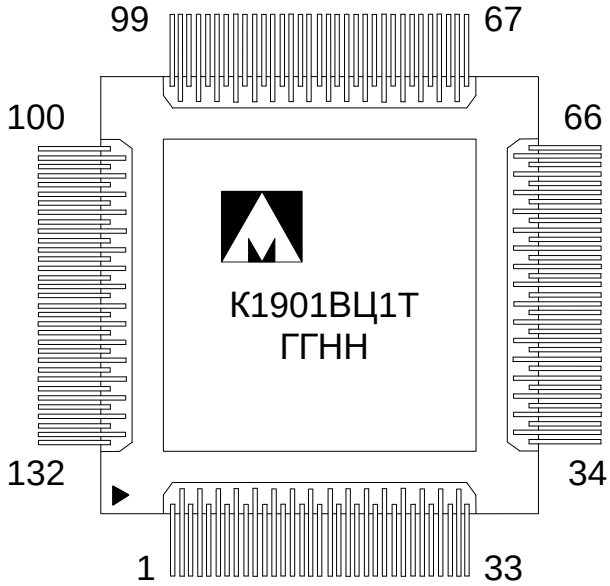
Двухъядерный процессор цифровой обработки сигналов с фиксированной запятой

1901ВЦ1Т, К1901ВЦ1Т, К1901ВЦ1ТК, К1901ВЦ1Н4



ГГ – год выпуска
НН – неделя выпуска

Основные параметры микросхемы

- 32 разрядная RISC архитектура
- 16 разрядная DSP архитектура
- 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 28149-89
- Компаратор
- Напряжение источника питания, (3,3 ± 10%) В

- Температурный диапазон:

Обозначение	Диапазон
1901ВЦ1Т	минус 60 – 85 °С
К1901ВЦ1Т	минус 60 – 85 °С
К1901ВЦ1ТК	0 – 70 °С

Тип корпуса:

– 132-выводной металлокерамический корпус 4229.132-3;
– микросхемы К1901ВЦ1Н4 поставляются в бескорпусном исполнении.

Особенности ядра и памяти RISC:

- 32-битное RISC ядро, 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

Особенности ядра и памяти DSP:

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

Особенности переферии и режима отладки:

- контроллер прямого доступа в память с функциями передачи Периферия-Память, Память-Память;
- контроллер USB интерфейса с функциями работы Device и Host;
- контроллеры интерфейсов UART, SPI, I2;
- до 96 пользовательский линий ввода-вывода;
- контроллер блок аппаратной поддержки шифрования ГОСТ 28147-89;
- SDIO интерфейса;
- последовательные отладочные интерфейсы SWD и JTAG.

Особенности аналоговых модулей:

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

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 RISC ядра;
- встроенный умножитель тактовой частоты PLL для DSP ядра;
- встроенный умножитель тактовой частоты PLL для контроллера USB
- встроенный умножитель тактовой частоты PLL для аудио кодека.

Специализированные особенности:

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

Содержание

1 Введение

Двухъядерный микропроцессор 1901ВЦ1Т 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 USB интерфейса со встроенным аналоговым приемопередатчиком (12 Мбит/с Full Speed, либо 1,5 Мбит/с Low Speed), стандартные интерфейсы UART(x3), SPI(x4), I2C(x1) и McBSP(x3). Контроллер внешней системной шины позволяет работать с внешними микросхемами статического ОЗУ и ПЗУ, Flash памятью и другими периферийными устройствами. Так же в микросхеме имеется интерфейс SDIO. Микроконтроллеры содержат три 16-ти разрядных таймера с 4-мя каналами схем захвата и интегрированными модулями ШИМ с функциями формирования «мертвой зоны» и аппаратной блокировки, системный 24-х разрядный таймер (RISC), системный таймер (DSP) и два сторожевых таймера. Для работы с аналоговыми устройствами микроконтроллер имеет два 12-ти разрядных высокоскоростных (до 0,5 Мвыб/с) АЦП с возможностью оцифровки информации из 16 каналов, встроенные датчики температуры и опорного напряжения, 12-ти 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 четыре схемы умножения тактовой частоты PLL отдельно для ядра RISC, USB интерфейса, аудио кодека 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 DMA контроллера. Контроллер DMA RISC части позволяет организовать обмен между любыми блоками памяти и периферийными устройствами (включая устройства домена DSP) без участи процессорного ядра RISC. Так же в системе имеется отдельный контроллер DMA для DSP части, который может управлятся 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 аудио кодек).

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 Описание выводов

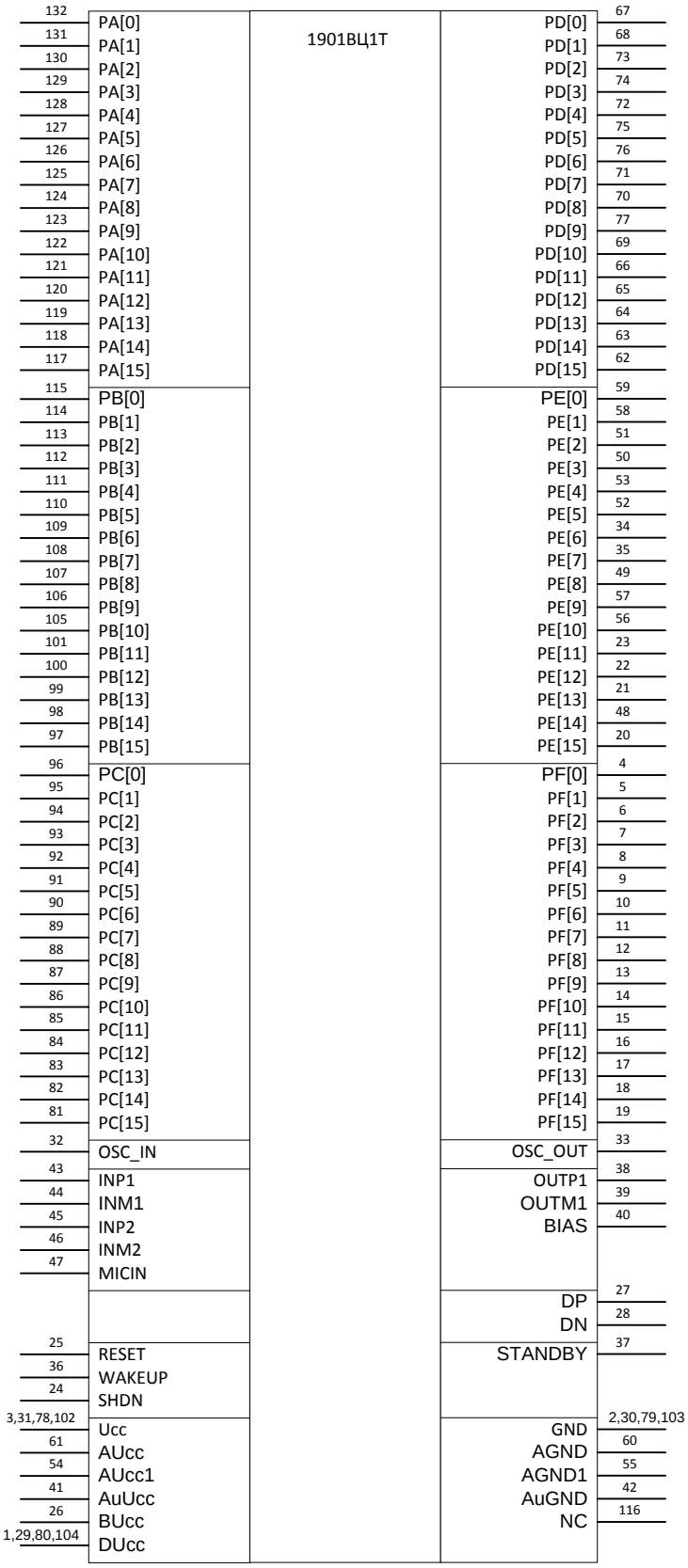


Рисунок 2–1 – Условное графическое обозначение микросхемы в корпусе 4229.132-3

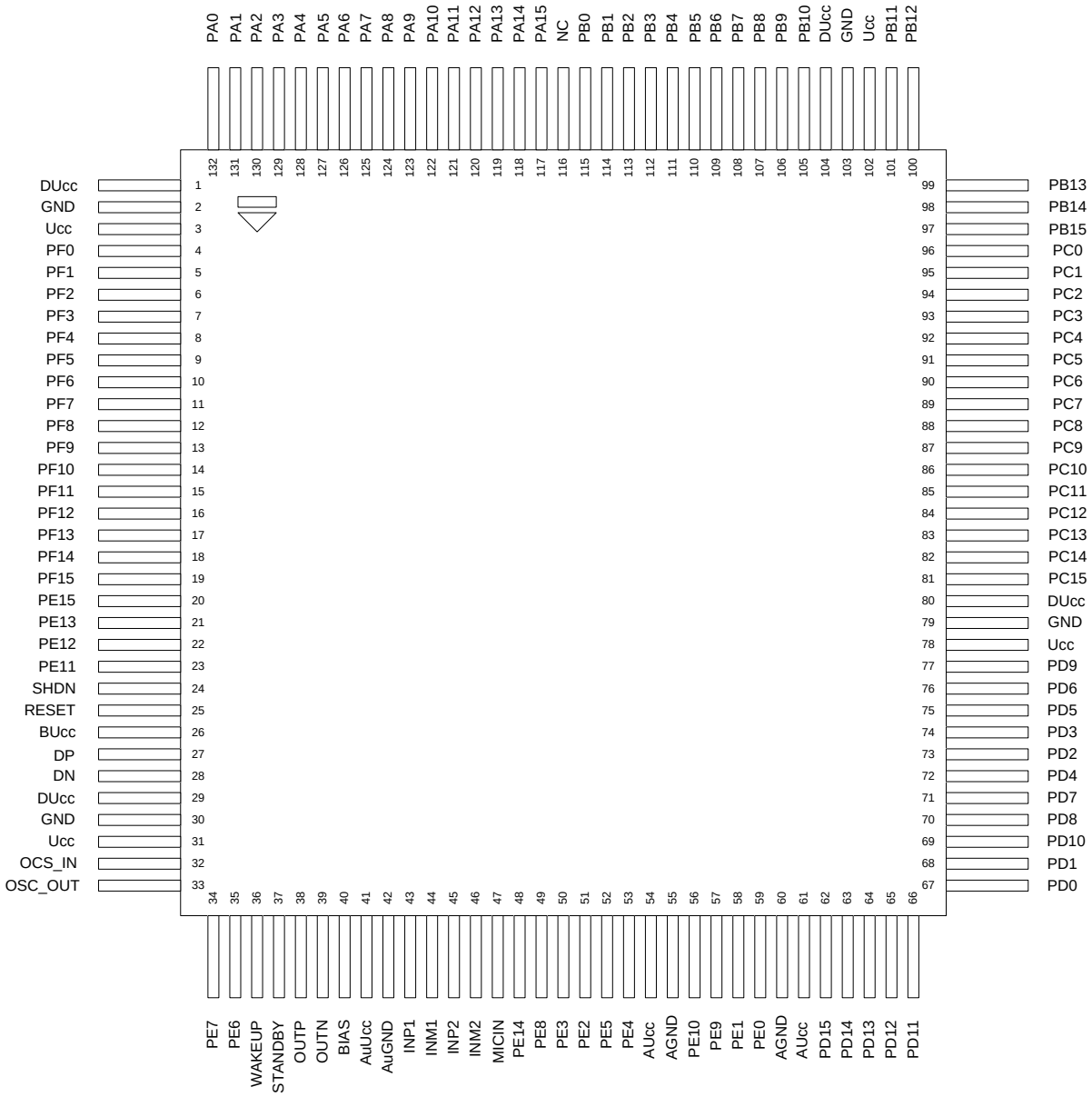


Рисунок 2–2 – Расположение выводов микросхемы в корпусе 4229.132-3

Таблица 2–1 – Описание выводов микроконтроллеров серии 1901ВЦ1Т

№ вывода корпуса	№ контактной площадки кристалла	Обозначение вывода	Назначение вывода
1	1	DU_{CC}	Питание 1,8 В (тестовый), рекомендуется оставить неподключенным
2	4	GND	Общий
3	2, 3	U_{CC}	Питание 3,0 ─ 3,6 В
4	5	PF0	Входной-выходной порт F
5	6	PF1	Входной-выходной порт F
6	7	PF2	Входной-выходной порт F
7	8	PF3	Входной-выходной порт F
8	9	PF4	Входной-выходной порт F
9	10	PF5	Входной-выходной порт F
10	11	PF6	Входной-выходной порт F
11	12	PF7	Входной-выходной порт F
12	13	PF8	Входной-выходной порт F
13	14	PF9	Входной-выходной порт F
14	15	PF10	Входной-выходной порт F
15	16	PF11	Входной-выходной порт F
16	17	PF12	Входной-выходной порт F
17	18	PF13	Входной-выходной порт F
18	19	PF14	Входной-выходной порт F
19	20	PF15	Входной-выходной порт F
20	21	PE15	Входной-выходной порт E
21	22	PE13	Входной-выходной порт E
22	23	PE12	Входной-выходной порт E
23	25	PE11	Входной-выходной порт E
24	24	SHDN	Вывод отключения внутреннего регулятора питания 1,8 В. Используется для тестовых целей. Рекомендуется оставить неподключенным
25	28	RESET	Вход Сброс
26	27	BU_{CC}	Питание батарейного домена
27	32	DP	Входной-выходной USB интерфейс
28	29	DN	Входной-выходной USB интерфейс
29	35, 36	DU_{CC}	Питание 1,8 (тестовый), рекомендуется оставить неподключенным
30	33, 34	GND	Общий
31	38, 39	U_{CC}	Питание 3,0 ─ 3,6 В
32	37	OSC_IN	Входной генератор HSE
33	40	OSC_OUT	Выходной генератор HSE
34	42	PE7	Входной-выходной порт E
35	43	PE6	Входной-выходной порт E
36	44	WAKEUP	Входной сигнал WakeUP
37	45	STANDBY	Входной сигнал StandBy
38	46	OUTP1	Вход Аудиокодек
39	47	OUTM1	Выход Аудиокодек
40	48	BIAS	Вход на задающий резистор
41	49	AuU_{CC}	Питание Аудиокодека 3,0 ─ 3,6 В
42	50	AuGND	Общий Аудиокодека
43	51	INP1	Вход Аудиокодека 1

№ вывода корпуса	№ контактной площадки кристалла	Обозначение вывода	Назначение вывода
44	52	INM1	Вход Аудиокодека 1
45	53	INP2	Вход Аудиокодека 2
46	54	INM2	Вход Аудиокодека 2
47	55	MICIN	Вход микрофона Аудиокодека
48	56	PE14	Входной-выходной порт E
49	57	PE8	Входной-выходной порт E
50	58	PE3	Входной-выходной порт E
51	59	PE2	Входной-выходной порт E
52	60	PE5	Входной-выходной порт E
53	61	PE4	Входной-выходной порт E
54	62, 63	AU_{CC1}	Питание аналоговое 3,0 ─ 3,6 В
55	64, 65	GND	Общий
56	66	PE10	Входной-выходной порт E
57	67	PE9	Входной-выходной порт E
58	68	PE1	Входной-выходной порт E
59	69	PE0	Входной-выходной порт E
60	70, 71	GND	Общий
61	72, 73	AU_{CC}	Питание аналоговое 3,0 ─ 3,6 В
62	74	PD15	Входной-выходной порт D
63	75	PD14	Входной-выходной порт D
64	76	PD13	Входной-выходной порт D
65	77	PD12	Входной-выходной порт D
66	78	PD11	Входной-выходной порт D
67	79	PD0	Входной-выходной порт D
68	81	PD1	Входной-выходной порт D
69	80	PD10	Входной-выходной порт D
70	83	PD8	Входной-выходной порт D
71	82	PD7	Входной-выходной порт D
72	85	PD4	Входной-выходной порт D
73	84	PD2	Входной-выходной порт D
74	86	PD3	Входной-выходной порт D
75	87	PD5	Входной-выходной порт D
76	88	PD6	Входной-выходной порт D
77	89	PD9	Входной-выходной порт D
78	90, 91	U_{CC}	Питание 3,0 ─ 3,6 В
79	92	GND	Общий
80	93	DU_{CC}	Питание 1,8 В (тестовый), рекомендуется оставить неподключенным
81	94	PC15	Входной-выходной порт C
82	95	PC14	Входной-выходной порт C
83	96	PC13	Входной-выходной порт C
84	97	PC12	Входной-выходной порт C
85	99	PC11	Входной-выходной порт C
86	98	PC10	Входной-выходной порт C
87	101	PC9	Входной-выходной порт C
88	100	PC8	Входной-выходной порт C
89	103	PC7	Входной-выходной порт C
90	102	PC6	Входной-выходной порт C
91	105	PC5	Входной-выходной порт C
92	104	PC4	Входной-выходной порт C
93	108	PC3	Входной-выходной порт C

№ вывода корпуса	№ контактной площадки кристалла	Обозначение вывода	Назначение вывода
94	106	PC2	Входной-выходной порт C
95	109	PC1	Входной-выходной порт C
96	107	PC0	Входной-выходной порт C
97	111	PB15	Входной-выходной порт B
98	110	PB14	Входной-выходной порт B
99	112	PB13	Входной-выходной порт B
100	113	PB12	Входной-выходной порт B
101	114	PB11	Входной-выходной порт B
102	115, 116	U_{CC}	Питание 3,0 ─ 3,6 В
103	117, 118	GND	Общий
104	119	DU_{CC}	Питание 1,8 В (тестовый), рекомендуется оставить неподключенным
105	120	PB10	Входной-выходной порт B
106	121	PB9	Входной-выходной порт B
107	122	PB8	Входной-выходной порт B
108	123	PB7	Входной-выходной порт B
109	124	PB6	Входной-выходной порт B
110	125	PB5	Входной-выходной порт B
111	126	PB4	Входной-выходной порт B
112	127	PB3	Входной-выходной порт B
113	128	PB2	Входной-выходной порт B
114	129	PB1	Входной-выходной порт B
115	130	PB0	Входной-выходной порт B
116	-	NC	Не используется
117	134	PA15	Входной-выходной порт A
118	135	PA14	Входной-выходной порт A
119	136	PA13	Входной-выходной порт A
120	137	PA12	Входной-выходной порт A
121	138	PA11	Входной-выходной порт A
122	139	PA10	Входной-выходной порт A
123	140	PA9	Входной-выходной порт A
124	141	PA8	Входной-выходной порт A
125	142	PA7	Входной-выходной порт A
126	143	PA6	Входной-выходной порт A
127	144	PA5	Входной-выходной порт A
128	145	PA4	Входной-выходной порт A
129	147	PA3	Входной-выходной порт A
130	148	PA2	Входной-выходной порт A
131	149	PA1	Входной-выходной порт A
132	150	PA0	Входной-выходной порт A
-	26, 30, 31, 41, 131-133, 146	-	Не развариваются

Таблица 2–2 – Описание выводов (по блокам) микроконтроллеров серии 1901ВЦ1Т

Вывод	№ вывода корпуса	№ КП кристалла	Дополнительные функции вывода			
			Аналог.	Основ.	Альтер.	Переопр & DSP
Порт A						
PA0	132	150	-	DATA0	SDIO_CLK	TMR1_CH1
PA1	131	149	-	DATA1	SDIO_CMD	TMR1_CH1n
PA2	130	148	-	DATA2	SDIO_DATA0	TMR1_CH2
PA3	129	147	-	DATA3	SDIO_DATA1	TMR1_CH2n
PA4	128	145	-	DATA4	SDIO_DATA2	TMR1_CH3
PA5	127	144	-	DATA5	SDIO_DATA3	TMR1_CH3n
PA6	126	143	-	DATA6	UART1_RXD	TMR1_CH4
PA7	125	142	-	DATA7	UART1_TXD	TMR1_CH4n
PA8	124	141	-	DATA8	SSP2_CLK	BSP1_RTCLK
PA9	123	140	-	DATA9	SSP2_FSS	BSP1_RTFR
PA10	122	139	-	DATA10	SSP2_TXD	BSP1_TX
PA11	121	138	-	DATA11	SSP2_RXD	BSP1_RX
PA12	120	137	-	DATA12	SSP1_RXD	TMR1_ETR
PA13	119	136	-	DATA13	SSP1_TXD	TMR1_BLK
PA14	118	135	-	DATA14	SSP1_CLK	TMR2_CH1
PA15	117	134	-	DATA15	SSP1_FSS	EXT_INT1/XF
Порт B						
PB0/ JA_TDO	115	130	-	DATA16	UART2_TXD	TMR2_CH1n
PB1/ JA_TMS	114	129	-	DATA17	TMR1_ETR	TMR2_CH2
PB2/ JA_TCK	113	128	-	DATA18	TMR1_BLK	TMR2_CH2n
PB3/ JA_TDI	112	127	-	DATA19	UART2_RXD	TMR2_CH3
PB4/ JA_TRST	111	126	-	DATA20	TMR1_CH1	TMR2_CH3n
PB5	110	125	-	DATA21	SSP3_TXD	BSP1_TX
PB6	109	124	-	DATA22	SSP3_FSS	BSP1_TFR
PB7	108	123	-	DATA23	SSP3_RXD	BSP1_RX
PB8	107	122	-	DATA24	SSP3_CLK	BSP1_TCLK
PB9	106	121	-	DATA25	TMR1_CH1n	TMR2_ETR
PB10	105	120	-	DATA26	TMR1_CH2	TMR2_BLK
PB11	101	114	-	DATA27	TMR1_CH2n	EXT_INT2
PB12	100	113	-	DATA28	SSP2_CLK	BSP1_RCLK
PB13	99	112	-	DATA29	SSP2_FSS	BSP1_RFR
PB14	98	110	-	DATA30	SSP2_TXD	BSP1_TX
PB15	97	111	-	DATA31	SSP2_RXD	BSP1_RX
Порт C						
PC0	96	107	-	COMPOUT	SSP4_FSS	SDIO_CLK
PC1	95	109	-	OE	UART1_TXD	SDIO_CMD
PC2	94	106	-	WE	UART1_RXD	SDIO_DATA0
PC3	93	108	-	BE0	SSP4_RXD	SDIO_DATA1
PC4	92	104	-	BE1	SSP4_TXD	SDIO_DATA2

