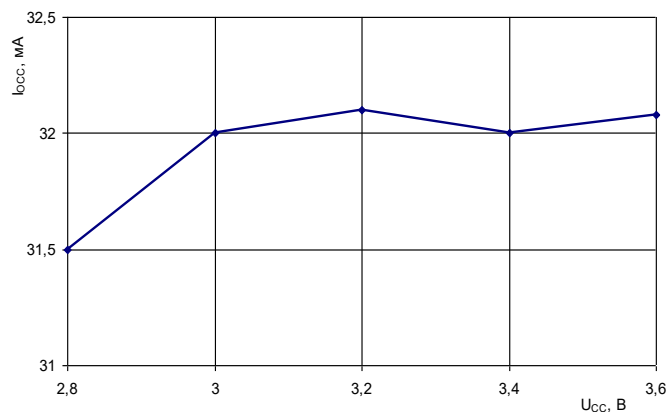


Рисунок 41–7 – Зависимость динамического тока потребления от напряжения источника питания, при $T=25\text{ }^{\circ}\text{C}$, $f_{C\_DSP}=0\text{ МГц}$, $f_{C\_RISC}=70\text{ МГц}$

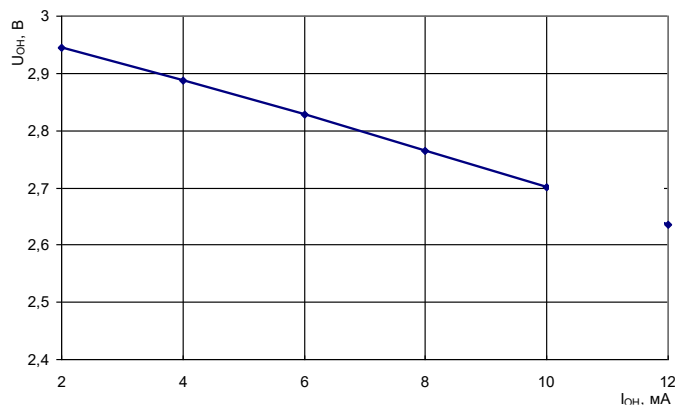


Рисунок 41–8 – Зависимость выходного напряжения высокого уровня от выходного тока нагрузки, при $T=25\text{ }^{\circ}\text{C}$, $U_{CC}=3,0\text{ В}$

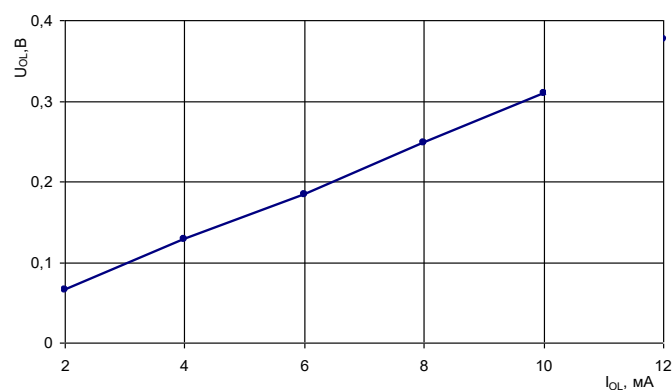


Рисунок 41–9 – Зависимость выходного напряжения низкого уровня от выходного тока нагрузки, при $T=25\text{ }^{\circ}\text{C}$, $U_{CC}=3,0\text{ В}$