Вывод	№ вывода корпуса	№ КП кристалла	Дополнительные функции вывода			
			Аналог.	Основ.	Альтер.	Переопр & DSP
PC5	91	105	-	BE2	SSP4_CLK	SDIO_DATA3
PC6	90	102	-	BE3	TMR1_CH3	UART2_TXD
PC7	89	103	-	CLOCK	TMR1_CH3n	UART2_RXD
PC8	88	100	-	SDIO_CLK	SSP3_TXD	BSP2_TX
PC9	87	101	-	SDIO_CMD	SSP3_FSS	BSP2_RTFR
PC10	86	98	-	SDIO_DATA0	SSP3_RXD	BSP2_RX
PC11	85	99	-	SDIO_DATA1	SSP3_CLK	BSP2_RTCLK
PC12	84	97	-	SDIO_DATA2	SSP4_FSS	BSP3_RTFR
PC13	83	96	-	SDIO_DATA3	SSP4_CLK	BSP3_RTCLK
PC14	82	95	-	I2C1_SCK	SSP4_TXD	BSP3_TX
PC15	81	94	-	I2C_SDA	SSP4_RXD	BSP3_RX
Порт D						
PD0/ JB_TMS	67	79	ADC0_REF+	TMR1_BLK	I2C1_SCK	BSP3_RX
PD1/ JB_TCK	68	81	ADC1_REF-	TMR1_ETR	I2C1_SDA	BSP3_TX
PD2/ JB_TRST	73	84	ADC2	BUSY1	SSP1_TXD	BSP2_TX
PD3/ JB_TDI	74	86	ADC3	TMR1_CH1	SSP1_FSS	BSP2_TFR
PD4/ JB_TDO	72	85	ADC4	TMR1_CH1n	SSP1_RXD	BSP2_RX
PD5	75	87	ADC5	CLE	SSP1_CLK	BSP2_TCKL
PD6	76	88	ADC6	ALE	TMR1_CH4	BSP3_TCLK
PD7	71	82	ADC7	TMR1_CH2	UART3_TXD	BSP3_TFR
PD8	70	83	ADC8	TMR1_CH2n	UART3_RXD	BSP3_TX
PD9	77	89	ADC9	TMR1_CH3	TMR1_CH4n	BSP3_RFR
PD10	69	80	ADC10	TMR1_CH3n	TMR2_BLK	BSP3_RX
PD11	66	78	ADC11	TMR1_CH4	TMR2_ETR	BSP3_RCLK
PD12	65	77	ADC12	TMR1_CH4n	SSP2_FSS	BSP2_RFR
PD13	64	76	ADC13	UART3_TXD	SSP2_RXD	BSP2_RX
PD14	63	75	ADC14	UART3_RXD	SSP2_CLK	BSP2_RCKL
PD15	62	74	ADC15	EXT_INT1	SSP2_TXD	BSP2_TX
Порт E						
PE0	59	69	DAC1_OUT	ADDR16	SSP4_FSS	BSP1_RTFR
PE1	58	68	DAC1_REF	ADDR17	SSP4_CLK	BSP1_RTCLK
PE2	51	59	COMP_IN1	ADDR18	SSP4_TXD	BSP1_TX
PE3	50	58	COMP_IN2	ADDR19	SSP4_RXD	BSP1_RX
PE4	53	61	COMP_REF+	ADDR20	SSP2_TXD	UART1_TXD
PE5	52	60	COMP_REF-	ADDR21	SSP2_FSS	UART1_RXD
PE6	35	43	OSC_IN32	ADDR22	SSP2_RXD	EXT_INT3
PE7	34	42	OSC_OUT32	ADDR23	SSP2_CLK	TMR2_CH4
PE8	49	57	COMP_IN3	ADDR24	TMR2_CH4	TMR2_CH4n
PE9	57	67	DAC2_OUT	ADDR25	TMR2_CH4n	TMR3_CH1
PE10	56	66	DAC2_REF	ADDR26	UART2_TXD	TMR3_ETR
PE11	23	25	-	ADDR27	UART2_RXD	TMR3_BLK
PE12	22	23	-	ADDR28	SSP1_RXD	BSP2_RTFR

Вывод	№ вывода корпуса	№ КП кристалла	Дополнительные функции вывода			
			Аналог.	Основ.	Альтер.	Переопр & DSP
PE13	21	22	-	ADDR29	SSP1_TXD	BSP2_RTCLK
PE14	48	56	-	ADDR30	SSP1_CLK	BSP2_TX
PE15	20	21	-	ADDR31	SSP1_FSS	BSP2_RX
Порт F						
PF0	4	5	-	ADDR0	UART3_RXD	TMR3_CH1n
PF1	5	6	-	ADDR1	UART3_TXD	TMR3_CH2
PF2	6	7	-	ADDR2	SSP4_RXD	BSP3_RX
PF3	7	8	-	ADDR3	SSP4_TXD	BSP3_TX
PF4/ MODE[0]	8	9	-	ADDR4	SSP4_CLK	BSP3_TCKL
PF5/ MODE[1]	9	10	-	ADDR5	SSP4_FSS	BSP3_TFR
PF6/ MODE[2]	10	11	-	ADDR6	TMR2_CH3n	TMR3_CH2n
PF7	11	12	-	ADDR7	TMR2_CH3	TMR3_CH3
PF8	12	13	-	ADDR8	TMR2_CH2n	TMR3_CH3n
PF9	13	14	-	ADDR9	TMR2_CH2	TMR3_CH4
PF10	14	15	-	ADDR10	TMR2_CH1n	TMR3_CH4n
PF11	15	16	-	ADDR11	TMR2_CH1	EXT_INT4
PF12	16	17	-	ADDR12	SSP3_TXD	BSP3_TX
PF13	17	18	-	ADDR13	SSP3_FSS	BSP3_RFR
PF14	18	19	-	ADDR14	SSP3_RXD	BSP3_RX
PF15	19	20	-	ADDR15	SSP3_CLK	BSP3_RCKL
Аудио интерфейс						
MICIN	47	55	-			
INP2	45	53	-			
INM2	46	54	-			
INP1	43	51	-			
INM1	44	52	-			
OUTP1	38	46	-			
OUTM1	39	47	-			
BIAS	40	48	-			
Системное управление						
RESET	25	28	Сигнал внешнего сброса			
WAKEUP	36	44	Сигнал внешнего выхода из режима Standby			
STANDBY	37	45	Флаг режима Standby			
OSC_IN	32	37	Вход генератора HSE			
OSC_OUT	33	40	Выход генератора HSE			
			USB интерфейс			
DP	27	32	Шина USB D+			
DN	28	29	Шина USB D-			
Питание						
Ucc	102,3, 78,31	115,116, 2,3,90,91 38,39	Питание 3,0…3,6 В			
AUcc	54,61	62, 63, 72, 73	Аналоговое питание АЦП и ЦАП 3,0…3,6 В (должно совпадать с Ucc)			

Вывод	**№ вывода корпуса**	**№ КП кристалла**	**Дополнительные функции вывода**			
			Аналог.	**Основ.**	**Альтер.**	**Переопр & DSP**
AuUcc	41	49	Аналоговое питание аудиокодека 3,0…3,6 В (должно совпадать с Ucc)			
BUcc	26	27	Батарейное питание 1,8…3,6 В			
GND	2,103, 30,79	4,117, 118,33, 34,92	Общий			
AuGND	42	50	Общий			
AGND	55,60	64,65,70, 71				
Выводы для тестирования и исследования						
SHDN	24	24	Вывод отключения внутреннего регулятора питания 1,8 В. Используется для тестовых целей. Рекомендуется оставить неподключенным			
DUcc	1,29, 80, 104	1,35,36, 93,119	Питание 1,8 В (тестовый), рекомендуется оставить неподключенным			
Не используются						
NC	116	-	Не используется			
-	-	26, 30, 31, 41, 131-133, 146	Не развариваются			

3 Структурная блок-схема микросхемы

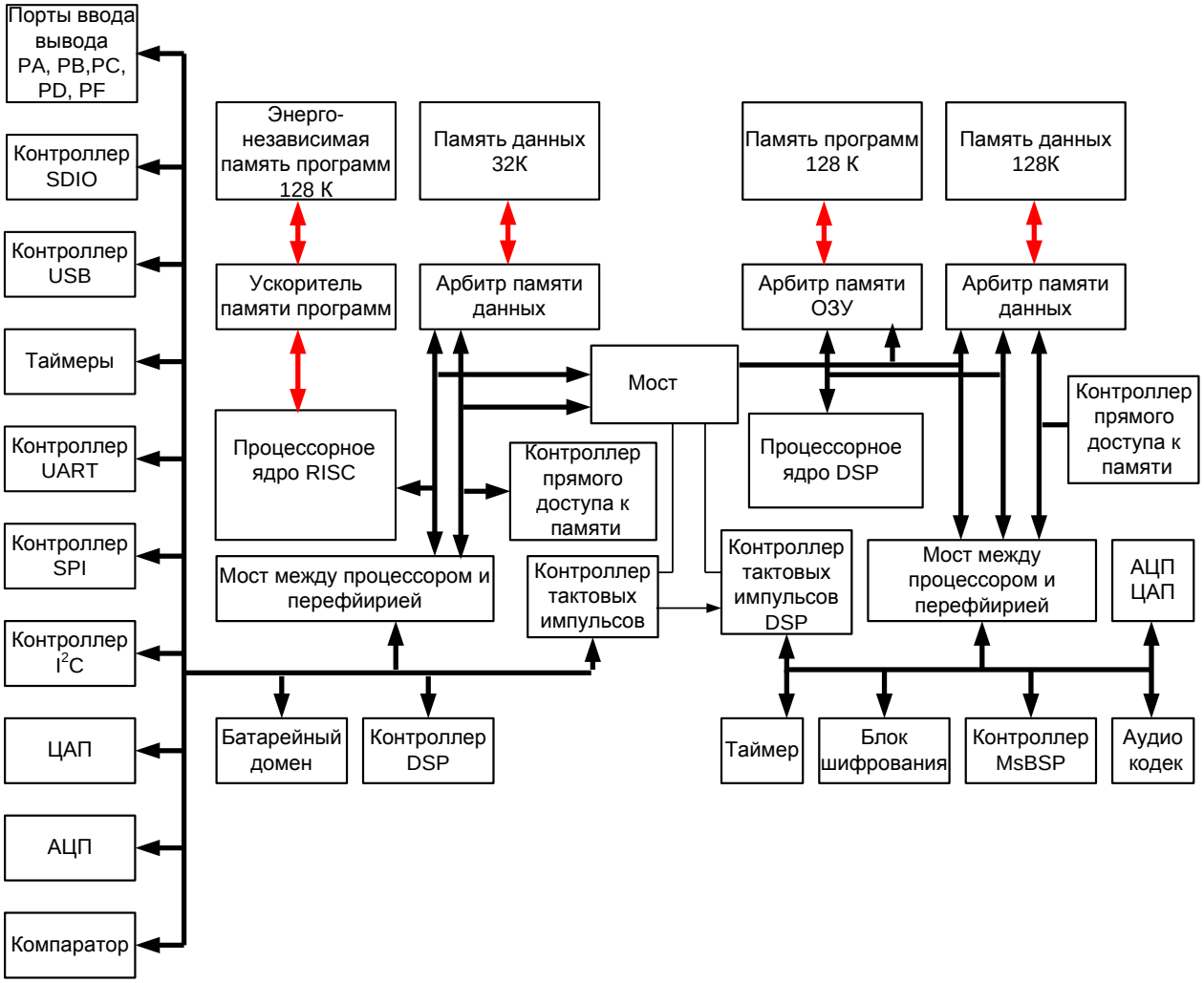


Рисунок 3–1 – Структурная блок-схема

4 Описание функционирования микросхемы

4.1 Система питания

Двухъядерный микропроцессор серии 1901ВЦ1Т имеет несколько типов выводов питания:

- **Ucc выводы:** Основное питание микросхемы, включает питание пользовательских выводов, встроенного регулятора напряжения, USB PHY и генераторов. Входное напряжение должно быть в пределах от 3,0 до 3,6 В.
- **DUcc выводы:** Питание внутренней цифровой части, памяти ОЗУ и Flash памяти. В нормальном режиме работы питание цифровой части формируется внутренним регулятором напряжения из UCC и этот вывод не используется (должен остаться не подсоединенным). Напряжение на выводе DUcc должно быть в пределах от 1,62 до 1,98 В.
- **BUcc вывод:** Питание батарейного домена используется при отсутствии основного питания Ucc для питания батарейного домена и LSE генератора. Переключение с основного питания на батарейное происходит автоматически при снижении ниже 2,0 В уровня основного питания Ucc. Переключение с батарейного питания на основное происходит автоматически, спустя примерно 4 мс, после повышения питания Ucc 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 BUcc должен быть объединен с UCC.
- **AUcc выводы:** Питание аналоговых блоков выведено на отдельные выводы для уменьшения помех создаваемых работой других блоков. На данные выводы должно подаваться напряжения с того же источника что и Ucc, но при этом на печатной плате должны быть применены меры по снижению наводки помех. Для корректной работы блока АЦП входное напряжение должно быть в пределах от 2,4 до 3,6 В. Если входное напряжение будет в пределах от 2,2 до 2,4 В то корректная работа АЦП не гарантируется.
- AuUcc выводы: Питание аналоговой части аудиокодека выведено на отдельные выводы для уменьшения помех создаваемых работой других блоков. На данные выводы должно подаваться напряжения с того же источника что и Ucc, но при этом на печатной плате должны быть применены меры по снижению наводки помех.
- **GND выводы:** Основная земля питания.
- **AGND выводы:** Земля аналогового питания AUCC. Данные выводы должны соединяться с GND, но при этом на печатной плате должны быть применены меры по снижению наводки помех.
- **AuGND выводы:** Земля аналогового питания AuUcc. Данные выводы должны соединяться с GND, но при этом на печатной плате должны быть применены меры по снижению наводки помех.

4.2 Структурная схема подачи питания (1, 2, 3, 4)

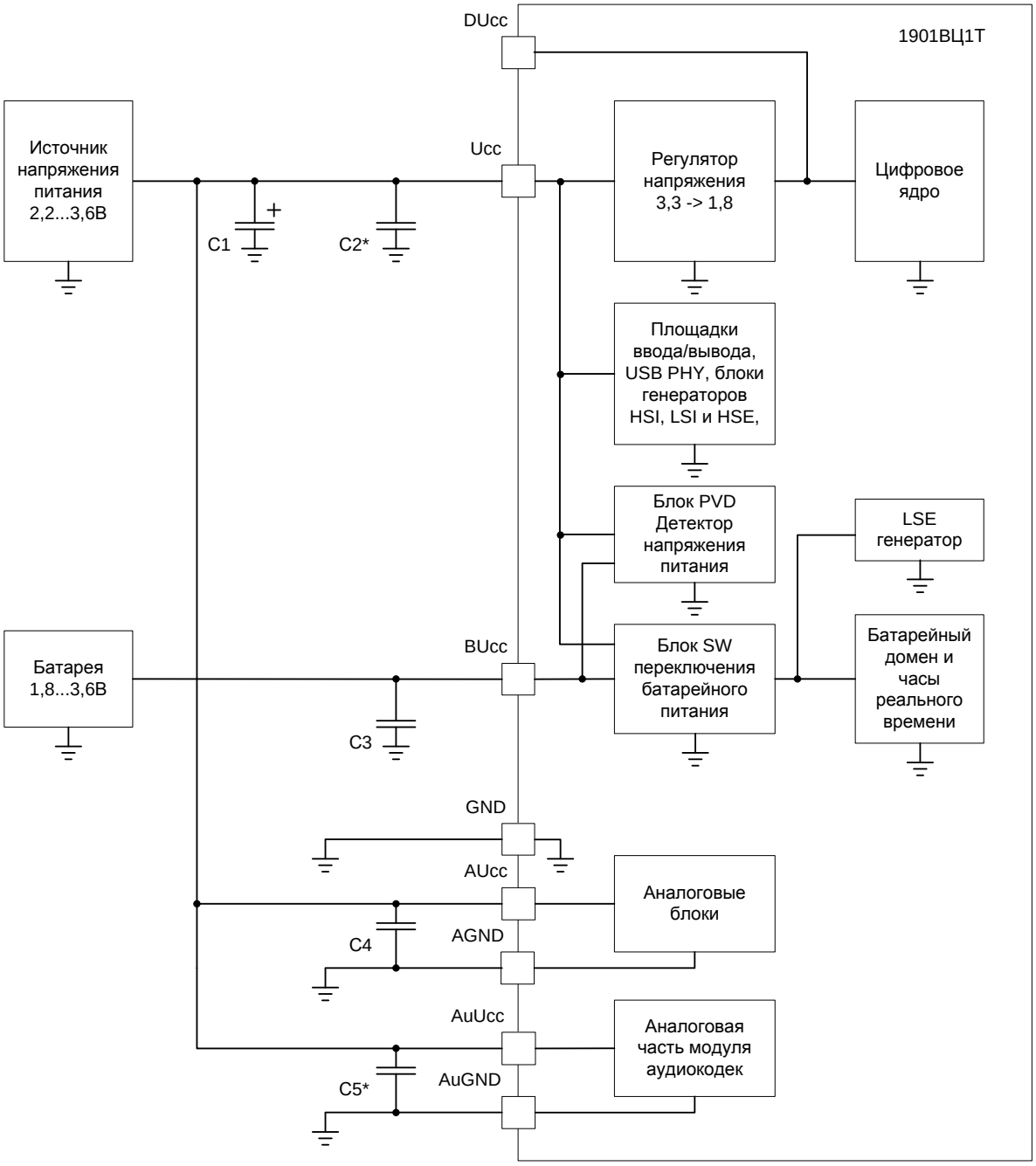


Рисунок 4–1 – Структурная блок-схема

Примечания:

* - конденсаторы должны быть установлены у каждого вывода питания.

1. Конденсатор C1 = 22 мкФ, C2 = C3 = C4 = C5 = 0,1 мкФ.

2. Если не используется батарейное питание, то вывод BUcc должен быть объединен с Ucc.

3. Если используется интерфейс USB, то напряжение питания Ucc должно быть в пределах от 3,0 до 3,6 В.

4. Если используется АЦП или ЦАП, то напряжение питания Ucc (AUcc и AuUcc) должно быть в пределах от 2,4 до 3,6 В.

Микроконтроллер имеет несколько режимов энергопотребления, подробнее смотри раздел «Управление электропитанием».

Микроконтроллер имеет встроенный детектор напряжения питания, подробнее смотри раздел детектор напряжения питания.

4.2.1 Схема сброса при включении и выключении основного питания

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

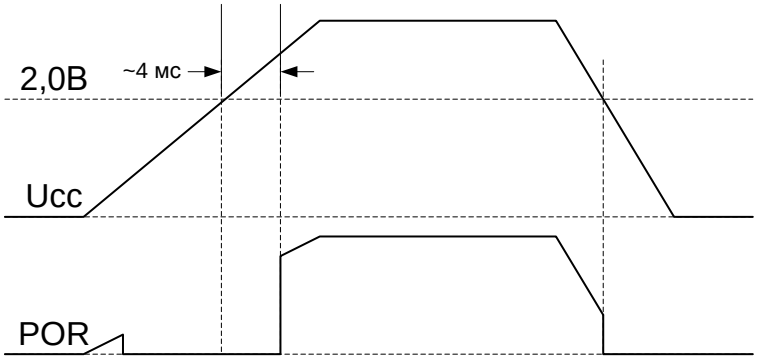


Рисунок 4–2 – Диаграмма формирования сигнала POR

При снижении напряжения питания Ucc ниже уровня Upor сигнал POR вырабатывается без задержки.

Сигнал POR так же служит для переключения питания батарейного домена между BU_{CC} и U_{CC}.

При включении основного напряжения питания Ucc автоматически включается встроенный регулятор напряжения для формирования напряжения DUcc питания цифрового ядра. В ходе работы микроконтроллера встроенный регулятор может быть отключен, подробнее в разделе «Управление электропитанием».

Микроконтроллер так же может быть сброшен внешним сигналом сброса RESET, 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.