42 Электрические параметры микросхемы

Таблица 42-1 – Электрические параметры микросхемы

Наименование параметра, единица измерения, режим измерения	Буквенное обозначение параметра	Норма параметра		Температура среды, °C
		не менее	не более	
Выходное напряжение низкого уровня, В, на выводах: PA, PB, PC, PD, PE, PF, STANDBY, DN, DP	U_{OL}	–	0,4	25, 85, – 60
Выходное напряжение высокого уровня, В, на выводах: PA, PB, PC, PD, PE, PF, STANDBY, DN, DP	U_{OH}	2,4	–	
Уровень напряжения питания для срабатывания схемы формирования общего сброса, В	U_{BOR}	1,8	2,1	
Входной ток утечки высокого уровня, мкА, на выводах: PA, PB, PC, PD, PE, PF, DN, DP, RESET, WAKEUP	I_{ILH1}	– 1	1	
Входной ток утечки высокого уровня, мкА, на выводе: OSC_IN при: BYPASS=1	I_{ILH2}	– 40	40	
Входной ток утечки низкого уровня, мкА, на выводах: PA, PB, PC, PD, PE, PF, RESET, WAKEUP, DN, DP	I_{ILL1}	– 1	1	
Входной ток утечки низкого уровня, мкА, на выводе: OSC_IN при: BYPASS=1	I_{ILL2}	– 1	1	
Статический ток потребления в режиме покоя (регулятор напряжения выключен), мкА	I_{CCS}	–	10	25, – 60
		–	20	85
Динамический ток потребления, мА, при: f_{C_DSP} = 0 МГц, f_{C_RISC} = 100 МГц	I_{OCC1}	–	180	25, 85, – 60
Динамический ток потребления, мА, при: f_{C_DSP} = 100 МГц, f_{C_RISC} = 100 МГц	I_{OCC2}	–	300	
Частота работы LSI RC-генератора, кГц	f_{O_LSI}	10	60	
Частота работы HSI RC-генератора, МГц	f_{O_HSI}	6	10	
Выходная частота PLL, МГц максимальная минимальная	f_{O_PLL}	100	–	
		–	6	
Время выхода из режима sleep, мкс	t_{ON}	–	100	

Наименование параметра, единица измерения, режим измерения	Буквенное обозначение параметра	Норма параметра		Температура среды, °C
		не менее	не более	
Параметры АЦП				
Разрядность АЦП	E _{NADC}	12	–	25, 85, – 60
Дифференциальная нелинейность АЦП, единица младшего разряда	E _{DLADC}	– 1	2	
Интегральная нелинейность АЦП, единица младшего разряда	E _{ILADC}	– 3	3	
Ошибка смещения АЦП, единица младшего разряда	E _{OFFADC}	– 6	6	
Параметры ЦАП				
Разрядность ЦАП	E _{NDAC}	12	–	25, 85, – 60
Дифференциальная нелинейность ЦАП, единица младшего разряда	E _{DLDAC}	– 1	2	
Интегральная нелинейность ЦАП, единица младшего разряда	E _{ILDAC}	– 6	6	
Ошибка смещения ЦАП, мВ	E _{OFFDAC}	– 40	40	
Минимальное выходное напряжение ЦАП, В при R _L =10 кОм	U _{O_DAC} min	–	0,08	
Максимальное выходное напряжение ЦАП, В при R _L = 10 кОм	U _{O_DAC} max	U _{REF(DAC)} - 0,08	–	
Параметры компаратора				
Время включения компаратора, мкс	t _{ON_C}	–	100	25, 85, – 60
Время задержки переключения компаратора, нс	t _{d_C}	–	400	
Параметры аудиокодека				
Кoeffициент нелинейных искажений + шум ЦАП, дБ, при: Синус, частота 1 кГц, U _I *= 0 дБ, DAGAIN = – 1 дБ	T _{DAC_} THD+N	68	–	25, 85, – 60
Кoeffициент нелинейных искажений + шум АЦП, дБ (для диф. входов INP1/INM1 и INP2/INM2) при: Синус, частота 1 кГц, U _I ** = 0 дБ ADGAIN = 0 дБ, INBG = 0 дБ	T _{ADC_} THD+N	80	–	
U _I **= – 6 дБ ADGAIN= 0 дБ, INBG= 6 дБ		78	–	
U _I **= – 12 дБ ADGAIN= 0 дБ, INBG= 12 дБ		74	–	
U _I **= – 24 дБ ADGAIN= 0 дБ, INBG= 24 дБ		64	–	