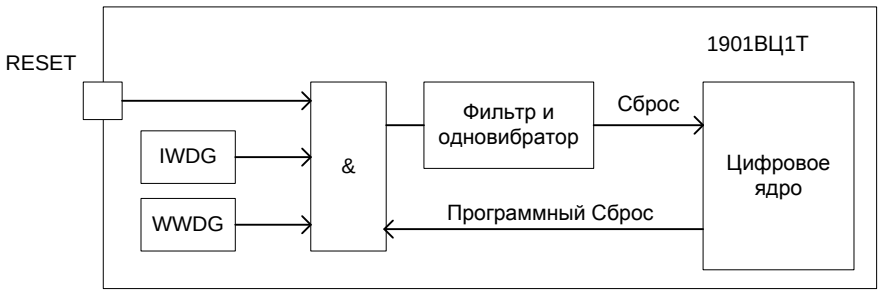


Рисунок 4–3 – Структурная блок-схема сброса

При приходе импульсов сброса длительностью менее tminreset, эти импульсы отфильтровываются и не приводят к сбросу процессора. Если длительность

импульса больше tmaxreset вырабатывается сигнал сброса. При этом длительность сформированного сигнала сброса будет не менее treset.

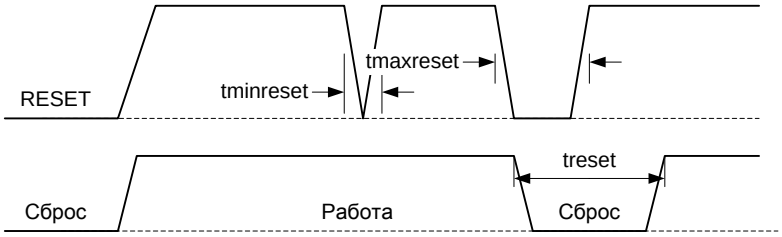


Рисунок 4–4 – Формирование сигнала сброса

4.3 Организация памяти

4.3.1 Структурная схема

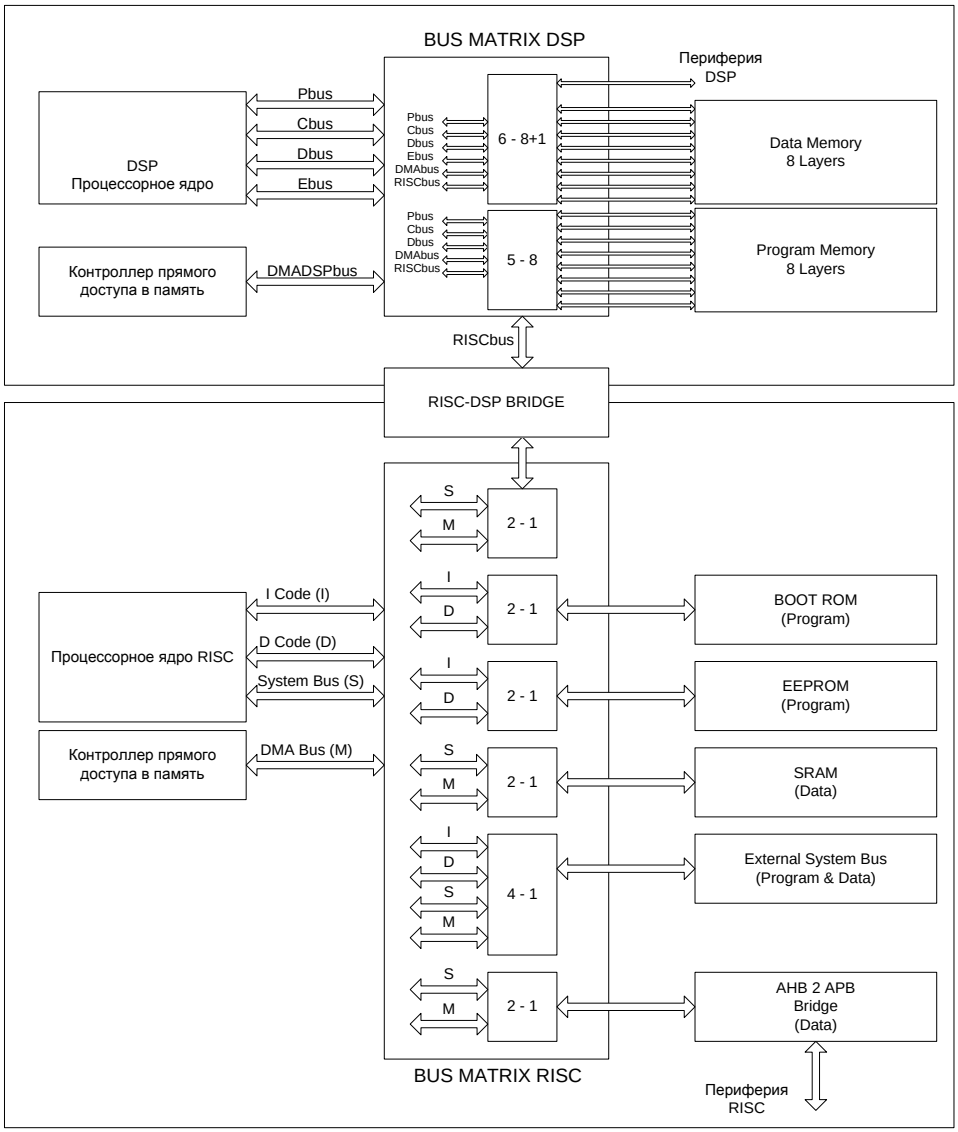


Рисунок 4–5 – Структурная схема микроконтроллера 1901ВЦ1Т

Вся система структурно разделена на две подсистемы: RISC и DSP. В подсистему DSP входят память программ (128 кБ), память данных (128 кБ),

периферийные модули McBSP, Аудио кодек, Крипто модуль, таймер, некоторые отдельные специальные управляющие регистры. Все перечисленные устройства отображены в памяти DSP и памяти RISC. Кроме того в память RISC отбражены и другие устройства: Flash память, RAM (RISC), внешние устройства, другие периферийные модули. Таким образом, DSP ядро имеет доступ к DSP подсистеме микроконтроллера, RISC ядро является ведущим в системе и имеет доступ ко всем частям системы и к внешним устройствам через 32-х разрядную внешнюю шину.

В подсистеме RISC реализован контроллер прямого доступа в память (DMA) осуществляющий выборку через шину DMA Bus. Он может работать с память RAM (RISC), внешней шиной, всей подсистемой DSP и всеми периферийными устройствами системы. В подсистеме DSP так же 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) частоте.

Потери на пересинхронизацию при чтении составляют три периода наименьшей частоты. При этом в модуль интегрирован механизм предчтения следующих, за запрашиваемыми 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 DSP подсистемы (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 RISC подсистеме 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 микроконтроллера. 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 микроконтроллера 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 контроллер EERPOM.

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

Память DATA Memory (DSP)

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

Карта памяти RISC

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

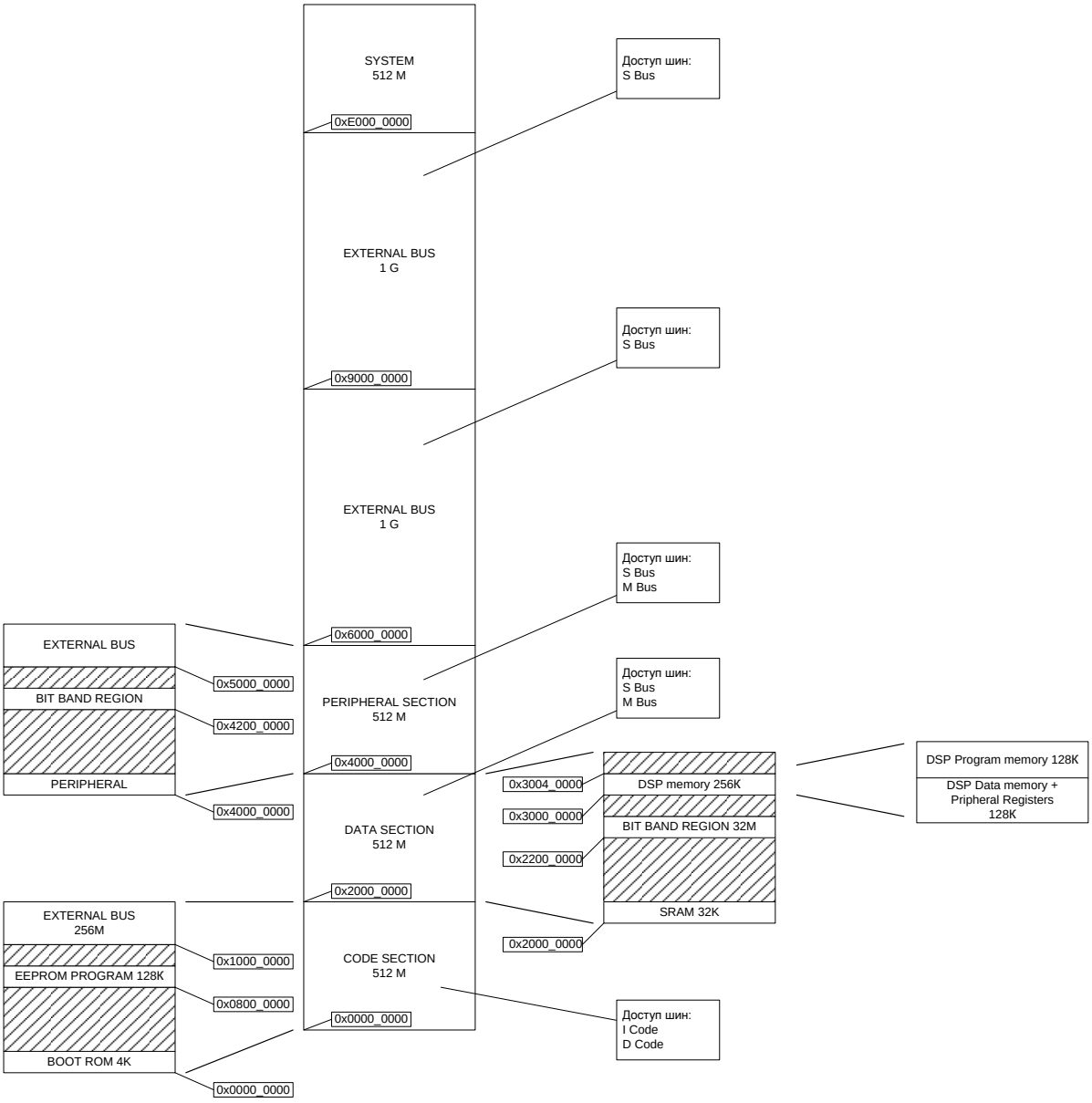


Рисунок 4–6 – Карта памяти RISC

Процессорное ядро RISC имеет три системных шины (Рисунок 4–5):

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

Так же в подсистеме RISC реализован шина контроллера прямого доступа в память (Рисунок 4–5) :

- M Bus.

Секция CODE:

Область BOOT ROM:

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

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 область так же 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 область Internal SRAM. Работа с BIT DAND REGION позволяет осуществлять операции «Чтение-Модификация-Запись», «Установка бита» и «Сброс Бита» одной инструкцией.

Область DSP MEMORY:

Область отображения адресного пространства DSP подсистемы в адресное пространство RISC Предназначена для обмена данными с DSP ядром 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 область PERIPHERAL. Работа с BIT DAND 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.

4.3.1.1 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 Stronly-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 4–1).

Таблица 4–1 – Последовательность обращений инструкций к памяти

A2 / A1	Доступ в Normal	Доступ в Device Не shareable	Доступ в Device shareable	Strongly-ordered
Доступ в Normal	-	-	-	-
Доступ в Device Не shareable	-	<	-	<
Доступ в Device shareable	-	-	<	<
Strongly-ordered	-	<	<	<

Где « - » означает, что система памяти не гарантирует последовательность выполнения обращений, а « < » означает что обращение инструкции A1 всегда будет выполнено перед инструкций A2.

Поведение обращений к памяти

Поведение обращений описано ниже (Таблица 4–2).

Таблица 4–2 – Последовательность обращений инструкций к памяти

Адресный диапазон	Регион памяти	Тип памяти	XN	Описание
0x00000000-0x1FFFFFFF	Code	Normal	-	Область памяти для кода программы, данные так же могут храниться здесь.
0x20000000-0x2FFFFFFF	SRAM	Normal	-	Область памяти для данных. Код программы так же может располагаться здесь. Этот регион содержит области bit-band доступа
0x30000000-0x3FFFFFFF	DSP	Normal	-	Область DSP памяти и регистров DSP периферии. Не рекомендуется использовать память DSP для расположения кодов команд RISC. При записи и чтения этой области памяти возможны задержки вследствии пересинхронизации обращений. Подробнее см. раздел “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 4–3 – 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	-	WBWA
0x80000000-0x9FFFFFFF				WT
0xA0000000-0xBFFFFFFF	External device	Device	Shareable	
0xC0000000-0xDFFFFFFF			”non-shareable”	
0xE0000000-0xE00FFFFF	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 происходить 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. Смотри описание инструкции DMB.

DSB

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

ISB

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

Инструкции барьерной синхронизации используются, например, в таких случаях:

- Программирование MPU:
  - используйте DSB инструкцию для уверенности в том, что изменение настроек MPU будет иметь эффект немедленно вслед за изменением контекста;
  - используйте ISB инструкцию 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 ISB инструкция 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 ISB инструкцию 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 DSB инструкцию после изменения. Это гарантирует, что изменение произойдет при завершении DSB инструкции.
- 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 DMB инструкции.

Bit-band регионы

Механизм bit-band отображает каждый бит в bit-band региона 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 как показано в Таблица 4–4.

Таблица 4–4 – 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 * 32) + (bit_number * 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

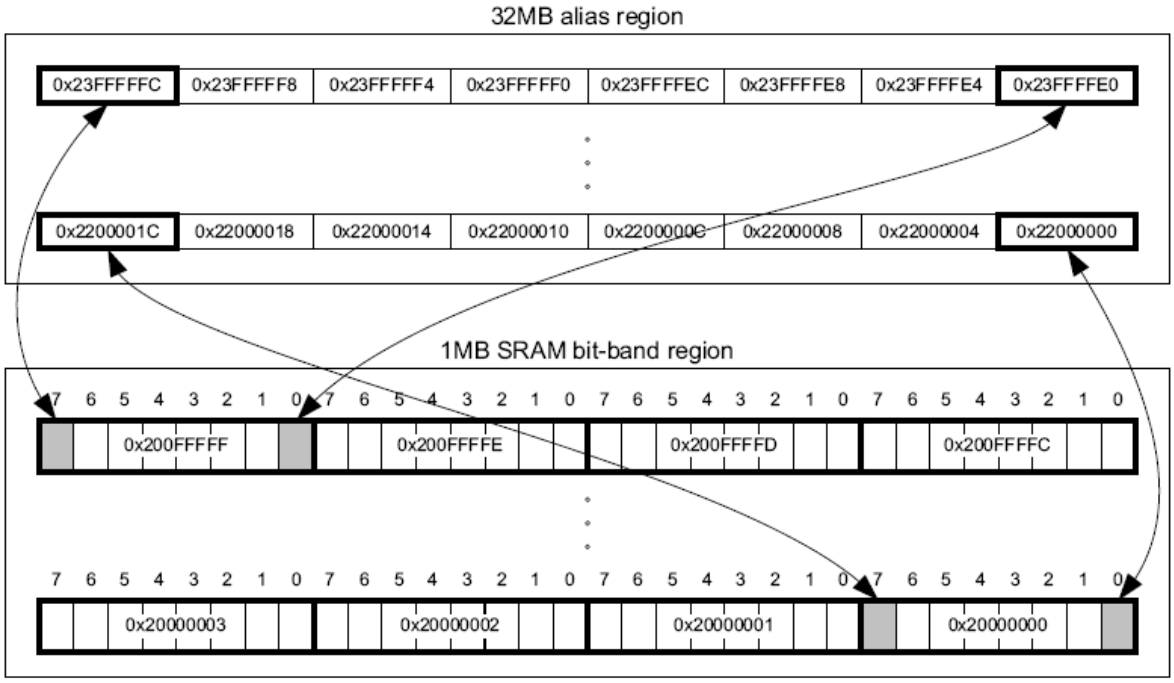


Рисунок 4–7 – Схема отображения региона bit-band alias в регионе bit-band region

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:

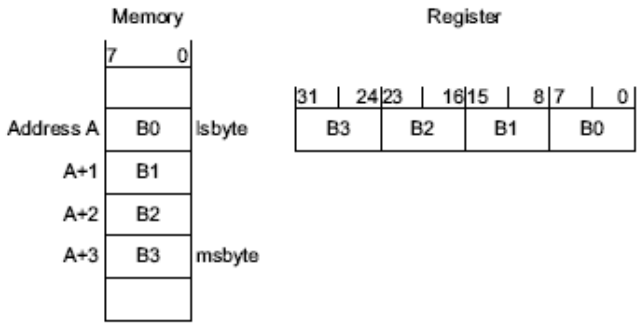


Рисунок 4–8 – Организация расположения байтов в 32-х битной памяти

4.3.1.2 Примитивы 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-Exlusive

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

Команда Store-Exlusive

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-Exlusive 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);*

4.3.1.3 Базовые адреса RISC

Таблица 4–5 – 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 DSP подситемы: Timer(DSP), DMA(DSP), АудиоКодер,

			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		WWDT	Регистры контроллера сторожевого таймера WWDT
0x4006_8000		IWDT	Регистры контроллера сторожевого таймера IWDT
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 A
0x400B_0000		PORTB	Регистры управления порта B
0x400B_8000		PORTC	Регистры управления порта C
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

4.3.2 Карта памяти DSP

Процессорное ядро DSP имеет 4 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 простанства представляют собой расслоеные ОЗУ размером по 128 Кбайт каждое, с организацией 64К х 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 DSP ядра 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 DSP подсистемы. 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.

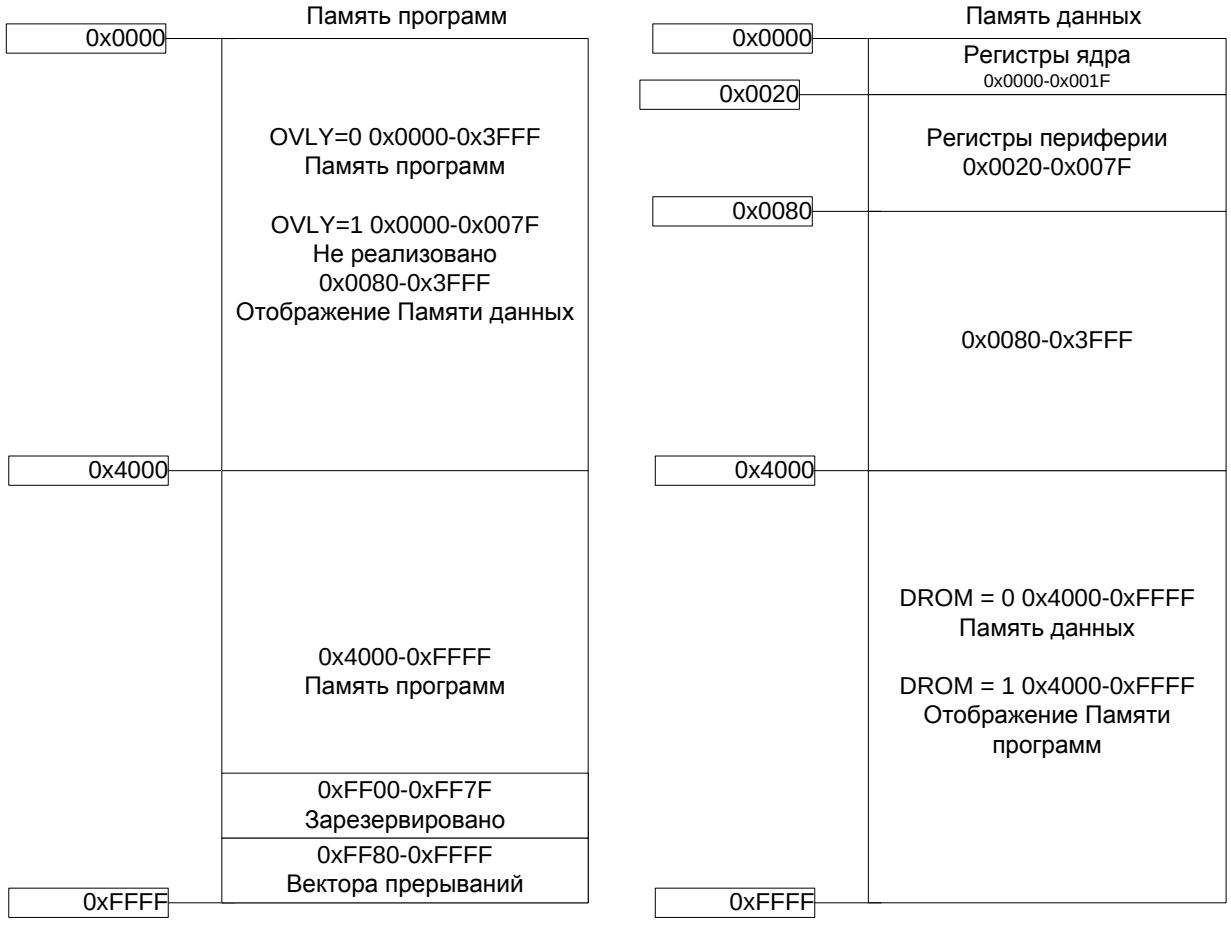


Рисунок 4–9 – Карта памяти DSP

4.3.2.1 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 времени. Регистр флага прерывания (IFR), указывает текущий статус прерываний.