Наименование параметра, единица измерения, режим измерения	Буквенное обозначение параметра	Норма параметра		Температура среды, °C
		не менее	не более	
Ошибка коэффициента входного предусилителя АЦП, дБ (для дифференциальных входов INP1/INM1 и INP2/INM2)	GAIN ERRADC	– 0,5	0,5	25, 85, – 60
Максимальная амплитуда выходного дифференциального сигнала ЦАП на выходах OUTP1 и OUTM1, В, при DAGAINT = – 1 дБ, U _I *= 0 дБ	U _{OAUDAC}	1,8	2,1	
Напряжение нуля выходного сигнала ЦАП, на выходах OUTP1 и OUTM1, В, GAINT = 0, MUTE1 = 1, ODRCT1 = 1	U _{OAUDAC0}	1,3	1,6	
Выходное напряжение BIAS (на выходе BIAS), В	U _{OBIAS}	1,3	1,6	
* Полная шкала сигнала для ЦАП U_I = 0 дБ соответствует сигналу с кодами отсчетов в диапазоне от минус 32 768 до 32 767. ** Полная шкала сигнала для АЦП U_I = 0 дБ соответствует сигналу с максимальной амплитудой U_{IAUADC} (на входах INP1/INM1 и INP2/INM2). При этом полной шкале сигнала для АЦП U_I = 0 дБ будет соответствовать появление отчетов с кодами от минус 22 000 до 22 000 для дифференциальных входов INP1/INM1 и INP2/INM2. Полная шкала сигнала для АЦП U_I = – 6 дБ соответствует сигналу с максимальной амплитудой (U_{IAUADC}/2) (на входах INP1/INM1 и INP2/INM2). Полная шкала сигнала для АЦП U_I = – 12 дБ соответствует сигналу с максимальной амплитудой (U_{IAUADC}/4) (на входах INP1/INM1 и INP2/INM2). Полная шкала сигнала для АЦП U_I = – 24 дБ соответствует сигналу с максимальной амплитудой (U_{IAUADC}/16) (на входах INP1/INM1 и INP2/INM2)				

Микросхемы устойчивы к воздействию статического электричества с потенциалом не менее 2 000 В.

43 Предельные и предельно-допустимые режимы работы

Таблица 43-1 – Предельные и предельно-допустимые режимы эксплуатации микросхемы

Наименование параметра, единица измерения	Буквенное обозначение параметра	Норма параметра			
		Предельно- допустимый режим		Предельный режим	
		не менее	не более	не менее	не более
Напряжение источника питания, В	U_{CC}	3,0	3,6	–	4,0
Напряжение источника* питания АЦП и ЦАП, В	U_{CCA}	3,0	3,6	–	4,0
Напряжение источника* питания аудиокодека, В	U_{CCAU}	3,0	3,6	–	4,0
Напряжение источника питания батареиногo домена, В	U_{CCB}	1,8	3,6	–	4,0
Входное напряжение низкого уровня, В, (при работе в цифровом режиме) на выводах: PA, PB, PC, PD, PE, PF, RESET, WAKEUP, DN, DP на выводе: OSC_IN при: BYPASS=1	U_{IL}	0	0,8	– 0,3	–
		0	$0,2 \cdot U_{CC}$	– 0,3	–
Входное напряжение высокого уровня, В, на выводах: PD, PE (0-10), DN, DP на выводах: PA, PB, PC, PE (11-15), PF, RESET, WAKEUP на выводе: OSC_IN при: BYPASS=1	U_{IH}	2,0	U_{CC}	–	4,0
		2,0	5,25	–	5,3
		$0,8 \cdot U_{CC}$	U_{CC}	–	4,0
Выходной ток низкого уровня, мА, (при работе в цифровом режиме) на выводах: PA, PB, PC, PD, PE(1-5, 8-12), PF, STANDBY на выводах: PE 6, 7 на выводах: DN, DP	I_{OL}	– 6	–	– 10	–
		– 3	–	– 10	–
		– 6	–	– 40	–
Выходной ток высокого уровня, мА, на выводах: PA, PB, PC, PD, PE (1-5, 8-12), PF, STANDBY на выводах: PE 6, 7 на выводах: DN, DP	I_{OH}	–	6	–	10
		–	3	–	10
		–	6	–	40
Частота следования импульсов тактовых сигналов процессора RISC, МГц	f_{C_RISC}	–	100	–	–
Частота следования импульсов тактовых сигналов модуля ПЦОС, МГц	f_{C_DSP}	–	100	–	–