Регистры состояния (ST0, ST1)

Регистры состояния ST0 и ST1 содержат статус различных условий и мод 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 (A, B)

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

Временный регистр (T)

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

- Один из сомножителей для операций умножения и умножения с накоплением.
- Счётчик динамического сдвига (программируемый во время выполнения) для инструкций с операцией сдвига, как например, ADD, LD, и SUB инструкции.
- 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 T, и затем инструкция NORM использует величину регистра T 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 предекрементом 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 моды процессора (PMST)

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

Таблица 4–6 – 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 A, младшая часть(биты 15-0)
0x0009	AH	Аккумулятор A, старшая часть(биты 31-16)
0x000A	AG	Аккумулятор A, биты защиты(биты 39-32)
0x000B	BL	Аккумулятор B, младшая часть(биты 15-0)
0x000C	BH	Аккумулятор B, старшая часть(биты 31-16)
0x000D	BG	Аккумулятор B, 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

Адрес (DSP)	Имя	Описание
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	Счётчик повторения блока
0x001B	RSA	Начальный адрес повторяемого блока
0x001C	REA	Конечный адрес повторяемого блока
0x001D	PMST	Регистр состояния моды процессора
0x001E	XPC	Регистр расширения счётчика команд
0x001E…0x001F	-	Зарезервировано

Периферийные регистрны DSP

В DSP части содержатся периферийный модули DMA (DSP), System timer (DSP), Audio Codec, 3 модуля McBSP, Crypto модуль. Так же в таблицу адресоввключены регистр управления синхросигналами CLKMD, регистры прерываний (AIRQ, DIRQ) и регистр ETDR. Адреса всех периферийных модулей представлены в области данных DSP в диапазоне с адреса 0x0020 по 0x007F. Подробное описание регитров периферийных модулей приведены в соответсвующих разделах. Адреса регистров DSP части для ядер DSP и RISC представлены в таблице ниже (Таблица 4–7).

Таблица 4–7 – Регистры периферийных модулей DSP подсистемы

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	CtrlDL1	BSP1
0x0027	0x3000004E	CtrlDH1	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	CtrlDL2	BSP2
0x002F	0x3000005E	CtrlDH2	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	CtrlDL3	BSP3
0x0037	0x3000006E	CtrlDH3	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	
0x003F	0x3000007E	ETDR	ETDR
0x0040	0x30000080	CRPT_CWR	Crypto
0x0041	0x30000082	зарезервировано	
0x0042	0x30000084	CRPT_SR	Crypto
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

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

4.3.3 Отображение памяти DSP в памяти RISC

Память DSP подсистемы (включая периферийные блоки) отображается в адресное пространство RISC в область DATA с адреса 0x30000000 по адрес 0x30040000. DMA RISC так же имеет доступ к памяти DSP по тем же адресам. RISC ядро и DMA RISC доступно все адресное пространство DSP подсистемы, кроме регистров ядра DSP. Соответстие карты памяти DSP и RISC показано на Рисунок 4–10 и, также, в Таблица 4–8. Ядро RISC использует байтовую адресацию, ядро DSP полусловную (16 бит). В таблице 8 показано подробное отображение регистров периферийных модулей DSP подсистемы в памяти RISC.

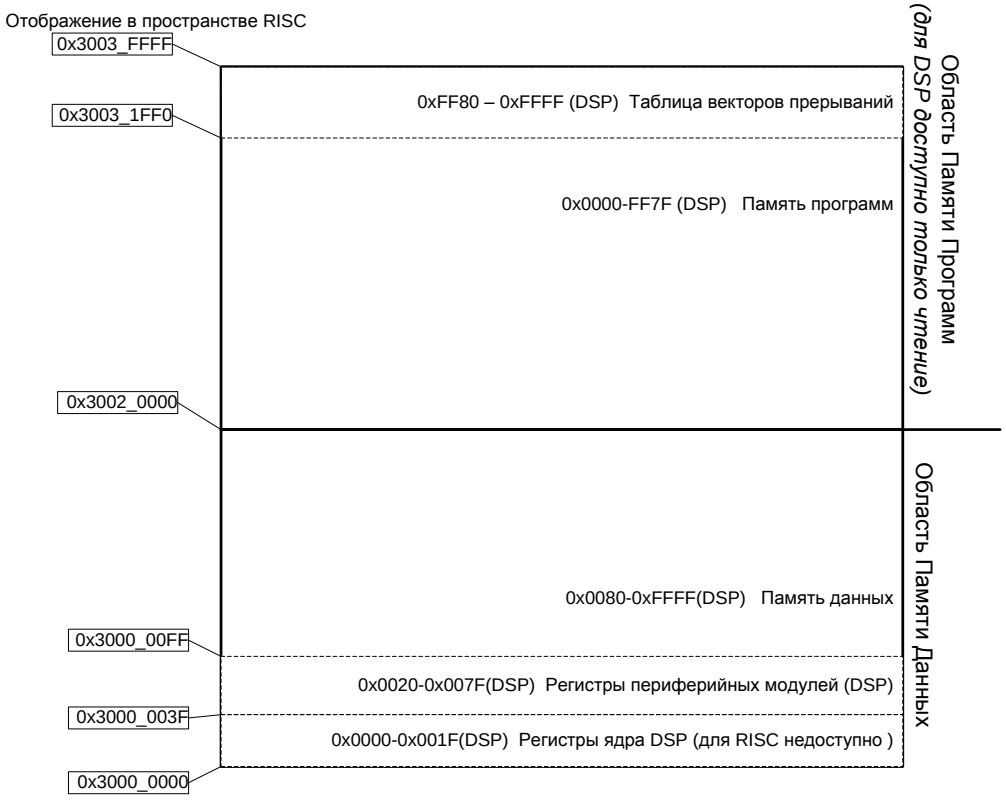


Рисунок 4–10 – Отображение памяти DSP в адресное пространство RISC

Таблица 4–8 – 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 2 расслоенных модуля ОЗУ, состоящие из 4 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 (4 части памяти программ и 4 части памяти данных) одним из устройств системы (ядром RISC, DMA RISC, ядром DSP илиDMA 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 уменьшеннию быстродействия системы.

4.4 Загрузочное ПЗУ и режимы работы микроконтроллера

После включения питания и снятия внутренних (POR) и внешних (RESET) сигналов сброса, микроконтроллер начинает выполнять программу из загрузочной области ПЗУ BOOT ROM. В загрузочной программе микроконтроллер определяет, в каком из режимов он 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 микроконтроллера уровни на выводах MODE[2:0] не влияют на режим функционирование микроконтроллера, если установлен бит FPOR. В пользовательской программе выводы PF[6:4] могут использоваться пользователем.

Таблица 4–9 – Режимы первоначального запуска микроконтроллера

MODE[2:0]	Режим	Стартовый адрес/таблица векторов прерываний	Описание
000	Микроконтроллер без отладки	0x0800_0000	Процессор начинает выполняет программу из внутренней FLASH

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

При работе в режиме отладки разрешается работа отладочного интерфейса JTAG/SW. При этом к микроконтроллеру может быть подключен JTAG/SW адаптер, с помощью которого программные средства разработки позволяют работать с микроконтроллером 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 4–11).

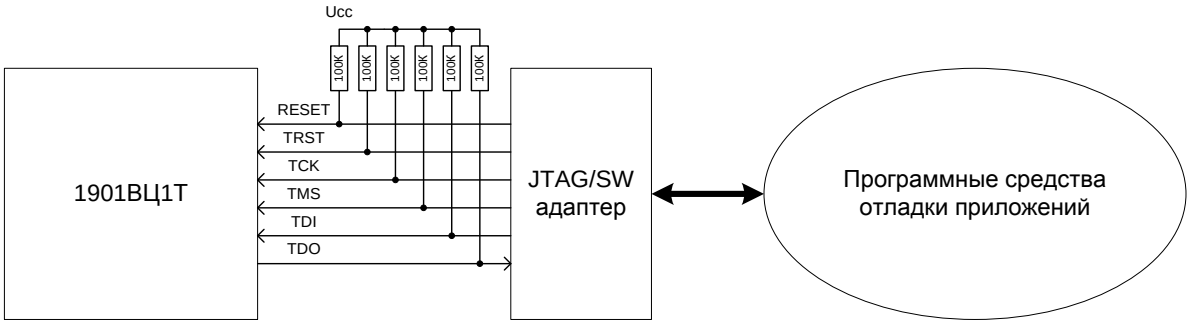


Рисунок 4–11 – Схема работы в режиме отладки

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 микроконтроллера представленные в Таблица 4–10.

Таблица 4–10 – Переопределение выводов интерфейса JTAG/SW

Вывод JTAG/SW	Вывод микроконтроллера	Описание
JTAG_A		
TRST	PB4/DATA20/JA_TRST	В качестве выводов интерфейса используются выводы порта B совмещенные с данными внешней системной шины, использование который 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 который 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	

4.5 UART загрузчик

Для загрузки программ в режиме UART загрузчика используется реализованный на периферийный модуль UART3 специальный протокол обмена. При этом для работы модуля используется выводы в соответствии с Таблица 4–11.

Таблица 4–11 – Используемые порты ввода/вывода UART загрузчиком

Режим	Rx	Tx
110b	PF0	PF1

Данные режимы предоставляют и 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 микроконтроллеров) весьма велик (от 6 МГц до 10 МГц), то требуется этап подбора значения делителя частоты UART3 для синхронизации с внешним устройством.

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 Таблица 4–12.

Таблица 4–12 – Команды UART загрузчика

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

4.5.1 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 ощибки 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.

4.5.1.1 Команда CMD_SYNC

Пустая команда.

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

Таблица 4–13 – 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

4.5.1.2 Команда CMD_CR

Выдача приглашения Master-у

Таблица 4–14 – 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. Выдает код команды CMD_CR. Выдает код 0x0A Выдает код 0x3E (ASCII символ '>')

4.5.1.3 Команда CMD_BAUD

Установка скорости обмена

Таблица 4–15 – 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	Если параметр принят с ошибками, то выдает код ошибки ERR_CHN или ERR_BAUD и завершает обработку текущей команды. Выдает код команды CMD_BAUD. Устанавливает новое значение скорости обмена

4.5.1.4 Команда CMD_LOAD

Загрузка массива байт в память микроконтроллера

Таблица 4–16 – 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 Если хотя бы один из параметров принят с ошибками, то выдает код ошибки ERR_CHN и завершает обработку текущей команды
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. 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')

4.5.1.5 Команда CMD_VFY

Выдача массива байт из памяти микроконтроллера

Таблица 4–17 – 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 Если хотя бы один из параметров принят с ошибками, то выдает код ошибки ERR_CHN и завершает обработку текущей команды
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. Выдает код команды CMD_VFY. Выдает массив байт младшим байтом вперед. По окончании передачи массива выдает код ответа REPLY_OK = 0x4B ('K')

4.5.1.6 Команда CMD_RUN

Запуск программы на выполнение.

Таблица 4–18 – 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.	Если параметр принят с ошибками, то выдает код ошибки ERR_CHN и завершает обработку текущей команды. Выдает код команды CMD_RUN. Устанавливает значение MSP и PC согласно таблице векторов (NVIC не перепрограммируется) и, таким образом, Slave завершает свое выполнение

4.5.1.7 Прием параметров команды

Параметры команд – это 4-х 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.

4.5.1.8 Сообщения об ошибках

Сообщения об ошибках – это 2-х 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.

4.6 Контроллер FLASH памяти программ

Микроконтроллер 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.

4.6.1 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 микроконтроллере реализуется Flash память с физической организацией 32К на 128 разрядов. Таким образом, за 40 нс из Flash памяти извлекается 16 байт, в которых может быть закодировано от 4 до 8 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 4–19 – 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.

Так же в контроллере EEPROM реализованы раздельные кэш для данных и для инструкций, объемом 16 x 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 сответственно. 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.

4.6.2 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 EERPOM_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.

Bank 31 1К x 128 16K x 8	0x0801_FFFC … 0x0801_F00C	0x0801_FFF8 … 0x0801_F008	0x0801_FFF4 … 0x0801_F004	0x0801_FFF0 … 0x0801_F000
	…	…	…	…
Bank 1 1К x 128 16K x 8	0x0800_1FFC … 0x0800_100C	0x0800_1FF8 … 0x0800_1008	0x0800_1FF4 … 0x0800_1004	0x0800_1FF0 … 0x0800_1000
Bank 0 1К x 128 16K x 8	0x0800_0FFC … 0x0800_001C 0x0800_000C	0x0800_0FF8 … 0x0800_0018 0x0800_0008	0x0800_0FF4 … 0x0800_0014 0x0800_0004	0x0800_0FF0 … 0x0800_0010 0x0800_0000
	Sector_D 1K x 32 4K x 8	Sector_C 1K x 32 4K x 8	Sector_B 1K x 32 4K x 8	Sector_A 1K x 32 4K x 8

Основная память (IFREN=0)

Bank 0 1К x 128 16K x 8	0x0800_0FFC … 0x0800_001C 0x0800_000C	0x0800_0FF8 … 0x0800_0018 0x0800_0008	0x0800_0FF4 … 0x0800_0014 0x0800_0004	0x0800_0FF0 … 0x0800_0010 0x0800_0000
	Sector_D 1K x 32 4K x 8	Sector_C 1K x 32 4K x 8	Sector_B 1K x 32 4K x 8	Sector_A 1K x 32 4K x 8

Информационная память (IFREN=1)

Рисунок 4–12 – Структура памяти Flash

4.6.2.1 Стирание всей памяти

Стирание памяти возможно только в режиме программирования. Для стирания всей памяти надо установить необходимое значение в бит IFREN (1 – для основной и информационной памяти и 0 – для основной памяти), затем установить биты XE, MAS1 и ERASE в единицу, и спустя время tnvs = 5 мкс установить бит NVSTR в единицу. Полное стирание памяти длится время tme = 40 мс. Спустя это время необходимо очистить бит ERASE, и спустя время tnvh1 = 100 мкс очистить биты XE, MAS1 и NVSTR. Последующие операции с память можно выполнять спустя время trcv = 1 мкс. Временная диаграмма стирания памяти представлена далее (см. Рисунок 4–13). При стирании информационной области, автоматически стриается и основная.

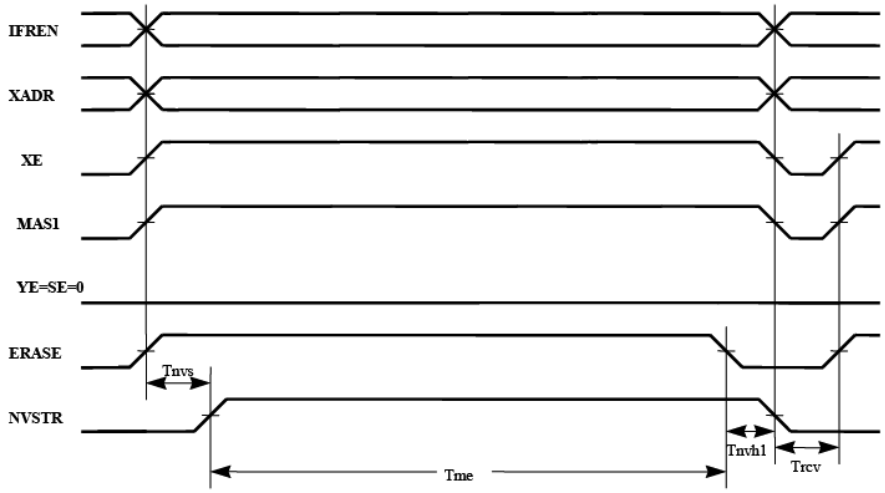


Рисунок 4–13 – Временная диаграмма стирания памяти

4.6.2.2 Стирание страницы памяти размером 4 Кбайт

Стирание 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 и установить биты XE и ERASE в единицу, и спустя время tnvs = 5 мкс установить бит NVSTR в единицу. Стирание страницы памяти длится время terase = 40 мс. Спустя это время необходимо очистить бит ERASE, и спустя время tnvh = 5 мкс очистить биты XE и NVSTR. Последующие операции с памятью можно выполнять спустя время trcv = 1 мкс. Временная диаграмма стирания страницы памяти представлена ниже (Рисунок 4–14).

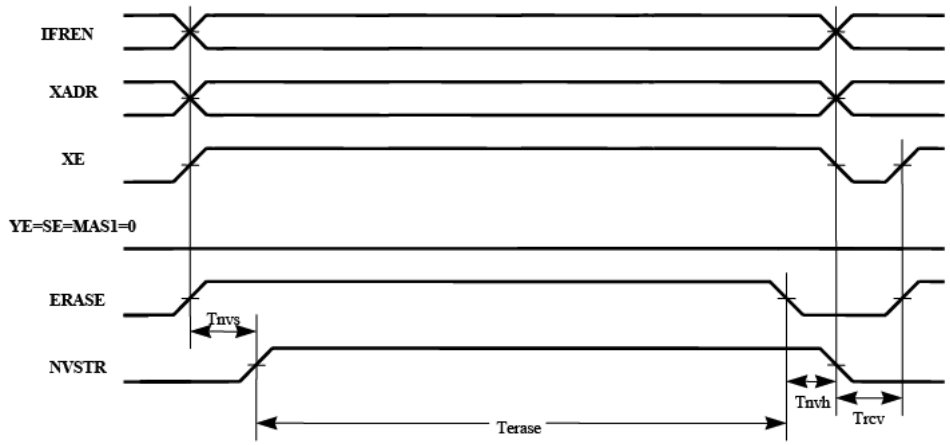


Рисунок 4–14 – Временная диаграмма стирания страницы памяти

4.6.2.3 Запись 32-х битного слова в память

Запись в память возможно только в режиме программирования. 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 записать записываемое в память слово и установить биты XE и PROG в единицу, и спустя время tnvs = 5 мкс установить бит NVSTR в единицу. Спустя время tpgs = 10 мкс установить бит YE в единицу. Запись в память длиться время tprog = 40 мкс. Спустя это время необходимо очистить бит YE, и спустя время tadh = 20 нс установить новый адрес и значение для записи в другую ячейку памяти. И спустя tadh = 20 нс установить YE в единицу и записать следующую слово. Если запись больше не требуется, то спустя время tpgh = 20 нс после очистки бита YE необходимо очистить бит PROG и спустя время tnvh = 5 мкс очистить биты XE и NVSTR.. Последующие операции с память можно выполнять спустя время trcv = 1 мкс. Временная диаграмма записи памяти представлена ниже (Рисунок 4–15).

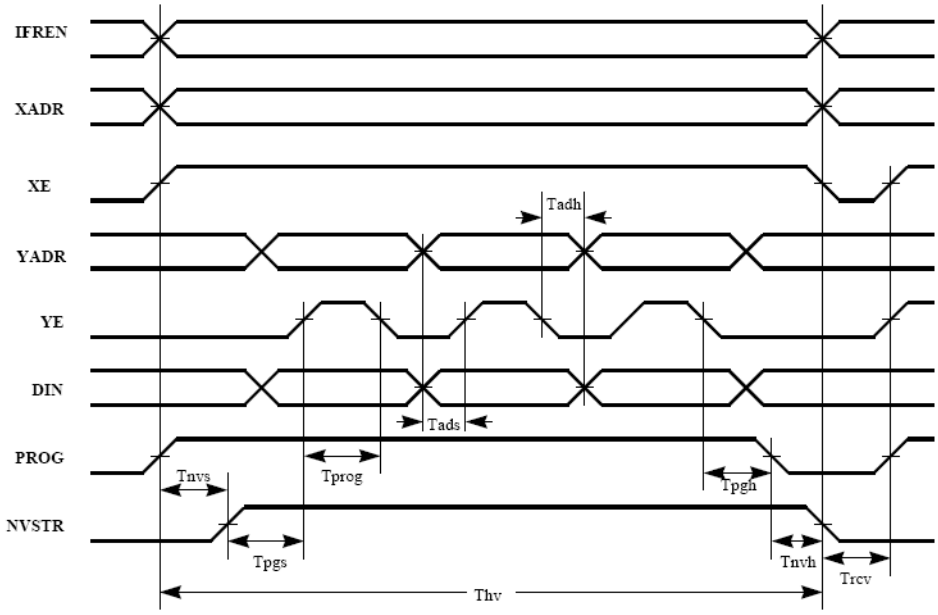


Рисунок 4–15 – Временная диаграмма записи памяти

4.6.2.4 Чтение 32-х битного слова из памяти

В обычном режиме работы для чтения доступна только основная память. Для этого 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 единицу, и спустя время txa = 30 нс из регистра EEPROM_DO можно считать данные. Если необходимо считать следующее слово, то в регистр EEPROM_ADR необходимо записать новый адрес и спустя время txa = 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 представлена ниже (Рисунок 4–16).