Наименование параметра, единица измерения	Буквенное обозначение параметра	Норма параметра			
		Предельно- допустимый режим		Предельный режим	
		не менее	не более	не менее	не более
Частота следования импульсов тактовых сигналов HSE, МГц при: BYPASS=0 при: BYPASS=1	f _{C_HSE}	2	16	—	—
		0	100	—	—
Частота следования импульсов тактовых сигналов LSE, кГц при: BYPASS=0 при: BYPASS=1	f _{C_LSE}	32	33	—	—
		0	1 000	—	—
Частота следования импульсов тактовых сигналов PLL, МГц	f _{C_PLL}	6	16	—	—
Параметры ЦАП					
Напряжение верхней границы опоры ЦАП на выводе: DACx_REF при: Cfg_M_REFx=1	U _{REF(DACx)}	2,4	U _{CCA}	—	—
Резистивная нагрузка ЦАП, кОм при которой гарантируются параметры	R _{LOAD}	10	—	—	—
Емкостная нагрузка ЦАП, пФ при которой гарантируются параметры	C _{LOAD}	—	100	—	—
Параметры АЦП					
Напряжение нижней границы внешнего опорного источника АЦП, В при: ADC1_Cfg_M_REF=1 или ADC2_Cfg_M_REF=1	U _{ADC1_REF-}	0	U _{CCA-3,0}	—	4,0
Напряжение верхней границы внешнего опорного источника АЦП, В при: ADC1_Cfg_M_REF=1 или ADC2_Cfg_M_REF=1	U _{ADC0_REF+}	3,0	U _{CCA}	—	4,0
Диапазон напряжения внешнего опорного источника АЦП, В при: U _{REF(ADC)} = U _{ADC0_REF+} – U _{ADC1_REF-} ADC1_Cfg_M_REF=1 или ADC2_Cfg_M_REF=1	U _{REF(ADC)}	3,0	U _{CCA}	—	—
Диапазон напряжения на входе АЦП, В**	U _{AIN}	U _{ADC1_REF-}	U _{ADC0_REF+}	– 0,3	4,0
Частота следования импульсов тактовых сигналов АЦП, МГц	f _{C_ADC}	—	14	—	—
Параметры аудиокодека					
Напряжения нуля входного сигнала АЦП, В, на выводах: INP1, INM1, INP2, INM2	U _{IAUADC0}	1,3	1,6	—	—

Наименование параметра, единица измерения	Буквенное обозначение параметра	Норма параметра			
		Предельно- допустимый режим		Предельный режим	
		не менее	не более	не менее	не более
Максимальная амплитуда входного дифференциального сигнала АЦП, В, на выводах: INP1, INM1, INP2, INM2	U _{IAUADC}	—	1,05	—	—
Напряжение на входе АЦП, В, на выводах: INP1, INM1, INP2, INM2, MICIN	U _{IADC}	—	—	–0,3	U _{CCA} + 0,3
Частота дискретизации аудиокодека, кГц	f _{C_AUC}	—	24	—	—
Резистивная нагрузка ЦАП, Ом	R _{LDAC}	600	—	—	—
Резистивная нагрузка BIAS, Ом	R _{LBIAS}	600	—	—	—
Длительность фронта нарастания/спада входных сигналов, нс при: BYPASS=1	τ _r τ _f	—	5	—	—
Емкость нагрузки, пФ на выводах: PA,PB,PC,PD,PE,PF, STANDBY	C _L	—	30	—	—
Число циклов записи/стирания данных, при: T=85 °C	N _{PR}	10 000	—	—	—
Время хранения информации, лет, при: T=25 °C при: T=85 °C	t _{GS}	25	—	—	—
		10	—	—	—
Примечание – Не допускается одновременное воздействие двух и более предельных режимов					
* Допускается использование отдельного источника для питания АЦП, ЦАП и аудиокодека, но при этом его выходное напряжение не должно отличаться от U _{CC} более чем на ± 0,2 В.					
** При использовании внутреннего опорного напряжения U _{ADC1_REF-} = GND, U _{ADC0_REF+} = U _{CCA}					

44 Справочные данные

Значение собственной резонансной частоты не менее 3800 Гц.

Предельная температура р–п перехода кристалла 150 °С.

Тепловое сопротивление кристалл-корпус – не более 6,40 °С/Вт.