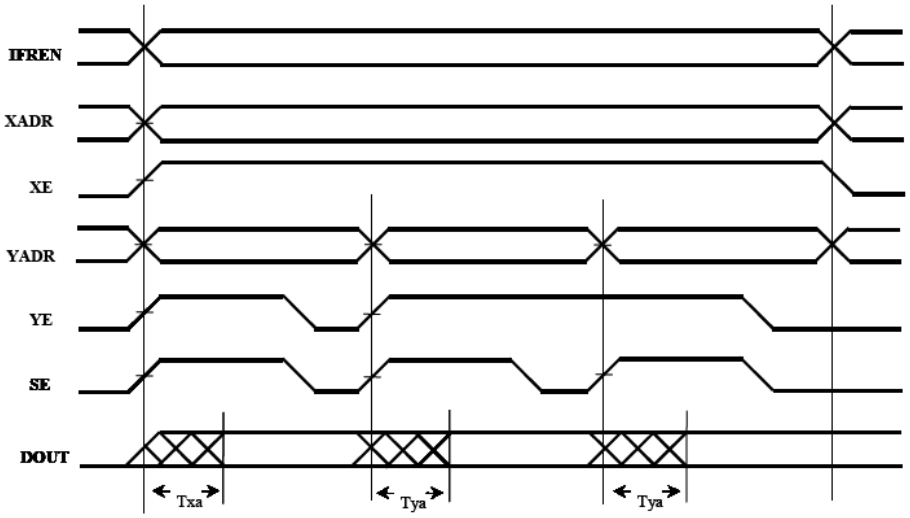


Рисунок 4–16 – Временная диаграмма чтения памяти

Flash память программ поддерживает до 20 000 тысяч циклов перезаписи. Нельзя повторять циклы стирания – записи и стирания – стирания одной ячейки памяти с периодом менее 4 мс.

4.6.3 Описание регистров управления контроллера Flash памяти программ

Таблица 4–20 предоставляет перечень регистров управления контроллера Flash-памяти программ.

Таблица 4–20 – 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.

Таблица 4–21 – Регистр команды EERPOM_CMD

Номер	31…14	13	12	11	10
Доступ	U	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0
		NVSTR	**PROG**	**MAS1**	**ERASE**

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	R/W
Сброс	0	0	0	0	100	0	0	0
	IFREN	**SE**	**YE**	**XE**	**Delay[2:0]**	**RD**	**WR**	**CON**

Таблица 4–22 – Описание бит регистра EEPROM_CMD

№ 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 EERPOM_DI: 0 – нет записи; 1 – есть запись
11	MAS1	Стереть весь блок, при ERASE =1: 0 – нет стирания; 1 – стирание
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
2	RD	Чтение из памяти EERPOM (в режиме программирования):

		0 – нет чтения; 1 – есть чтение
1	WR	Запись в память EERPOM (в режиме программирования): 0 – нет записи; 1 – есть запись
0	CON	Переключение контроллера памяти EEPROM на регистровое управление, не может производиться при исполнении программы из области EERPOM: 0 – управление EERPOM от ядра, рабочий режим; 1 – управление от регистров, режим программирования

4.6.3.1 MDR_EEPROM->ADR

Таблица 4–23 – Регистр адреса EERPOM_ADR

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

Таблица 4–24 – Описание бит регистра адреса EEPROM_ADR

№ 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

4.6.3.2 MDR_EEPROM->DI

Таблица 4–25 – Регистр записываемых данных EERPOM_DI

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

Таблица 4–26 – Описание бит регистра записываемых данных EERPOM_DI

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31...0	DATA[31:0]	Данные для записи в EERPOM

4.6.3.3 MDR_EEPROM->DO

Таблица 4–27 – Регистр считываемых данных EERPOM_DO

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

Таблица 4–28 – Описание бит регистра считываемых данных EERPOM_DO

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**

31...0	DATA[31:0]	Данные, считанные из EERPOM

4.6.3.4 MDR_EEPROM->KEY

Таблица 4–29 – Регистр ключа EEPROM_KEY

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

Таблица 4–30 – Описание бит регистра ключа EEPROM_KEY

№ 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 EERPOM_KEY записать комбинацию 0x8AAA5551

4.6.3.5 MDR_EEPROM-> CTRL

Таблица 4–31 – Регистр управления EERPOM_CTRL

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

Таблица 4–32 – Описание бит регистра EEPROM_CTRL

№ 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;

5 Процессорное ядро RISC

5.1 Структурная схема процессорного ядра RISC

- Процессорное ядро RISC – высокопроизводительный 32-х 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 интегрированными блоком защиты памяти MPU.

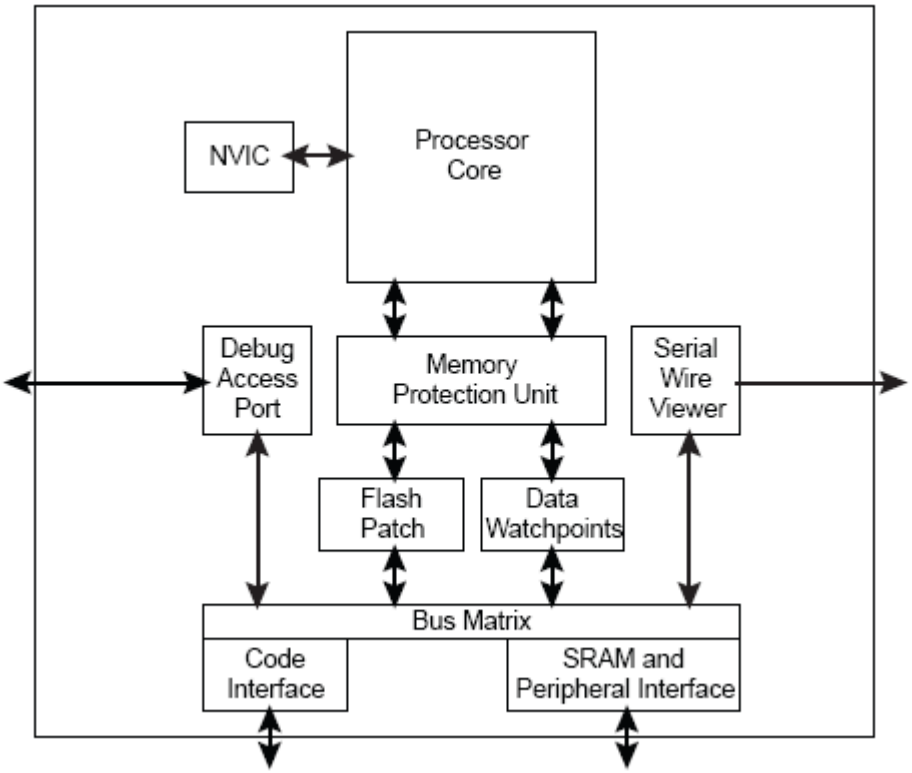


Рисунок 5–1 – Структурная схема процессорного ядра

Процессор построен на высокопроизводительном ядре с 3-х стадийным конвейером и Гарвардской архитектурой, что делает его идеальным для микроконтроллерных приложений. Процессор предлагает превосходную энергоэффективность, эффективный набор инструкций, оптимальный дизайн аппаратных средств, включающих однотактную инструкцию умножения 32x32 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-х битных архитектур с размером кода, сравнимым с 8-ми и 16-ти 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 инструкций 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 JTAG разъем или 2-х проводной 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-х 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.

5.2 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 Thread режим.

Уровни привилегий при исполнении программ:

- Unprivileged
  Программное обеспечение:
  - имеет ограниченный доступ к MSR и MRS инструкциям, и не может использовать CPS инструкцию;
  - не имеет доступа к системному таймеру, NVIC и блоку системного управления;
  - может иметь пониженный уровень доступа к памяти или периферии.

Непривилегированное программное обеспечение исполняется с уровнем *unprivileged*.

- Privileged
  Программное обеспечение имеет полный доступ ко всем инструкциям и ресурсам.

Привилегированное программное обеспечение исполняется с уровнем *privileged*.

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

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

5.3 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 инкрементируется указатель и затем записывается новый элемент в память. Процессор реализует два стека – *main* и *process* с независимыми указателями стеков, подробнее смотрите указатели стека.

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

Таблица 5–1 – Режимы работы процессора при выполнении программы

Режим процессора	Использование	Уровни привилегии для программного обеспечения	Используемый стек
Thread	Выполнение приложений	*Privileged* или *Unprivileged* (1)	*Main* или *Process* (1)
Handler	Обработка исключений	Всегда *Privileged*	*Main* стек

1 – Подробнее см. описание регистра CONTROL.

5.4 Регистры ядра

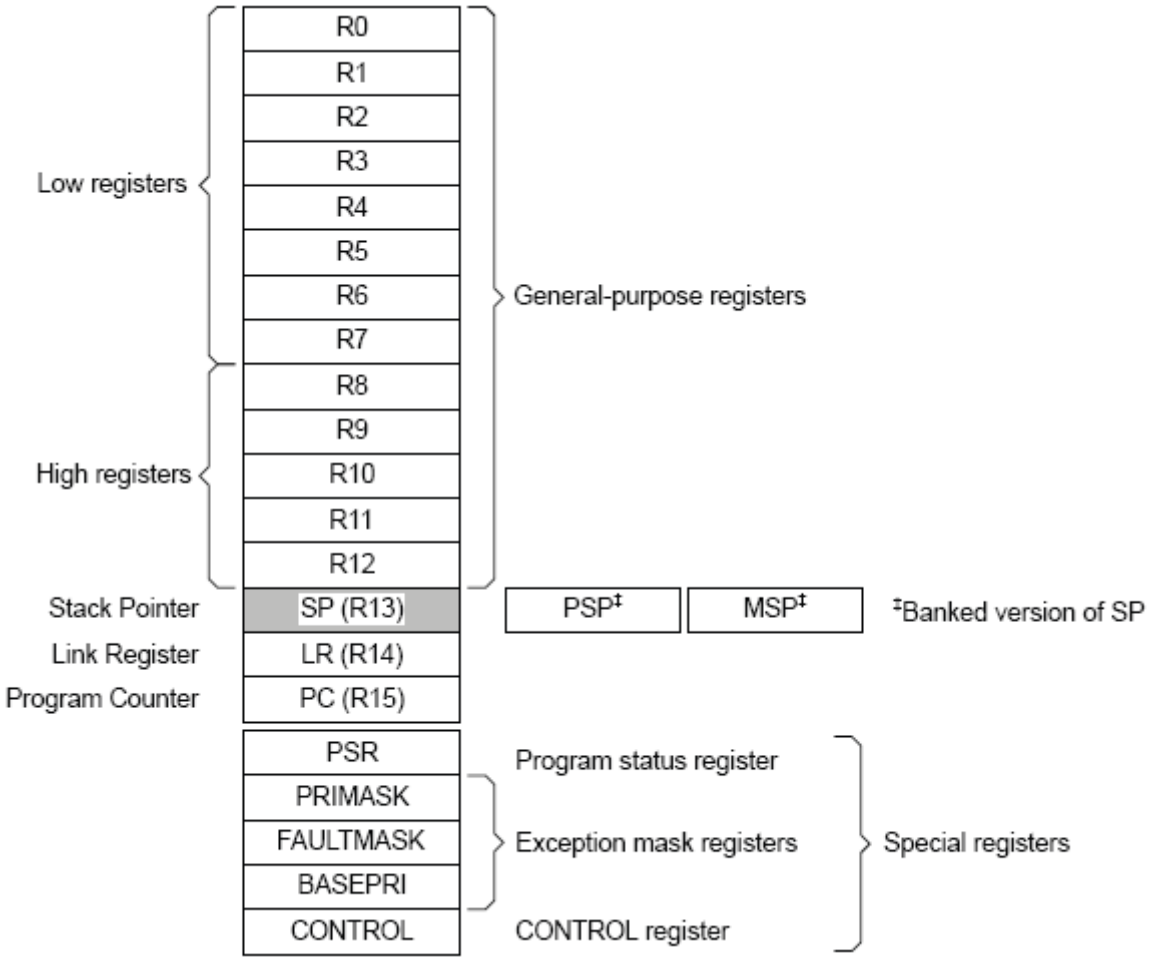


Рисунок 5–2 – Регистры ядра

Таблица 5–2 – Сводная таблица регистров ядра

Название	Тип[1]	Требуемый уровень привилегий	Значение после сброса	Описание
R0-R12	RW	Оба[2]	Неизвестно	Регистры общего назначения
MSP	RW	Privileged	См. описание	Указатель стека *main* Stack Pointer
PSP	RW	Оба[2]	Неизвестно	Указатель стека *process* Stack Pointer
LR	RW	Оба[2]	0xFFFFFFFF	Регистр связи Link Register
PC	RW	Оба[2]	См. описание	Счетчик команд Program Counter
PSR	RW	Privileged	0x01000000	Программный регистр состояния Program Status Register
ASPR	RW	Оба[2]	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

1 – Определяет режим доступа при исполнении программы в *thread* и *handler* режимах. В режиме отладки может отличаться;

2 – Регистр доступен при исполнении программы с обоими уровнями привилегий

5.4.1 Регистры общего назначения R0-R12

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

5.4.2 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.

5.4.3 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.

5.4.4 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.

5.4.5 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-х 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 MRS инструкцией;
- записать только в APSR, используя APSR с MSR инструкцией.

Таблица 5–3 – 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 IPSR биты
2 – При чтении EPSR бит читаются нули, и запись в них игнорируется.

Подробнее в описании инструкции MRS и MSR.

5.4.6 Программный регистр состояния приложения APSR

Регистр APSR содержит текущие флаги состояния выполнения предыдущей инструкции.

Таблица 5–4 – 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 5–5 – Описание бит регистра APSR

№ 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

5.4.7 Программный регистр состояния прерываний IPSR

Регистр IPSR содержит номер типа исключения для текущего обработчика прерывания.

Таблица 5–6 – Регистр IPSR

Номер	31…9	8…0
Доступ	-	RO
Сброс	-	0
	-	**ISR_NUMBER**

Таблица 5–7 – Описание бит регистра IPSR

№ 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; 4 – Memory Management Fault; 5 – Bus Fault; 6 – Usage Fault; 7…10 – зарезервировано; 11 – SVCall; 12 – зарезервировано для отладки; 13 – PendSV; 15 – SysTick; 16 – IRQ0; … 48 – IRQ31. Более подробно в разделе «Прерывания и исключения»

5.4.8 Программный регистр состояния выполнения EPSR

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

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

Таблица 5–8 – 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 5–9 – Описание бит регистра EPSR

№ 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 MSR инструкцию всегда возвращает ноль. Попытка записать EPSR используя MSR напрямую приложением игнорируется. Обработчик сбойной ситуации может считать значение EPSR сохранив его в стеке для отображения операции вызвавшей сбой. Подробнее раздел вход и выход в исключения.

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.

If-Then блок инструкций

Блок If-Then содержит до 4 инструкций, следующих за 16-ти битной инструкцией IT. Каждая инструкция в этом блоке условная. Условие для инструкций может быть одним либо обратным для некоторых из них. Подробнее смотрите в разделе инструкция IT.

5.4.9 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 разделе инструкции MSR, MRS и CPS.

5.4.10 Регистр маски приоритетов Priority Mask

Регистр PRIMASK предотвращает активацию всех исключений с конфигурируемым приоритетом.

Таблица 5–10 – Регистр PRIMASK

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

Таблица 5–11 – Описание бит регистра PRIMASK

№ 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

5.4.11 Регистр маски сбоев Fault Mask

Регистр FAULTMASK предотвращает активацию всех исключений.

Таблица 5–12 – Регистр FAULTMASK

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

Таблица 5–13 – Описание бит регистра FAULTMASK

№ 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.

5.4.12 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 5–14 – Регистр BASEPRI

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

Таблица 5–15 – Описание бит регистра BASEPRI

№ 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 смотрите Таблицу 34–8 – Регистры приоритета прерываний. Помните, что большее значение в поле приоритета соответствует меньшему приоритету обработчика.

5.4.13 Регистр управления CONTROL

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

Таблица 5–16 – Регистр CONTROL

Номер	31…2	1	0
Доступ	U	R/W	R/W
Сброс	0	0	0
	-	**Active Stack Pointer**	**Thread Mode Privilige Level**

Таблица 5–17 – Описание бит регистра CONTROL

№ 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 Privilige Level	0 – привелегирован. 1 – непривелегирован

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

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

По определению, режим Thread использует MSP. Для переключения указателя стека в Thread режиме 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 ISB инструкцию немедленно за MSR инструкцией. Это позволяет быть уверенным, что инструкции, следующие за выполнением ISB, будут использовать новый указатель стека. Смотрите описание инструкции ISB.

5.5 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 п. «Контроллер прерываний NVIC» 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 little-endian режиме. Доступ в память инструкций и Private Peripheral Bus (PPB) всегда в little-endian режиме.

6 Система команд RISC

В процессоре реализована версия системы команд Thumb. Поддерживаемые команды представлены в Таблица 6–1.

В таблице используются следующие обозначения:

- в угловых скобках <> 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 6–1 – 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	-	
DMB	-	Барьер синхронизации доступа к памяти данных	-	

Мнемокод команды	Операнды	Краткое описание	Флаги	Прим.
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	-	
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	-	

Мнемокод команды	Операнды	Краткое описание	Флаги	Прим.
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	-	
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	-	

Мнемокод команды	Операнды	Краткое описание	Флаги	Прим.
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	-	

6.1 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 6–2 – 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 6–3 – 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)

6.2 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.

6.2.1 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. п. “Формат второго операнда”.

6.2.2 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.

6.2.3 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 параметров сдвига.

6.2.3.1 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;
- любая константа в виде 0x00XY00XY;
- любая константа в виде 0xXY00XY00;

- любая 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 разделе «Константа», ассемблер может сгенерировать код с использованием другой инструкции, обеспечивающей необходимую функциональность.

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

6.2.3.2 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, $1 \le n \le 32$;

LSL #n – логический сдвиг влево на n бит, $1 \le n \le 31$;

LSR #n – логический сдвиг вправо на n бит, $1 \le n \le 32$;

ROR #n – циклический сдвиг вправо на n бит, $1 \le n \le 31$;

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 разделе "Операции сдвига".

6.2.4 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 (Rm – сдвигаемый регистр, n – длина сдвига).

6.2.4.1 ASR

Арифметический сдвиг вправо на n бит переносит крайние слева 32-n бит регистра Rm вправо на n позиций, то есть на место крайних справа 32-n. Бит [31] исходного значения регистра записывается в n крайних слева бит результата. Смотрите Рисунок 6–1.

Операцию ASR # n можно использовать для деления значения регистра Rm 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 Rm.

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

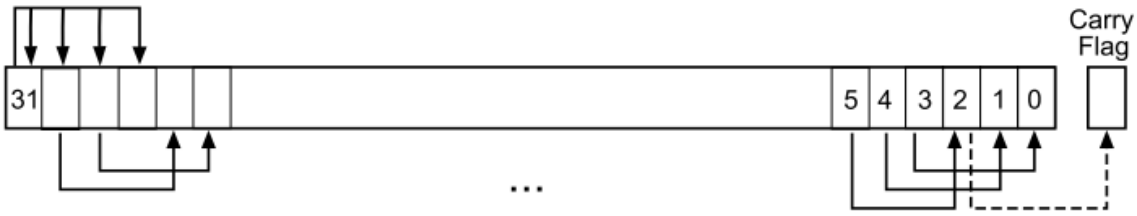


Рисунок 6–1 – Инструкция ASR # 3

6.2.4.2 LSR

Логический сдвиг вправо на n бит переносит крайние слева 32-n бит регистра Rm вправо на n позиций, то есть на место крайних справа 32-n. При этом в n крайних слева бит результата записывается 0. Смотрите Рисунок 6–3.

Операцию LSR # n можно использовать для деления значения регистра Rm на 2n 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 Rm.

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

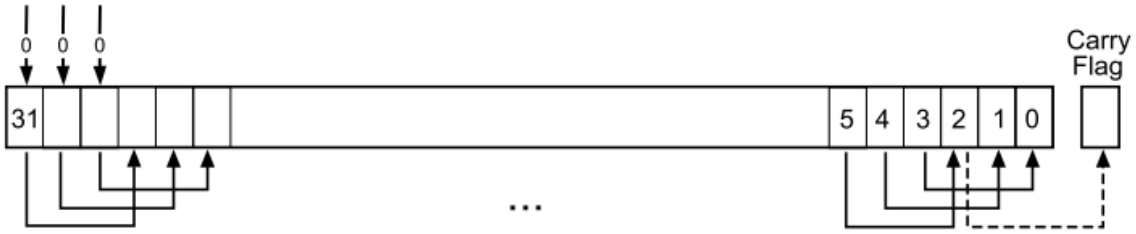


Рисунок 6–2 – Инструкция LSR # 3

6.2.4.3 LSL

Логический сдвиг влево на n бит переносит крайние справа 32-n бит регистра Rm влево на n позиций, то есть на место крайних слева 32-n. При этом в n крайних слева бит результата записывается 0. Смотрите Рисунок 6–3.

Операцию LSL # n можно использовать для умножения значения регистра Rm 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 Rm. Инструкция LSL #0 не влияет на значение флага переноса.

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

Рисунок 6–3 – Инструкция LSL # 3

6.2.4.4 ROR

Циклический сдвиг вправо на n бит переносит крайние слева 32-n бит регистра Rm 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. Смотрите Рисунок 6–4.

При использовании инструкции RORS, а также в случае, если сдвиг ROR #n используется при вычислении второго операнда команд MOVS, MVNS, ANDS, ORRS, ORNS, EORS, BICS, TEQ или TST, флаг переноса принимает значение последнего сдвинутого бита, то есть бита [n-1] регистра Rm.

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 Rm.

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

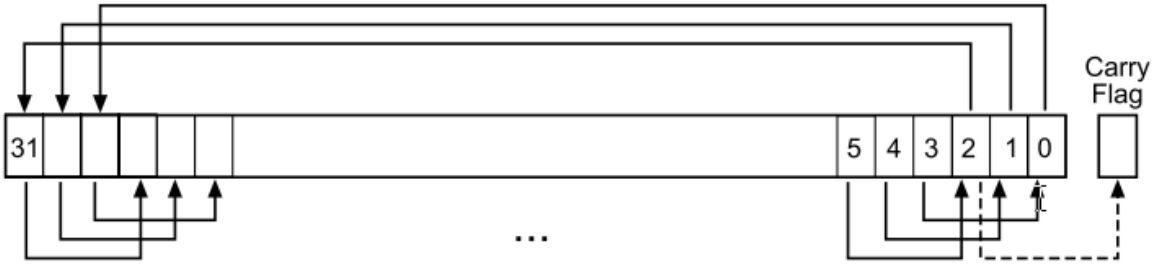


Рисунок 6–4 – Инструкция ROR # 3

6.2.4.5 RRX

Циклический сдвиг вправо на один бит с переносом переносит разряды регистра Rm вправо на одну позицию, при этом в позицию [31] результата записывается значение флага переноса. Смотрите Рисунок 6–5.

При использовании инструкции RRXS, а также в случае, если сдвиг RRX #n используется при вычислении второго операнда команд MOVS, MVNS, ANDS, ORRS, ORNS, EORS, BICS, TEQ или TST, флаг переноса принимает значение бита [0] регистра Rm.

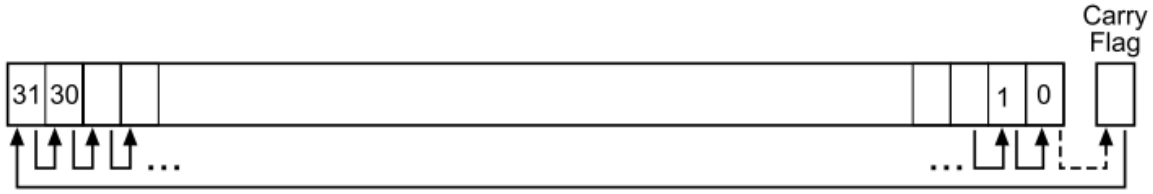


Рисунок 6–5 – Инструкция RRX

6.2.5 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 разделе "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. п. «SCB->CCR»).

6.2.6 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].

6.2.7 Условное исполнение

Большая часть команд обработки данных способна изменять значения флагов в регистре состояния прикладной программы (APSR) в зависимости от результата выполнения (см. «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. В Таблица 6–4 представлен список суффиксов, которые можно добавить к инструкции для того, чтобы сделать ее условной.

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 – IT-блок). Ассемблеры некоторых поставщиков могут самостоятельно вставлять инструкцию IT в случае, если программист разместит условную инструкцию за пределами IT-блока.

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.

6.2.7.1 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 «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.

6.2.7.2 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 6–4). Там же указаны соответствующие логические выражения для значений флагов.

Условные команды рекомендуется использовать для снижения количества ветвлений в программе.

Таблица 6–4 – 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 and Z = 0	Больше, беззнаковое сравнение
LS	C = 0 or Z = 1	Меньше или равно, беззнаковое сравнение
GE	N = V	Больше или равно, знаковое сравнение
LT	N != V	Меньше, знаковое сравнение
GT	Z = 0 and N = V	Больше, знаковое сравнение

Суффикс	Флаги	Значение
LE	Z = 1 and N != 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 = ABS(R1).

```
MOVS R0, R1 ; R0 = R1, установка флагов
IT MI; IT инструкция 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; IT инструкция 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
```

6.2.8 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
```

6.3 Команды доступа к памяти

Обобщенные данные о командах доступа к памяти представлены в Таблица 6–5.

Таблица 6–5 – 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	

6.3.1 ADR

Загрузка адреса, заданного относительно счетчика команд.

Синтаксис

ADR{cond} Rd, label

где:

cond – необязательный код условия, см. “Условное исполнение”.

Rd – регистр-получатель.

label – относительный адрес, см. "Адресация относительно счетчика команд".

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. «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.
```

6.3.2 LDR и STR, непосредственно заданное смещение

Загрузка или сохранение регистра в режиме адресации со смещением, адресации с пре-индексированием или адресации с пост-индексированием.

Синтаксис

```
op{type}{cond} Rt, [Rn {, #offset}]      ; адресация со смещением
op{type}{cond} Rt, [Rn, #offset]!        ; преиндексирование
op{type}{cond} Rt, [Rn], #offset         ; пост-индексирование
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         ; пост-индексирование, двойное слово
```

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. “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 Rn. Результат используется в качества адреса чтения или записи. Значение регистра Rn остается неизменным.
Синтаксис задания данного режима:
[Rn, #offset]

Адресация с пре-индексированием — Значение смещения добавляется к или вычитается из содержимого регистра Rn. Результат используется в качества адреса чтения или записи, а также записывается обратно в регистр Rn.
Синтаксис задания данного режима:
[Rn, #offset]!

Адресация с пост-индексированием — Содержимое регистра Rn используется в качества адреса чтения или записи. Значение смещения добавляется к или вычитается из содержимого регистра Rn, после чего записывается обратно в регистр Rn.
Синтаксис задания данного режима:
[Rn], #offset

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. “Выравнивание адресов”).

Таблица 6–6 показывает диапазоны значений смещения для различных форм адресации.

Таблица 6–6 – Диапазон значений смещения

Тип инструкции	Смещение	Преиндексирование	Пост-индексирование
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 пост-индексированием 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 IT-блоке.

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 пост-индексированием 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-struc]  ; const-struc - 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.
```

6.3.3 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. “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. “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 IT-блоке.

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]
                            ; Считать 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*R2.
```

6.3.4 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. “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 так же, как и инструкции доступа к памяти с непосредственно задаваемым смещением, см. "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 ре-
                      ; гистра R2 и 8 в регистр R2, с непривилегированным доступом.
```

6.3.5 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. “Условное исполнение”.

Rt – регистр, в который должна производиться загрузка.

Rt2 – второй регистр, в который должна производиться загрузка.

label – относительный адрес, см. "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. “Выравнивание адресов”).

Метка должна располагаться на ограниченном расстоянии от текущей инструкции. Таблица 6–7 показывает возможные значения смещений между меткой данных и текущим значением счетчика команд. Для использования больших смещений, возможно, придется указать суффикс .W размера команды (см. "Выбор размера кода инструкции").

Таблица 6–7 – 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. “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 Rn до Rn+4*(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 Rn+4*(n-1) записывается обратно в регистр Rn.

Команды LDMDB, LDMEA, STMDB и STMFD для доступа используют адреса памяти в интервале от Rn до Rn-4*(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 Rn-4*(n-1) записывается обратно в регистр Rn.

Инструкции PUSH и POP могут быть выражены через инструкции LDM и STM. Подробности см. в разделе “PUSH и POP”.

Ограничения

В описываемых в разделе командах:

- в качестве регистра Rn 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 Rn 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 IT-блоке.

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.
```

6.3.6 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. “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 номеров считывается из памяти с меньшим значением адресом.

Подробности см. в разделе “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}
```

6.3.7 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. “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.
```

6.3.8 CLREX

Сброс эксклюзивного доступа.

Синтаксис

CLREX{cond}

где:

cond – необязательный код условия, см. “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
```

6.4 Инструкции обработки данных

В Таблица 6–8 представлены инструкции обработки данных.

Таблица 6–8 – 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	

6.4.1 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:

op – 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. “Условное исполнение”.

cond – необязательный суффикс условного исполнения, см. “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. “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 IT-блоке.
- 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 находится 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.
```

6.4.2 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. “Условное исполнение”.

cond – необязательный суффикс условного исполнения, см. “Условное исполнение”.

Rd – регистр-получатель результата.

Rn – регистр, содержащий значение первого операнда.

Operand2 – второй операнд, см. “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. “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
```

6.4.3 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. “Условное исполнение”.

cond – необязательный суффикс условного исполнения, см. “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 разделе “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. “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.
```

6.4.4 CLZ

Определить количество ведущих нулей.

Синтаксис

CLZ{cond} Rd, Rm

где:

cond - необязательный суффикс условного исполнения, см. “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 только 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.
```

6.4.5 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. “Условное исполнение”.
R – регистр, содержащий первый операнд.
Operand2 – второй операнд. См. “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
```

6.4.6 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. “Условное исполнение”.

Cond – необязательный суффикс условного исполнения, см. “Условное исполнение”.

Rd – регистр-получатель результата.

Operand2 – второй операнд. См. “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 так же, 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. “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 0xFFFFFFF0 (инверсия значения 0x0F)
                   ; в регистр R2, установить флаги.
```

6.4.7 MOVT

Записать в старшее полуслово регистра.

Синтаксис

MOVT{cond} Rd, #imm16

где:

cond – необязательный суффикс условного исполнения, см. “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
```

6.4.8 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 полусловах;
- REVSH – изменить на обратный порядок байт в младшем полуслове, произвести распространение знакового бита в старшее полуслово.
- RBIT – изменить порядок бит в 32-разрядном слове.

cond – необязательный суффикс условного исполнения, см. “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 слове.

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 полуслове R0
REVSH R0, R5    ; Обратить полуслово 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
```

6.4.9 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. “Условное исполнение”.
Rn – регистр, содержащий первый операнд.
Operand2 – второй операнд. См. “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 Исключающее 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 Исключающее 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. “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.
```

6.5 Инструкции умножения и деления

В следующей таблице представлена информация о командах умножения и деления.

Таблица 6–9 – 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 x 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 x 32 + 64), 64-битный результат
UMULL	Умножение чисел без знака, 64-битный результат

6.5.1 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. “Условное исполнение”.
- cond – необязательный суффикс условного исполнения, см. “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 Rn и Rm, вычитает произведение из регистра Ra, после чего сохраняет 32 младших значащих бита результата в Rd.

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 Rd, Rn и Rm должны находиться в диапазоне от R0 до R7;
- регистр Rd должен совпадать с Rm;
- 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 R10, R2, R5          ; R10 = R2 x R5
MLA R10, R2, R1, R5      ; R10 = (R2 x R1) + R5
MULS R0, R2, R2          ; R0 = R2 x R2, установить флаги
MULLT R2, R3, R2         ; условное исполнение R2 = R3 x R2
MLS R4, R5, R6, R7       ; R4 = R7 - (R5 x R6)
```

6.5.2 UMULL, UMLAL, SMULL и SMLAL

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

Синтаксис

op{cond} RdLo, RdHi, Rn, Rm

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. “Условное исполнение”.

RdLo, RdHi – регистры-получатели младшей и старшей частей результата, соответственно. Для инструкций UMLAL и SMLAL они также содержат накапливаемое значение.

Rn, Rm – 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 x R6
SMLAL R4, R5, R3, R8    ; Операция со знаком (R5,R4) = (R5,R4) + R3 x R8
```

6.5.3 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. “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 Rn не делится нацело на число в Rm, 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.
```

6.6 Инструкции преобразования данных с насыщением

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

6.6.1 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. “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 $-2^{(n-1)} \le x \le 2^{(n-1)}-1$ 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 $0 \le x \le 2^n-1$ 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/
```

6.7 Команды работы с битовыми полями

В Таблица 6–10 показаны инструкции, позволяющие манипулировать последовательностями смежных бит данных в регистрах или битовых полях.

Таблица 6–10 – 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

6.7.1 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. “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.
```

6.7.2 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. “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.
```

6.7.3 SXT и UXT

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

Синтаксис

SXT*extend*{cond} {Rd,} Rm {, ROR #n}
UXT*extend*{cond} {Rd}, Rm {, ROR #n}

где:

Суффикс *extend* может принимать одно из следующих значений:

- B – преобразование 8-битного числа в 32-битное;

- H – преобразование 16-битного числа в 32-битное.

Cond – необязательный суффикс условного исполнения, см. “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.
```

6.8 Инструкции передачи управления

В таблице ниже представлен список инструкций передачи управления.

Таблица 6–11 – 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
TBH	Табличный переход по индексу, смещения – полуслова

6.8.1 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. “Условное исполнение”.
Label – относительный адрес, см. "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 находится за пределами IT-блока. Все остальные условно исполняемые инструкции передачи управления должны располагаться внутри IT-блока, а за пределами этого блока должны использоваться только в безусловной форме. Подробности см. в разделе "IT".

Ниже (Таблица 6–12) представлен диапазон адресуемых переходов для различных команд ветвления. Для достижения максимального диапазона может потребоваться указать суффикс .W размера инструкции. Подробности см. в разделе "Выбор размера кода инструкции".

Таблица 6–12 – Диапазон адресуемых переходов для команд ветвления

Инструкция	Диапазон адресации
B label	от -16 Мбайт до +16 Мбайт относительно текущей позиции
B cond label (вне IT-блока)	от -1 Мбайт до +1 Мбайт относительно текущей позиции
B cond label (внутри IT-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 IT-блока любая из инструкций ветвления должна располагаться последней.
- B cond – единственная условно исполняемая команда, которую допустимо использовать за пределами IT-блока. Тем не менее, внутри IT-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.
```

6.8.2 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.
```

6.8.3 IT

Начало блока условно исполняемых инструкций.

Синтаксис

IT{x{y{z}}} cond

где:

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

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 IT-блоке на месте cond условие AL (всегда истинное). В этом случае все инструкции в IT-блоке 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 IT-блок.

Мнемокоды команд внутри IT-блока, в том числе и команд ветвления, должны включать в себя суффикс условного исполнения {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 BKPT внутри IT-блока всегда выполняется, вне зависимости от истинности или ложности условия.

Обработка исключений внутри IT-блока, 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 IT-блока продолжается с прерванной позиции. Это единственный допустимый способ передачи управления внутрь IT-блока 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 IT-блока, либо на последнюю инструкцию IT-блока.
- 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 IT-блока, за исключением случая возврата из обработчика исключения;
- все условные инструкции, за исключением B cond, должны находится внутри IT-блока. Команда B cond может быть расположена как внутри, так и вне IT-блока, однако внутри IT-блока она обеспечивает более широкий диапазон адресуемых переходов;
- каждая инструкция внутри IT-блока должна быть снабжена суффиксом условного исполнения с кодом, либо совпадающим, либо противоположным коду условия IT-блока.

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

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

```
ITTE NE                 ; Следующие три инструкции - условные
```

```
ANDNE R0, R0, R1        ; ANDNE не изменяет состояние флагов
ADDSNE R2, R2, #1       ; ADDSNE 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                   ; IT-блок 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 IT-блоке.
```

6.8.4 TBB и TBH

Табличный переход по индексу.

Синтаксис

TBB [Rn, Rm]
TBH [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 TBB или TBH.

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 TBB) или полуслова (для команды TBH).

Регистр Rn содержит указатель на начало таблицы, а регистр Rm – индекс требуемого элемента.

Для команды TBB смещение вычисляется путем умножения на два значения байта из заданной ячейки таблицы, интерпретируемого как целое число без знака.

Для команды TBH 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 TBB или TBH.

Ограничения

- в качестве регистра Rn нельзя использовать SP;
- в качестве регистра Rm нельзя использовать SP и PC;
- при использовании инструкций TBB или TBH внутри IT-блока они должны быть последней командой блока.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

```
        ADR.W R0, BranchTable_Byte
        TBB [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
```

6.9 Прочие инструкции

В таблице ниже представлен список инструкций процессора RISC, не рассмотренных в предыдущих разделах.

Таблица 6–13 – 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

6.9.1 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 разделе “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 IT-блока.

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.
```

6.9.2 DMB

Барьер синхронизации доступа к памяти данных.

Синтаксис

DMB{cond}

где:

cond – необязательный код условия, см. “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.
```

6.9.3 DSB

Барьер синхронизации доступа к памяти данных.

Синтаксис

DSB{cond}

где:

cond – необязательный код условия, см. “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.
```

6.9.4 ISB

Барьер синхронизации доступа к инструкциям.

Синтаксис

ISB{cond}

где:

cond – необязательный код условия, см. “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.
```

6.9.5 MRS

Считать содержимое специального регистра в регистр общего назначения.

Синтаксис

MRS{cond} Rd, spec_reg

где:

cond – необязательный код условия, см. “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.
```

6.9.6 MSR

Записать регистр общего назначения в специальный регистр.

Синтаксис

MSR{cond} spec_reg, Rn

где:

cond – необязательный код условия, см. “Условное исполнение”.

Rn – регистр-источник данных.

spec_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. “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
```

6.9.7 NOP

Нет операции.

Синтаксис

NOP{cond}

где:

cond – необязательный код условия, см. “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
```

6.9.8 SEV

Установить признак события.

Синтаксис

SEV{cond}

где:

cond – необязательный код условия, см. “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 раздел «Управление электропитанием».

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

```
SEV   ; Послать признак события.
```

6.9.9 SVC

Вызов супервизора.

Синтаксис

SVC{cond} #imm

где:

cond – необязательный код условия, см. “Условное исполнение”.

Imm – константное выражение, целое число в диапазоне от 0 до 255 (8-битное 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.
```

6.9.10 WFE

Ожидать событие.

Синтаксис

WFE{cond}

где:

cond – необязательный код условия, см. “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 разделе "Управление электропитанием".

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

```
WFE          ; Ожидание события.
```

6.9.11 WFI

Ожидание прерывание.

Синтаксис

WFI{cond}

где:

cond – необязательный код условия, см. “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, разрешон или запрещен этот режим.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