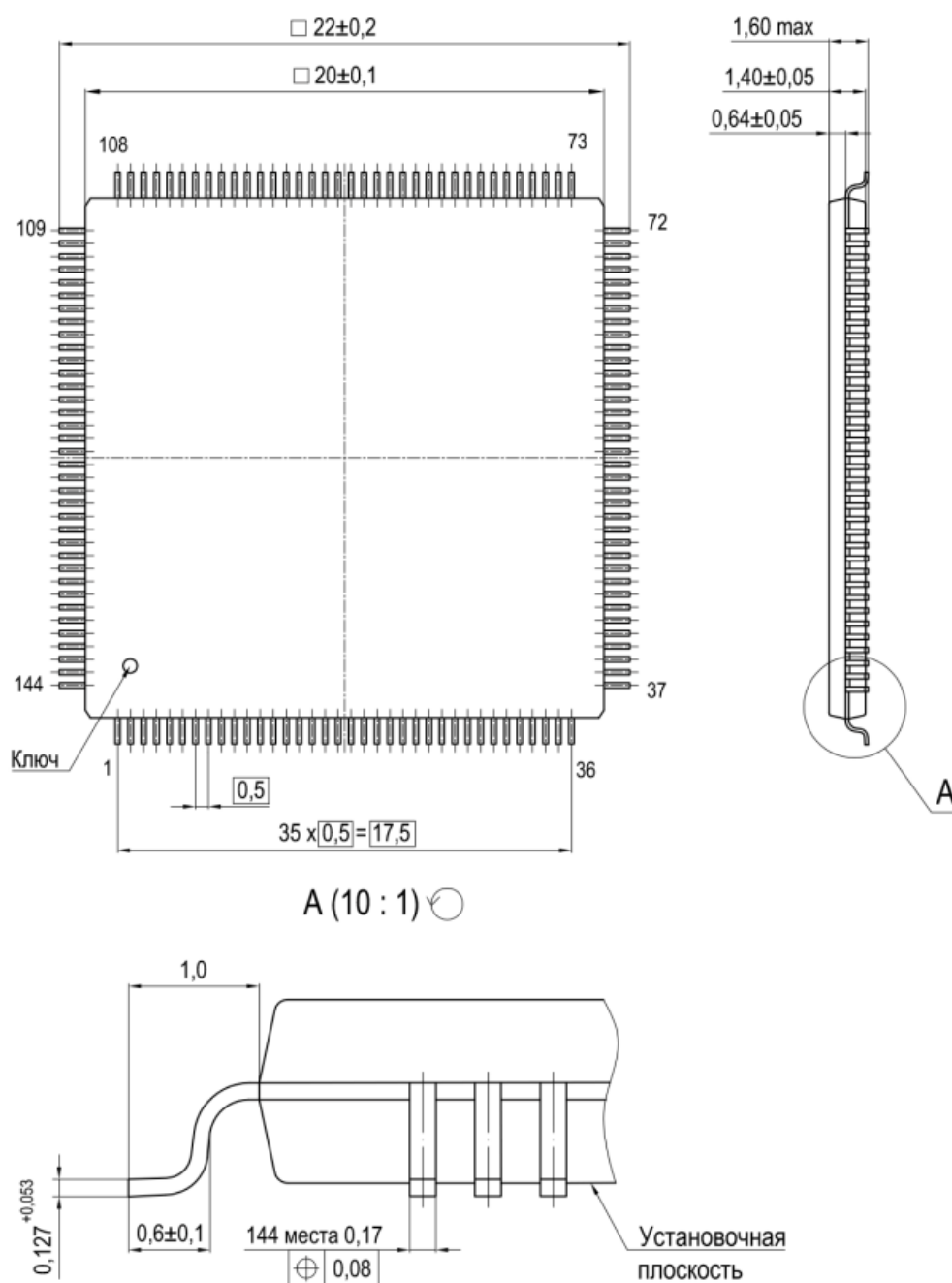
Таблица 44-1 – Справочные параметры

Наименование параметра, единица измерения, режим измерения	Буквенное обозначение параметра	Норма параметра		Температура среды, °C
		не менее	не более	
Динамический ток потребления модуля ПЦОС, мА, при: f _{C_DSP} = 100 МГц, f _{C_RISC} = 0 МГц	I _{OCC1}	–	95	25, 85, – 60
Выходное напряжение регулятора LDO, В при: U _{CC} = 3,6 В, I _{OL} = 200 мА	U _{O_LDO}	1,62	1,98	
Ток потребления батарейного домена, мкА, при: U _{CC} = U _{CCA} = 0 В	I _{CC_B}		5	
Гистерезис портов ввода/вывода, мВ, на выводах: PA-PF при: ModeRX = 0 ModeRX = 1	ΔU _{TH(PA-PF)}	100 200	400 500	
Длительность фронта переключения выходных сигналов, нс, на выводе: STANDBY, C _I = 30 пФ	t _{W(STANDBY)}	–	10	
на выводах: PA – PF при: PowerTX=00, C _I = 50 пФ PowerTX=01, C _I = 50 пФ PowerTX=10, C _I = 50 пФ PowerTX=11, C _I = 50 пФ PowerTX=11, C _I = 30 пФ	t _{W(PA-PF)}	– – – – –	10 100 20 10 5	
на выводах: DN, DP при: Full Speed, C _I = 50 пФ Low Speed, C _I = 600 пФ		– –	15 300	
Компаратор				
Опорное напряжение компаратора при: CVRSS = 1 на выводах: PE4_COMP_REF+ и PE5_COMP_REF-	U _{REFCOMP}	2,4	U _{CC}	25, 85, – 60
Напряжение смещения компаратора, мВ, при: U _{CC} = 3,6 В	U _{IO_C}	– 0,5	0,5	
Гистерезис компаратора, мВ	ΔU _{TH_C}	8	12	
Напряжение внутреннего опорного источника компаратора, В	U _{REF_C}	1,17	1,23	
Тактовые частоты и генераторы				