```
WFI    ; Ожидание прерывания.
```

7 Системный таймер SysTick

Процессор имеет 24-х 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 счёт продолжается по последующему фронту.

Когда процессор остановлен для отладки, таймер не декрементируется.

7.1 Описание регистров системного таймера SysTick

Таблица 7–1 – Описание регистров системного таймера SysTick

Адрес	Название	Тип	Доступ	Значение после сброса	Описание
0xE000E000	SysTick				Системный таймер SYSTICK
0x000	CTRL	RW	привилегированный	0x00000004	SysTick->CTRL
0x004	LOAD	RW	привилегированный	0x00000000	SysTick->LOAD
0x008	VAL	RW	привилегированный	0x00000000	SysTick->VAL
0x00C	CALIB	RO	привилегированный	0x00002904[(1)]	SysTick->CAL

[1)] Калибровочное значение системного таймера.

7.1.1 SysTick->CTRL

Регистр CTRL разрешает основные функции системного таймера.

Назначение бит:

Таблица 7–2 – Регистр контроля и статуса CTRL

Номер	31…17	16	15…3	2	1	0
Доступ						
Сброс						
	-	COUNTFLAG	-	CLKSOURCE	TICKINT	ENABLE

COUNTFLAG

Возвращает 1, если таймер досчитал до нуля с последнего момента чтения.

CLKSOURCE

Указывает источник синхросигнала:

0 – LSI

1 – HCLK

TCKINT

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 счёт.

7.1.2 SysTick->LOAD

Регистр LOAD устанавливает стартовое значение, загружаемое в регистр VAL.

Таблица 7–3 – Регистр перегружаемого значения LOAD

Номер	31…24	23…0
Доступ		
Сброс		
	-	**RELOAD**

RELOAD

Значение, загружаемое в регистр VAL, когда таймер разрешён и когда достигается значение нуля.

Расчёт 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.

Расчёт 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.

7.1.3 SysTick->VAL

Регистр VAL содержит текущее значение системного таймера.

Таблица 7–4 – 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.

7.1.4 SysTick->CAL

Регистр CALIB показывает калибровочное значение системного таймера.

Таблица 7–5 – 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 0x0002904.

Калибровочное значение фиксировано и равно 0x0002904 (10500), что позволяет генерировать базовое время 1 мс с частотой 10,5 МГц (84/8=10,5 МГц).

7.2 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.

8 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 (перекрытие) 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 запрещённую 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. раздел “Регионы памяти, типы и атрибуты RISC”.

Таблица 8–1 показывает возможные атрибуты MPU регионов. Здесь включены такие атрибуты памяти, как *shareable* и кэшируемость, которые не существенны во многих реализациях микроконтроллеров.

Таблица 8–1 – 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

8.1 Описание регистров MPU

Применяются следующие MPU регистры для определения регионов и их атрибутов.

Таблица 8–2 – 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

8.1.1 MPU->TYPE

Регистр TYPE показывает, присутствует или нет MPU, и как много регионов поддерживается.

Таблица 8–3 – 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 раздельная карта памяти для инструкций и данных:

0 – унифицированная.

8.1.2 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 8–4 – Регистр CTRL

Номер	31…4	3	2	1	0
Доступ					
Сброс					
	-	**PRIVDEFENA**		**HFNMIENA**	**ENABLE**

PRIVDEFENA

Разрешает привилегированный программный доступ к default карте памяти:
0 – если MPU разрешон, запрещение применяется к default карте памяти. Любой доступ к памяти, не покрываемой разрешённым регионом, вызывает ошибку;
1 – если MPU разрешон, разрешает применение default карты памяти как фонового региона для привилегированного программного доступа.

Когда разрешено, фоновый регион считается как номер региона -1. Любой регион, который определён и разрешен, имеет приоритет выше этой default памяти.

Если MPU запрещён, то процессор игнорирует этот бит.

HFNMIENA

Разрешает операции MPU во время возникновения аппаратной ошибки, NMI, и FAULTMASK обработчик.

Если MPU разрешён:
0 – MPU запрещён во время возникновения аппаратной ошибки, NMI, FAULTMASK обработчик, несмотря на значения бита ENABLE;
1 – MPU разрешён во время возникновения аппаратной ошибки, NMI, FAULTMASK обработчик.

Если MPU запрещён и этот бит устанавливается в единицу, то поведение непредсказуемо.

ENABLE

Разрешает MPU:
0 – MPU запрещён;
1 – MPU разрешён.

Если ENABLE и PRIVDEFENA одновременно установлены в единицу:

Любой неадресованый доступ привилегированным программным обеспечением к разрешённому региону, ведёт себя как определено default картой памяти. Любой неадресованый доступ непривилегированным программным обеспечением к разрешённому 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 единицу, по крайней мере, один регион карты памяти должен быть разрешён для системных функций, за исключением PRIVDEFENA установлен в единицу. Если PRIVDEFENA установлен в единицу и нет

разрешённых регионов, тогда только привилегированное программное обеспечение может исполняться.

Когда ENABLE установлен в ноль, 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 разрешён, доступ к системному пространству управления и таблице векторов всегда разрешён. 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 разрешён. Установка бита HFNMIENA в единицу разрешает действовать этим двум приоритетам.

8.1.3 MPU->RNR

Регистр RNR выбирает, на какой регион памяти ссылаются регистры RBAR и RASR.

Таблица 8–5 – 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 разрешённое 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.

8.1.4 MPU->RBAR

Регистр RBAR определяет базовый адрес MPU региона, выбранного RNR, вы можете изменить значение RNR. Запись RBAR с битом VALID установленным в единицу изменяет текущий номер региона и обновляет RNR.

Таблица 8–6 – 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 смотрите раздел “Поле ADDR”.

VALID

Бит верности номера региона MPU:
Запись:
0 – RNR не изменяется, и процессор:
- обновляет базовый адрес для региона, определённого в RNR;
- игнорирует значение поля REGION.

1 – процессор:
- обновляет значение RNR на значение из поля REGION;
- обновляет базовый адрес региона, определённого 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 определён в регистре RNR.

Поле ADDR

Поле ADDR это [31:N] бит регистра RBAR. Размер региона определяется полем SIZE в регистре RASR, как

N = Log2 (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.

8.1.5 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 8–7 – Назначение бит регистра RASR

Номер	31	30	29	28	27	26	24	23	22	21	19	18	17	16	15	8	7	6	5	1	0
Доступ																					
Сброс																					
	-		**XN**	**-**		**AP**		**-**		**TEX**		**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 8–11 – Кодирование привилегий доступа в поле AP.

TEX, C, B

Атрибуты доступа к памяти, см. Таблица 8–9 – Кодирование бит разрешения доступа.

S

Бит общего доступа, см. Таблица 8–8 – Пример значений поля SIZE.

SRD

Бит запрещения подрегиона. Для каждого бита в этом поле:
0 – соответствующий подрегион разрешон;
1 – соответствующий подрегион запрещён.

Для более подробной информации см. раздел “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 разрешённое значение 3(b00010), для более подробной информации см. раздел “Значения поля SIZE”.

ENABLE

Бит разрешения региона.
Для более подробной информации о разрешении доступа, см. раздел “Атрибуты разрешения доступа MPU”.

Значения поля SIZE

Поле SIZE определяет размер памяти MPU региона выбранного регистром RNR следующим образом:

(Region size in bytes)= 2(SIZE+1)

Наименьший разрешенный размер региона 32 байт, соответствует значению SIZE, равному 4. В Таблица 8–8 представлены примеры значений SIZE, соответствующие размеру региона и значению N регистра RBAR.

Таблица 8–8 – Пример значений поля SIZE

Значение SIZE	Размер региона	Значение N[(1)]	Комментарий
b00100 (4)	32 байт	5	Минимальный разрешонный размер
b01001 (9)	1 кбайт	10	-
b10011 (19)	1 Мбайт	20	-
b11101 (29)	1 Гбайт	30	-
b11111 (31)	4 Гбайт	b01100	Максимальный разрешонный размер

[1)]. Содержится в RBAR, см. раздел “MPU->RBAR”.

8.1.6 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 8–9 – Кодирование бит разрешения доступа TEX, C, B, S

TEX	C	B	S	Тип памяти	Возможность общего доступа	Другие атрибуты
b000	0	0	x[1]	Строго упорядоченная	Общий доступ	
		1	x[1]	Устройство	Общий доступ	
	1	0	0	Обычная	Не общий доступ	Внешний и внутренний кэш, синхронное обновление памяти. Запись без кэширования
			1		Общий доступ	
		1	0	Обычная	Не общий доступ	Внешний и внутренний кэш, отложенное обновление памяти. Запись без кэширования
			1		Общий доступ	
b001	0	0	0	Обычная	Не общий доступ	
			1		Общий доступ	
		1	x[1]	Зарезервировано		-
	1	0	x[1]	Реализация определяется атрибутами		-
		1	0	Обычная	Не общий доступ	Внешний и внутренний кэш, отложенное обновление памяти. Запись и чтение пакетные
			1		Общий доступ	
b010	0	0	x[1]	Устройство	Не общий доступ	Индивидуальное устройство
		1	x[1]	Зарезервировано		-
	1	x[1]	x[1]	Зарезервировано		-
b1BB	A	A	0	Обычное	Не общий доступ	
			1		общий доступ	

[1] MPU игнорирует значение этих бит.

Таблица 8–10 поясняет кодирование режима кэша атрибутом TEX в диапазоне значений атрибута от 4 до 7.

Таблица 8–10 – 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	Стложенное обновление, запись без кэширования

Таблица 8–11 поясняет кодирование бит AP, определяющих разрешение на доступ для привилегированного и непривилегированного программного обеспечения (ПО).

Таблица 8–11 – Кодирование привилегий доступа в поле AP

AP[2:0]	Привелегирован-ный доступ	Непривелегирован-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
110	RO	RO	Только чтение и привилегированным, и непривилегированным ПО
111	RO	RO	Только чтение и привилегированным, и непривилегированным ПО

8.1.7 Несоответствие MP

Когда происходит нарушение разрешения доступа MPU, процессор генерирует ошибку управления памятью, см. раздел «Прерывания и исключения». Регистр MMFSR указывает причину ошибки. Для более подробной информации см. раздел «Поле MMFS».

8.1.8 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 разрешён. 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 DSR 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. раздел “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,
 ; совмещённые 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. раздел “MPU->RASR”. Младший значащий бит SRD контролирует первый подрегион, и старший значащий бит контролирует последний подрегион. Запрещение подрегиона означает, что другой регион перекрывает запрещённую область. Если другой разрешённый регион не перекрывает запрещённый 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 КВ и регион размером 512 КВ. Убедитесь, что атрибуты для региона один установлены для первых 128 КВ, установите SRD поле для региона два в значение b00000011 для запрещения первых двух подрегионов, как показано на рисунке ниже.

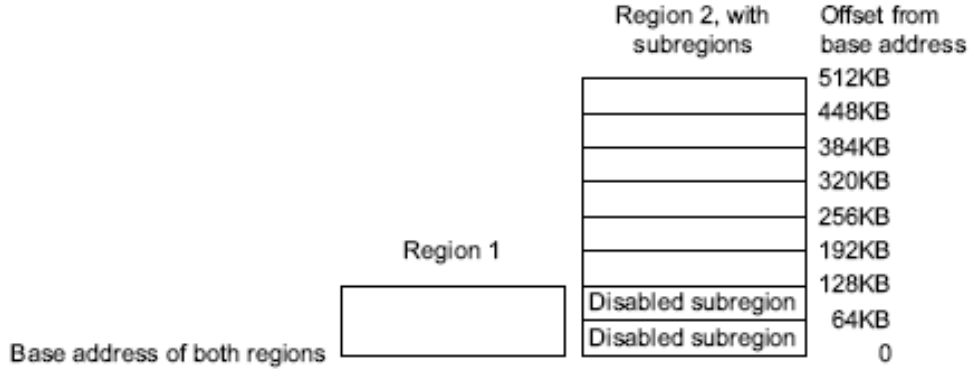


Рисунок 8–1 – Применение SRD

8.2 Советы и особенности применения MPU

Во избежание непредвиденных ситуаций, запретите прерывания перед обновлением атрибутов региона, к которому может осуществляется 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 микроконтроллера

Обычно, микроконтроллерные системы имеют только один процессор и не имеют кэша. В таких системах MPU программируется следующим образом:

Таблица 8–12 – Атрибуты регионов памяти для микроконтроллера

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 приложениях, 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.

9 Сигналы тактовой частоты

Микроконтроллер имеет 2 встроенных генератора, 2 внешних осциллятора и специализированный блок формирования тактовой синхронизации микроконтроллера.

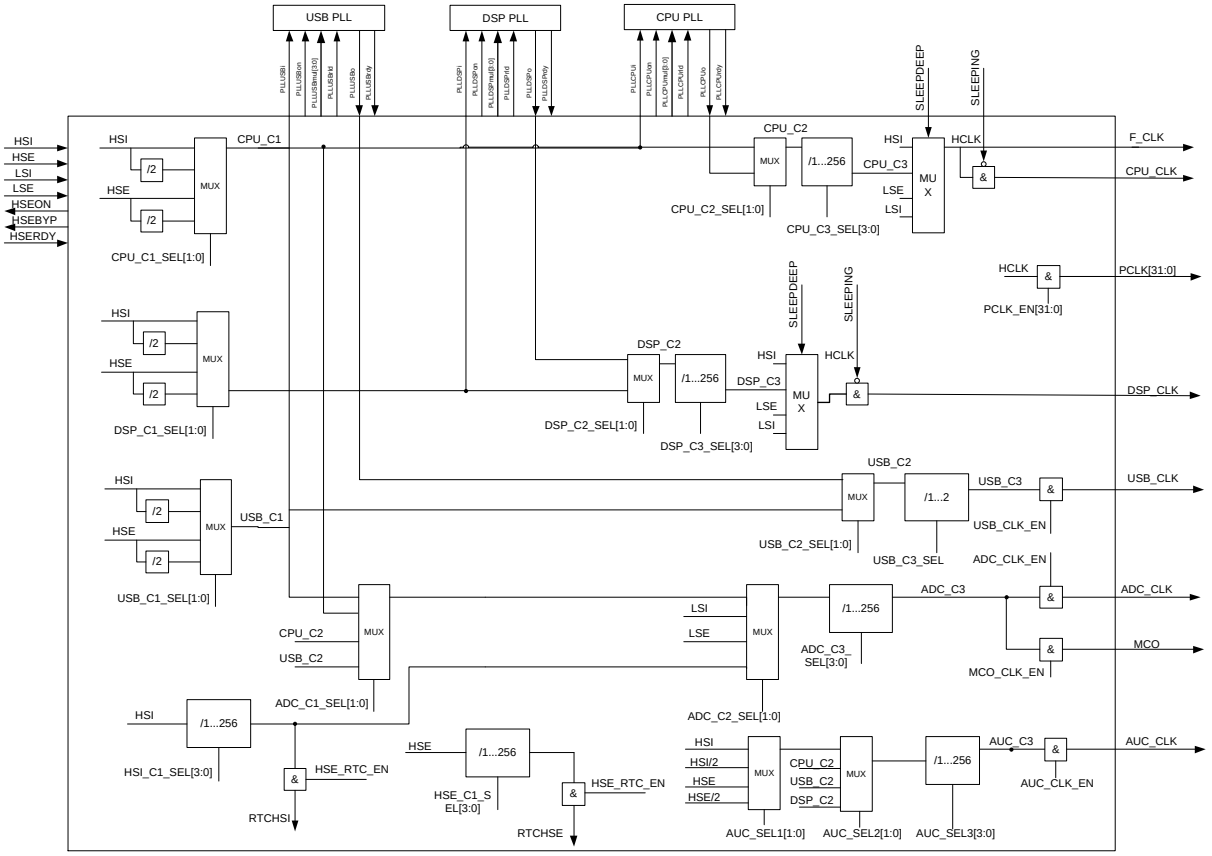


Рисунок 9–1 – Структурная блок-схема формирования тактовой частоты

Встроенный RC Генератор HSI

Генератор HSI вырабатывает тактовую частоту 8 МГц. Генератор автоматически запускается при появлении питания Ucc 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. Так же генератор может быть подстроен при помощи сигнала HSITRIM в регистре BKP_REG_0F.

Встроенный RC генератор LSI

Генератор LSI вырабатывает тактовую частоту 40 КГц. Генератор автоматически запускается при появлении питания Ucc 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 tpor. При дальнейшей работе генератор LSI может быть отключен при помощи сигнала LSION в регистре BKP_REG_0F.

Внешний осциллятор HSE

Осциллятор HSE предназначен для выработки тактовой частоты 2…16 МГц с помощью внешнего резонатора. Осциллятор запускается при появлении питания Ucc и сигнала разрешения HSEON в регистре HS_CONTROL. При выходе в нормальный режим работы вырабатывает сигнал HSERDY в регистре CLOCK_STATUS. Так же 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.

Внешний осциллятор LSE

Осциллятор LSE предназначен для выработки тактовой частоты 32 КГц с помощью внешнего резонатора. Осциллятор запускается при появлении питания BDUcc и сигнала разрешения LSEON в регистре BKP_REG_0F . При выходе в нормальный режим работы вырабатывает сигнал LSERDY в регистре BKP_REG_0F. Так же 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 BDUcc 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.

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. Входная частота блока умножителя должна быть в диапазоне 2…16 МГц выходная до 100 МГц. 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.

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. Входная частота блока умножителя должна быть в диапазоне 2…16 МГц выходная до 100 МГц. 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.

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. Входная частота блока умножителя должна быть в диапазоне 2…16 МГц выходная 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.

9.1 Описание регистров блока контроллера тактовой частоты

Управление тактовыми частотами ведется через периферийный блок RST_CLK. При включении питания микроконтроллер 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, CAN, USB, Таймеры) могут работать на частотах отличных от частоты процессорного ядра, по этому в соответствующих регистрах (UART_CLOCK, CAN_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.

Таблица 9–1 – Описание регистров блока контроллера тактовой частоты

Базовый Адрес	**Название**	**Описание**
0x4002_0000	MDR_RST_CLK	Контроллер тактовой частоты
Смещение		
0x00	CLOCK_STATUS	DR_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 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

9.1.1 DR_RST_CLK->CLOCK_STATUS

Таблица 9–2 – Регистр CLOCK_STATUS

Номер	31…3	2	1	0
Доступ	U	RO	RO	RO
Сброс	0	0	0	0
	-	**HSE RDY**	**PLL CPU RDY**	**PLL USB RDY**

Таблица 9–3 – Описание бит регистра CLOCK_STATUS

№ 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

9.1.2 MDR_RST_CLK->PLL_CONTROL

Таблица 9–4 – Регистр PLL_CONTROL

Номер	31...12	11…8	7…4	3	2	1	0
Доступ	U	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0	0
	-	**PLL**	**PLL**	**PLL**	**PLL**	**PLL**	**PLL**

		CPU MUL[3:0]	**USB MUL[3:0]**	**CPU PLD**	**CPU ON**	**USB RLD**	**USB ON**

Таблица 9–5 – Описание бит регистра PLL_CONTROL

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31…12	-	Зарезервировано
23...20	PLL DSP MUL[3:0]	Коэффициент умножения для DSP PLL: PLLDSPo = PLLDSPi x (PLLDSPMUL+1)
19…18	-	Зарезервировано
17	PLL DSP PLD	Бит перезапуска PLL При смене коэффициента умножения в рабочем режиме необходимо задать равным 1
16	PLL DSP ON	Бит включения PLL 0 – PLL выключена 1 – PLL включена
15...12	-	
11...8	PLL CPU MUL[3:0]	Коэффициент умножения для CPU PLL: PLLCPUo = PLLCPUi x (PLLCPUMUL+1)
7...4	PLL USB MUL[3:0]	Коэффициент умножения для USB PLL: PLLUSBo = PLLUSBi x (PLLUSBMUL+1)
3	PLL CPU PLD	Бит перезапуска PLL. При смене коэффициента умножения в рабочем режиме необходимо задать равным 1
2	PLL CPU ON	Бит включения PLL: 0 – PLL выключена; 1 – PLL включена
1	PLL USB RLD	Бит перезапуска PLL. При смене коэффициента умножения в рабочем режиме необходимо задать равным 1
0	PLL USB ON	Бит включения PLL: 0 – PLL выключена; 1 – PLL включена

9.1.3 MDR_RST_CLK->HS_CONTROL

Таблица 9–6 – Регистра HS_CONTROL

Номер	31…2	1	0
Доступ	U	R/W	R/W
Сброс	0	0	0