Наименование параметра, единица измерения, режим измерения	Буквенное обозначение параметра	Норма параметра		Температура среды, °C
		не менее	не более	
Время установления сигнала HSIRDY относительно HSION, мкс	t _{SU(HSI)}	—	1	25, 85, – 60
Время установления сигнала LSIRDY относительно LSION, мс	t _{SU(LSI)}	—	80	
Время установления сигнала HSERDY относительно HSEON, мкс	t _{SU(HSE)}	—	2048/f _{C_HSE}	
Время установления сигнала LSERDY относительно LSEON, мкс	t _{SU(LSE)}	—	4096/ f _{C_LSE}	
Время установления сигнала PLLRDY относительно PLLON, мкс	t _{SU(PLL)}	—	100	
Длительность сигнала сброса, мкс	t _{W(RESET)}	20	—	
Время запуска после сброса по POR, мс	t _{POR}	—	6	
АЦП				
Время выборки заряда АЦП, нс	t _{A_ADC}	—	4 • f _{C_ADC}	25, 85, – 60
Время преобразования АЦП, нс	t _{AO_ADC}	—	28 • f _{C_ADC}	
Ток потребления по входу внешней верхней границы опоры АЦП, мкА при: ADC1_Cfg_M_REF=1 или ADC2_Cfg_M_REF=1	I _{ADC0_VREF+}	—	50	
Ток потребления по входу внешней нижней опоры АЦП, мкА при: ADC1_Cfg_M_REF=1 или ADC2_Cfg_M_REF=1	I _{ADC0_VREF-}	– 50	—	
Ток потребления по питанию АЦП, мА при: U _{CCA} =3,6В, F _{adc} =14 МГц	I _{OCCADC}	—	3	
Минимальная частота преобразования АЦП, кГц	F _{C_ADCMIN}	10	—	
ЦАП				
Время установления сигнала ЦАП, мкс, при: C _l = 100 пФ, R _l = 10 кОм	t _{SU(DAC)}	—	5,2	25, 85, – 60
Время включения ЦАП, мкс	t _{ON_DAC}	—	10	
Ток потребления по входу опоры, мкА при Cfg_M_REF0 = 1	I _{DAC1_VREF}	—	500	
Ток потребления по входу опоры, мкА при Cfg_M_REF1 = 1	I _{DAC2_VREF}	—	500	
Ток потребления ЦАП, мА при отсутствии нагрузки	I _{OCCDAC}	—	2	

Наименование параметра, единица измерения, режим измерения	Буквенное обозначение параметра	Норма параметра		Температура среды, °C
		не менее	не более	
USB				
Сопротивление резистора между линиями DN, DP и питанием, кОм при: D-PULLUP=1	R _{DN-UCC}	1	2	25, 85, – 60
D+PULLUP=1	R _{DP-UCC}	1	2	
Сопротивление резистора между линиями DN, DP и шиной общий, кОм при: D-PULLDOWN=1	R _{DN-GND}	10	20	
D+PULLDOWN=1	R _{DP-GND}	10	20	
Согласующее сопротивление на линиях DN, DP, Ом	R _{DN} R _{DP}	14	34	
Аудиокодек				
Коэффициент нелинейных искажений + шум АЦП, дБ (для микрофонного входа MICIN) Сигнал: Синус, частотой 1 кГц, AINSEL= 10 U _I *= 0 дБ ADGAIN= 0 дБ, INBG= 0 дБ	T _{ADC_} THD+N	63	–	25, 85, – 60
U _I *= минус 6 дБ ADGAINT= 0 дБ, INBG= 6 дБ		58	–	
U _I *= минус 12 дБ ADGAINT= 0 дБ, INBG= 12 дБ		51	–	
U _I *= минус 24 дБ ADGAINT= 0 дБ, INBG= 24 дБ		38	–	
Коэффициент усиления ошибки, дБ	T _{ERR}	– 0,5	0,5	
Коэффициент межканального усиления, дБ	T _{CH_ERR}	–	60	
Коэффициент усиления шума питания, дБ	T _{PWR_ERR}	–	60	
Ток короткого замыкания ЦАП, мА, на выводах: OUTP1, OUTM1	I _{OSDAC}	–	20	
Входной ток АЦП, мкА, на выводах INP1, INP2, INM1, INM2, MICIN	I _{IADC}	–	100	

45 Габаритный чертеж микросхемы



- 1 Размеры фасок не регламентируются.
2 Нумерация выводных площадок показана условно.

Рисунок 45–1 – Микросхема в корпусе LQFP144

46 Информация для заказа

Обозначение	Маркировка	Тип корпуса	Температурный диапазон, °C
K1901BЦ1QI	MDR32FG16S1QI	LQFP144	от – 40 до 85

Лист регистрации изменений

№ п/п	Дата	Версия	Краткое содержание изменения	№№ изменяемых листов
1	29.07.2021	2.0.0	Введена впервые	
2	22.09.2021	2.1.0	Замена термина «пластиковый» на «пластмассовый» Изменено примечание в разделе 36.1.2 Скорректированы названия и описания битов 3 и 17 в таблицах 12-4, 12-5 Исправлено название параметра «Частота следования импульсов тактовых сигналов аудиокодека» на «Частота дискретизации аудиокодека» Добавлен раздел «Логика работы прерываний контроллера NVIC.»	1 780 165 838 794
3	06.05.2024	2.2.0	Информация о массе микросхем добавлена Пункты «Микроконтроллер в режиме отладки» и «Микропроцессор в режиме отладки» добавлены Пункт 6.5.3 дополнен Таблица 6-13 - Таблица 6-18 скорректированы Пункты 7.2.1-7.2.4 скорректированы Раздел 12 – добавлена информация об условии переключения источников частоты Рисунок 13-1 скорректирован Подраздел 13.1 скорректирован и дополнен Таблица 13-1 – описание битов RTC_DIV, RTC_PRL, RTC_ALRM скорректировано Таблица 13-7 – описание битов RTC_CAL[7:0] скорректировано Таблица 13-13, Таблица 13-15, Таблица 13-17 – описание битов скорректировано Рисунок 16-2 и Рисунок 16-3 скорректированы Подраздел 16.1 Пункт 25.4.1 – п.4 дополнен Пункт 25.4.2 – п.6 дополнен Таблица 25-5 – описание битов INBG скорректировано Раздел 34 – исключена информация о функции управления модемом и организации аппаратного управления потоком данных Исправление опечаток	1 60-61 61 63-65 69-71 161 176 176 177 180 183-184 199 200 601 602 603 735-766 По тексту
4	12.03.2025	2.3.0	Заменено «IWDТ» на «IWDG», «WWDТ» на «WWDG», «EERPOМ» на «EEPROM» Раздел 17 – «2R» исправлено на «2 ^R »; Добавлено название пункта перед таблицей 7-3;	По тексту По тексту 73

			Таблица 10-1 – удалено примечание;	146
			Пункт 10.1.4 – исправлено описание TENMS, добавлена информация о нереализации калибровки системного таймера;	148
			Таблица 12-17 – исправлено «SPIx» на «SSPx», «SDIO» на «SDIO»;	170
			Раздел 12 – скорректирован;	161 – 175
			Таблица 14-4 – добавлено описание PORT RXTX[15:0];	190
			Подпункт 17.4.5.3 – скорректирован пункт 2 в нумерованном списке;	225
			Пункт 33.8.4 – исправлено описание;	734
			Подраздел 36.3 – дополнен;	782
			Таблица 24-2 скорректирован адрес RISC для регистров ERR_CLRL и ERR_CLRH;	595
			Таблица 27-59, таблица 28-26, таблица 29-14, таблица 34-19 – добавлено примечание;	639, 676, 691, 765
5	19.03.2025	2.3.1	Рисунок 12-1 – исправлено DSP_C2_SEL[1:0] на DSP_C2_SEL, DSP_C3_SEL[3:0] на DSP_C3_SEL, USB_C2_SEL[1:0] на USB_C2_SEL	161