	-	**HSE BYP**	**HSE ON**

Таблица 9–7 – Описание бит регистра HS_CONTROL

№ 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

9.1.4 MDR_RST_CLK->CPU_CLOCK

Таблица 9–8 – 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 9–9 – Описание бит регистра CPU_CLOCK

№ 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 – HIS 01 – HSI/2 10 – HSE 11 – HSE/2

9.1.5 MDR_RST_CLK->USB_CLOCK

Таблица 9–10 – 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 9–11 – Описание бит регистра USB_CLOCK

№ 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
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

9.1.6 MDR_RST_CLK->ADC_MCO_CLOCK

Таблица 9–12 – 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 9–13 – Описание бит регистра ADC_MCO_CLOCK

№ 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
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

9.1.7 MDR_RST_CLK->RTC_CLOCK

Таблица 9–14 – 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 SEL[1:0]**	**HSE SEL[1:0]**

Таблица 9–15 – Описание бит регистра RTC_CLOCK

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31...10	-	Зарезервировано
9	HSI RTC EN	Бит разрешения HSI RTC: 0 – запрещен; 1 – разрешон
8	HSE RTC EN	Бит разрешения HSE RTC: 0 – запрещен; 1 – разрешон
7…4	HSI SEL[3:0]	Биты выбора делителя для HSI_C1: 0xxx – RTCHSI = HSI _C2 1000 - RTCHSI = HSI _C2 / 2 1001 - RTCHSI = HSI _C2 / 4 1010 - RTCHSI = HSI _C2 / 8 … 1111 - RTCHSI = HSI _C2 / 256
3…0	HSE SEL[3:0]	Биты выбора делителя для HSE_C1: 0xxx – RTCHSE = HSE _C2 1000 - RTCHSE = HSE _C2 / 2 1001 - RTCHSE = HSE _C2 / 4 1010 - RTCHSE = HSE _C2 / 8 … 1111 - RTCHSE = HSE _C2 / 256

9.1.8 MDR_RST_CLK->PER_CLOCK

Таблица 9–16 – Регистр PER_CLOCK

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

Таблица 9–17 – Описание бит регистра PER_CLOCK

№ 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] – SPI1 PCLK[9] – SDI0 PCLK[10] – I2C1 PCLK[11] – POWER PCLK[12] – WWDT PCLK[13] – IWDT PCLK[14] – TIMER1 PCLK[15] – TIMER2 PCLK[16] – TIMER3 PCLK[17] – ADC PCLK[18] – DAC PCLK[19] – COMP PCLK[20] – SPI2 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

9.1.9 MDR_RST_CLK->TIM_CLOCK

Таблица 9–18 – 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 9–19 – Описание бит регистра TIM_CLOCK

№ 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
23..16	TIM3 BRG [7:0]	Делитель тактовой частоты TIM3: xxxxx000 – TIM3_CLK == HCLK xxxxx001 – TIM3_CLK == HCLK/2 xxxxx010 – TIM3_CLK == HCLK/4 … xxxxx111 – TIM3_CLK == HCLK/128
15…8	TIM2 BRG [7:0]	Делитель тактовой частоты CAN2: xxxxx000 – TIM2_CLK == HCLK xxxxx001 – TIM2_CLK == HCLK/2 xxxxx010 – TIM2_CLK == HCLK/4 … xxxxx111 – TIM2_CLK == HCLK/128
7…0	TIM1 BRG [7:0]	Делитель тактовой частоты CAN1: xxxxx000 – TIM1_CLK == HCLK xxxxx001 – TIM1_CLK == HCLK/2 xxxxx010 – TIM1_CLK == HCLK/4 … xxxxx111 – TIM1_CLK == HCLK/128

9.1.10 MDR_RST_CLK->UART_CLOCK

Таблица 9–20 – Регистр UART_CLOCK

Номер	31…26	25	24	23…16	15…0	7…0
Доступ	U	R/W	R/W	U	R/W	R/W
Сброс	0	0	0	0	0	0
	-	**UART2 CLK EN**	**UART 1 CLK EN**	**-**	**UART 2 BRG [7:0]**	**UART 1 BRG [7:0]**

Таблица 9–21 – Описание бит регистра UART_CLOCK

№ 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.. 19	-	Зарезервировано
18.. 16	UART3 BRG [7:0]	Делитель тактовой частоты UART 3: xxxxx000 – UART 3_CLK == HCLK 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 CAN1: xxxxx000 – UART 1_CLK == HCLK xxxxx001 – UART 1_CLK == HCLK/2 xxxxx010 – UART 1_CLK == HCLK/4 … xxxxx111 – UART 1_CLK == HCLK/128

9.1.11 MDR_RST_CLK->SSP_CLOCK

Таблица 9–22 – 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 9–23 – Описание бит регистра SSP_CLOCK

№ 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 BRG [7:0]	Делитель тактовой частоты SSP 2: 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 CAN1: xxxxx000 – SSP 1_CLK == HCLK xxxxx001 – SSP 1_CLK == HCLK/2 xxxxx010 – SSP 1_CLK == HCLK/4 … xxxxx111 – SSP 1_CLK == HCLK/128

9.1.12 MDR_RST_CLK->DSP_CLOCK

Таблица 9–24 – 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 9–25 – Описание бит регистра DSP_CLOCK

№ 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

9.1.13 MDR_RST_CLK->SSP_CLOCK2

Таблица 9–26 – 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 9–27 – Описание бит регистра SSP_CLOCK2

№ 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 xxxxx000 – SSP 4_CLK == HCLK xxxxx001 – SSP 4_CLK == HCLK/2 xxxxx010 – SSP 4_CLK == HCLK/4 … xxxxx111 – SSP 4_CLK == HCLK/128

10 Батарейный домен и часы реального времени MDR_BKP

Блок батарейного домена предназначен для обеспечения функций часов реального времени и сохранения некоторого набора пользовательских данных при отключении основного источника питания. При снижении питания U_{CC} в блоке SW происходит автоматическое переключение питания BDU_{CC} с U_{CC} на BU_{CC}. Если на BU_{CC} имеется отдельный источник питания (батарейка), то батарейный домен остается включенным и может выполнять свои функции.

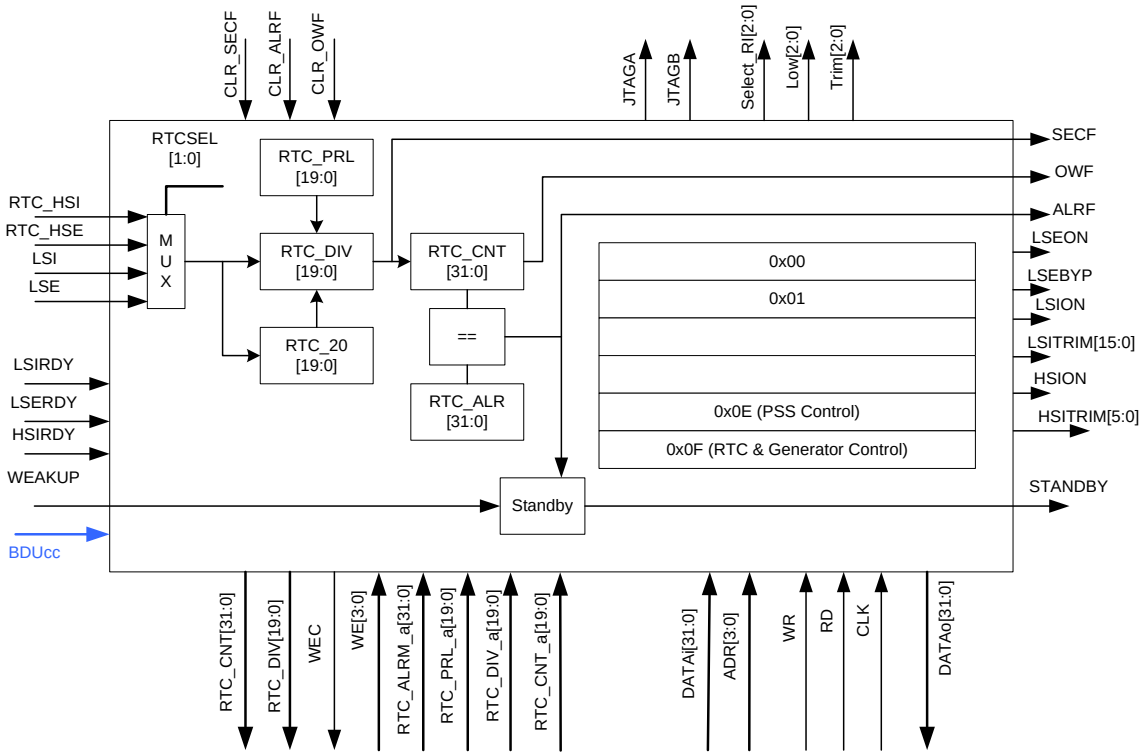


Рисунок 10–1 – Структурная блок-схема батарейного домена и часов реального времени

10.1 Часы реального времени

Часы реального времени позволяют организовать механизм отсчета времени в кристалле, в том числе при отключении основного источника питания. Включение часов реального времени осуществляется битом RTCEN. В качестве источника тактовой частоты часов реального времени может выступать генератор LSI, или осциллятор LSE, или HSE, или HSI с дополнительным делителем до 256 (HSE и HSI формируются в блоке управления тактовыми частотами и могут быть выбраны только при наличии питания DU_{CC}, LSI может быть выбран при наличии питания U_{CC}, LSE может быть выбран при наличии U_{CC} или BU_{CC}). Выбор между источниками осуществляется битами RTCSEL. При возможном отключении основного источника питания U_{CC} в качестве источника тактовой частоты должен использоваться осциллятор LSE, так как он также имеет питание BDU_{CC}. Биты управления осциллятором LSE расположены в батарейном домене и таким образом при отключении основного питания они не сбрасываются.

Для калибровки тактовой частоты используются биты CAL[6:0]. Значение CAL определяет, какое число тактов из 2^{20} будет замаскировано. Таким образом, с помощью бит CAL[6:0] производится замедление хода часов. Изменение значения бит CAL может быть осуществлено в ходе работы часов реального времени.

Регистр RTC_DIV выступает в роли 20-ти битного предварительного делителя входной тактовой частоты, таким образом, чтобы на его выходе была тактовая частота в 1 Гц. Для задания коэффициента деления регистра RTC_DIV используется регистр RTC_PRL.

Регистр RTC_CNT предназначен для отсчета времени в секундах и работает на выходной частоте делителя RTC_DIV. Регистр RTC_ALR предназначен для задания времени, при совпадении с которым вырабатывается флаг прерывания и пробуждения процессора. Таким образом, бит STANDBY, отключающий внутренний регулятор напряжения, автоматически сбрасывается при совпадении RTC_CNT и RTC_ALR.

Бит STANDBY также может быть сброшен с помощью вывода WAKEUP.

10.2 Регистры аварийного сохранения

Батарейный домен имеет 16 встроенных 32-х 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.

10.3 Описание регистров блока батарейного домена

Таблица 10–1 – Описание регистров блока батарейного домена

Базовый Адрес	Название	Описание
0x400D_8000	MDR_BKP	Контроллер батарейного домена и часов реального времени
Смещение		
0x00	REG_00	Регистр MDR_BKP-> 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 предварительного делителя основного счетчика
0x48	RTC_PRL	Регистр MDR_BKP->RTC_PRL основания счета предварительного делителя
0x4C	RTC_ALRM	Регистр MDR_BKP->RTC_ALRM значения для сравнения основного счетчика и выработки сигнала ALRF
0x50	RTC_CS	Регистр

		MDR_BKP->RTC_CS управления и состояния флагов часов реального времени

10.3.1 MDR_BKP-> REG_[0D…00]

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 10–2 – Регистры REG_[0D…00]

Номер	31…0
Доступ	R/W
Сброс	U
	BKP REG[31:0]

Таблица 10–3 – Описание бит регистров REG_[0D…00]

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31…0	BKP REG[31:0]	Регистр аварийного сохранения

10.3.2 MDR_BKP->REG_0E

Таблица 10–4 – 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	U	U	0	0
	-	**MODE [2:0]**	**FPOR**	**Trim [2:0]**	**JTAG_B**	**JTAG_A**	**SelectRI [2:0]**	**LOW [2:0]**

Таблица 10–5 – Описание бит регистра REG_0E

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31…15	-	Зарезервировано
14...12	MODE[2..0]	Режим работы микроконтроллера, 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 без отладки 100 – зарезервировано 101 – UART загрузчик 110 – UART загрузчик 111 – зарезервировано (режим тестирования кристалла) Подробнее в разделе Загрузочное ПЗУ и режимы работы микроконтроллера. 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 DU_{CC}. С помощью Trim осуществляется подстройка напряжения DU_{CC}: 000 – DU_{CC} + 0,10В – значение по умолчанию. 001 – DU_{CC} + 0,06В 010 – DU_{CC} + 0,04В 011 – DU_{CC} + 0,01В 100 – DU_{CC} – 0,01В 101 – DU_{CC} – 0,04В 110 – DU_{CC} – 0,06В 111 – DU_{CC} – 0,10В
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 она не функционирует как микроконтроллер
6	JTAG A	Разрешение работы порта JTAG A: 0 – запрещен; 1 – разрешон
5…3	SelectRI[2:0]	Выбор дополнительной стабилизирующей нагрузки для встроенного регулятора напряжения DU_{CC}: 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 DU_{CC}. Значение LOW должно совпадать со значением SelectRI и выставляться в зависимости от тактовой частоты микроконтроллера: 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

10.3.3 MDR_BKP->REG_0F

Таблица 10–6 – 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**	**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	R/W	R/W	RO	R/W
Сброс	0	0	0	1	1	1	0
	RTC RESET	**STANDBY**	**HSI TRIM [5:0]**	**HSI RDY**	**HSI ON**	**LSI RDY**	**LSI TRIM [4:0]**

Таблица 10–7 – Описание бит регистра REG_0F

№ 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
30	STANDBY	Режим отключения регулятора DU_{CC}: 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 HIS. Смотрите диаграмму зависимости (Рисунок 10–3)
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 10–2)
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	CAL[7:0]	Коэффициент подстройки тактовой частоты часов реального времени, из каждых 2^{20} тактов будет замаскировано CAL тактов: 00000000 – 0 тактов 00000001 – 1 такт …. 11111111 – 256 тактов Таким образом, при частоте 32768.00000 Гц при CAL = 0 тактов частота = 32768.00000 Гц при CAL = 1 такт частота = 32767,96875 Гц; … при CAL = 256 тактов частота = 32760.00000 Гц
4	RTC EN	Бит разрешения работы часов реального времени: 0 – работа запрещена; 1 – работа разрешена
3…2	RTC SEL[1:0]	Биты выбора источника тактовой синхронизации часов реального времени: 00 – LSI 01 – LSE 10 – HSIRTC (формируется в блоке CLKRST) 11 – HSERTC (формируется в блоке CLKRST)
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

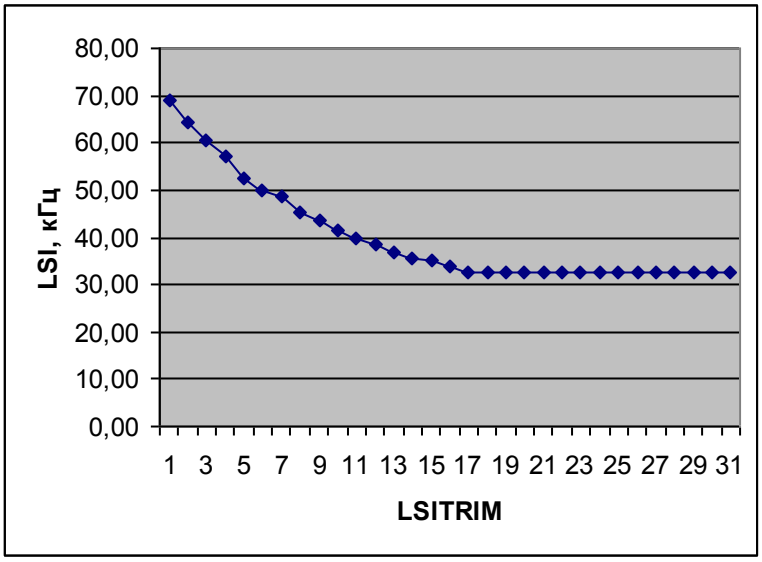


Рисунок 10–2 – Зависимость частоты LSI от значения LSITRIM

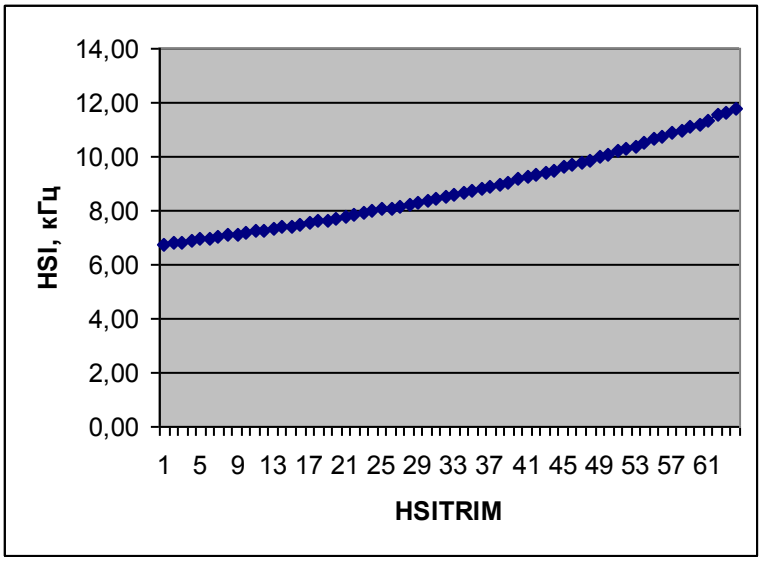


Рисунок 10–3 – Зависимость частоты HSI от значения HSITRIM

10.3.4 MDR_BKP->RTC_CNT

Таблица 10–8 – Регистр RTC_CNT

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

Таблица 10–9 – Описание бит регистра RTC_CNT

№ 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

10.3.5 MDR_BKP->RTC_DIV

Таблица 10–10 – Регистр RTC_DIV

Номер	31…20	19…0
Доступ	U	R/W
Сброс	0	0
	-	**RTC DIV[19:0]**

Таблица 10–11 – Описание бит регистра RTC_DIV

№ 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

10.3.6 MDR_BKP->RTC_PRL

Таблица 10–12 – Регистр RTC_PRL

Номер	31...20	19…0
Доступ	U	R/W
Сброс	0	0
	-	**RTC PRL[19:0]**

Таблица 10–13 – Описание бит регистра RTC_PRL

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31…20	-	Зарезервировано
19...0	RTC PRL[19:0]	Значение основания для счета счетчика предварительного делителя часов реального времени

10.3.7 MDR_BKP->RTC_ALRM

Таблица 10–14 – Регистр RTC_ALRM

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

Таблица 10–15 – Описание бит регистра RTC_ALRM

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31...0	RTC ALRM[31:0]	Значения для сравнения основного счетчика и выработки сигнала ALRF

10.3.8 MDR_BKP->RTC_CS

Таблица 10–16 – Регистр RTC_CS

Номер	31…7	6	5	4	3	2	1	0
Доступ	U	R/W	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 10–17 – Описание бит регистра RTC_PRL

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
30…7	-	Зарезервировано
6	**WEC**	Запись завершена: 0 – можно записывать в регистры RTC; 1 – запись в регистры запрещена, идет запись в регистры RTC
5	**ALRF_IE**	Флаг разрешения прерывания по совпадению основного счетчики и регистра RTC_ALRM: 0 – нет совпадения; 1 – есть совпадение
4	**SECF_IE**	Флаг разрешения прерывания по разрешению счета основного счетчика от счетчика предварительного деления: 0 – нет разрешения счета; 1 – разрешение счета
3	**OWF_IE**	Флаг разрешения прерывания по переполнению основного счетчика RTC_CNT: 0 – нет переполнения 1 – было переполнение
2	**ALRF**	Флаг совпадения основного счетчики и регистра RTC_ALRM: 0 – нет совпадения; 1 – есть совпадение
1	**SECF**	Флаг разрешения счета основного счетчика от счетчика предварительного деления: 0 – нет разрешения счета; 1 – разрешение счета
0	**OWF**	Флаг переполнения основного счетчика RTC_CNT: 0 – нет переполнения; 1 – было переполнение

11 Порты ввода-вывода

Микроконтроллер имеет 6 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.

Таблица 11–1 – 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	
Порт C										
PC0	-		PC0		COMPOUT	1	SSP4_FSS	17	SDIO_CLK	2
PC1	-		PC1		OE		UART1_TXD	10	SDIO_CMD	
PC2	-		PC2		WE		UART1_RXD		SDIO_DATA0	
PC3	-		PC3		BE0		SSP4_RXD	17	SDIO_DATA1	
PC4	-		PC4		BE1		SSP4_TXD		SDIO_DATA2	
PC5	-		PC5		BE2		SSP4_CLK		SDIO_DATA3	
PC6	-		PC6		BE3		TMR1_CH3	3	UART2_TXD	14
PC7	-		PC7		CLOCK		TMR1_CH3n		UART2_RXD	
PC8	-		PC8		SDIO_CLK	2	SSP3_TXD	4	BSP2_TX	19
PC9	-		PC9		SDIO_CMD		SSP3_FSS		BSP2_RTFR	

PC10	-		PC10	SDIO_DATA0		SSP3_RXD		BSP2_RX	
PC11	-		PC11	SDIO_DATA1		SSP3_CLK		BSP2_RTCLK	
PC12	-		PC12	SDIO_DATA2		SSP4_FSS	17	BSP3_RTFR	20
PC13	-		PC13	SDIO_DATA3		SSP4_CLK		BSP3_RTCLK	
PC14	-		PC14	I2C1_SCK	11	SSP4_TXD		BSP3_TX	
PC15	-		PC15	I2C_SDA		SSP4_RXD		BSP3_RX	
Порт D									
PD0	ADC0_REF+	5	PD0 JB_TMS	TMR1_BLK	3	I2C1_SCK	11	BSP3_RX	20
PD1	ADC1_REF-		PD1 JB_TCK	TMR1_ETR		I2C1_SDA		BSP3_TX	
PD2	ADC2		PD2 JB_TRST	BUSY1	1	SSP1_TXD	16	BSP2_TX	19
PD3	ADC3		PD3 JB_TDI	TMR1_CH1	3	SSP1_FSS		BSP2_TFR	
PD4	ADC4		PD4 JB_TDO	TMR1_CH1n		SSP1_RXD		BSP2_RX	
PD5	ADC5		PD5	CLE	1	SSP1_CLK		BSP2_TCKL	
PD6	ADC6		PD6	ALE		TMR1_CH4	3	BSP3_TCLK	20
PD7	ADC7		PD7	TMR1_CH2	3	UART3_TXD	18	BSP3_TFR	
PD8	ADC8		PD8	TMR1_CH2n		UART3_RXD		BSP3_TX	
PD9	ADC9		PD9	TMR1_CH3		TMR1_CH4n	3	BSP3_RFR	
PD10	ADC10		PD10	TMR1_CH3n		TMR2_BLK	15	BSP3_RX	
PD11	ADC11		PD11	TMR1_CH4		TMR2_ETR		BSP3_RCLK	
PD12	ADC12		PD12	TMR1_CH4n		SSP2_FSS	13	BSP2_RFR	19
PD13	ADC13		PD13	UART3_TXD	18	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	DAC1_OUT	6	PE0	ADDR16	1	SSP4_FSS	17	BSP1_RTFR	18
PE1	DAC1_REF		PE1	ADDR17		SSP4_CLK		BSP1_RTCLK	
PE2	COMP_IN1	7	PE2	ADDR18		SSP4_TXD		BSP1_TX	
PE3	COMP_IN2		PE3	ADDR19		SSP4_RXD		BSP1_RX	
PE4	COMP_REF+		PE4	ADDR20		SSP2_TXD	13	UART1_TXD	10
PE5	COMP_REF-		PE5	ADDR21		SSP2_FSS		UART1_RXD	
PE6	OSC_IN32	8	PE6	ADDR22		SSP2_RXD		EXT_INT3	9
PE7	OSC_OUT32		PE7	ADDR23		SSP2_CLK		TMR2_CH4	15
PE8	COMP_IN3	7	PE8	ADDR24		TMR2_CH4	15	TMR2_CH4n	
PE9	DAC2_OUT	6	PE9	ADDR25		TMR2_CH4n		TMR3_CH1	12
PE10	DAC2_REF		PE10	ADDR26		UART2_TXD	14	TMR3_ETR	
PE11	-		PE11	ADDR27		UART2_RXD		TMR3_BLK	
PE12	-		PE12	ADDR28		SSP1_RXD	16	BSP2_RTFR	19
PE13	-		PE13	ADDR29		SSP1_TXD		BSP2_RTCLK	
PE14	-		PE14	ADDR30		SSP1_CLK		BSP2_TX	
PE15	-		PE15	ADDR31		SSP1_FSS		BSP2_RX	
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 MODE[0]	ADDR4		SSP4_CLK		BSP3_TCKL	
PF5	-		PF5 MODE[1]	ADDR5		SSP4_FSS		BSP3_TFR	
PF6	-		PF6 MODE[2]	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	20
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.
18) Выводы управляются контроллером интерфейса McBSP2.
18) Выводы управляются контроллером интерфейса McBSP3.

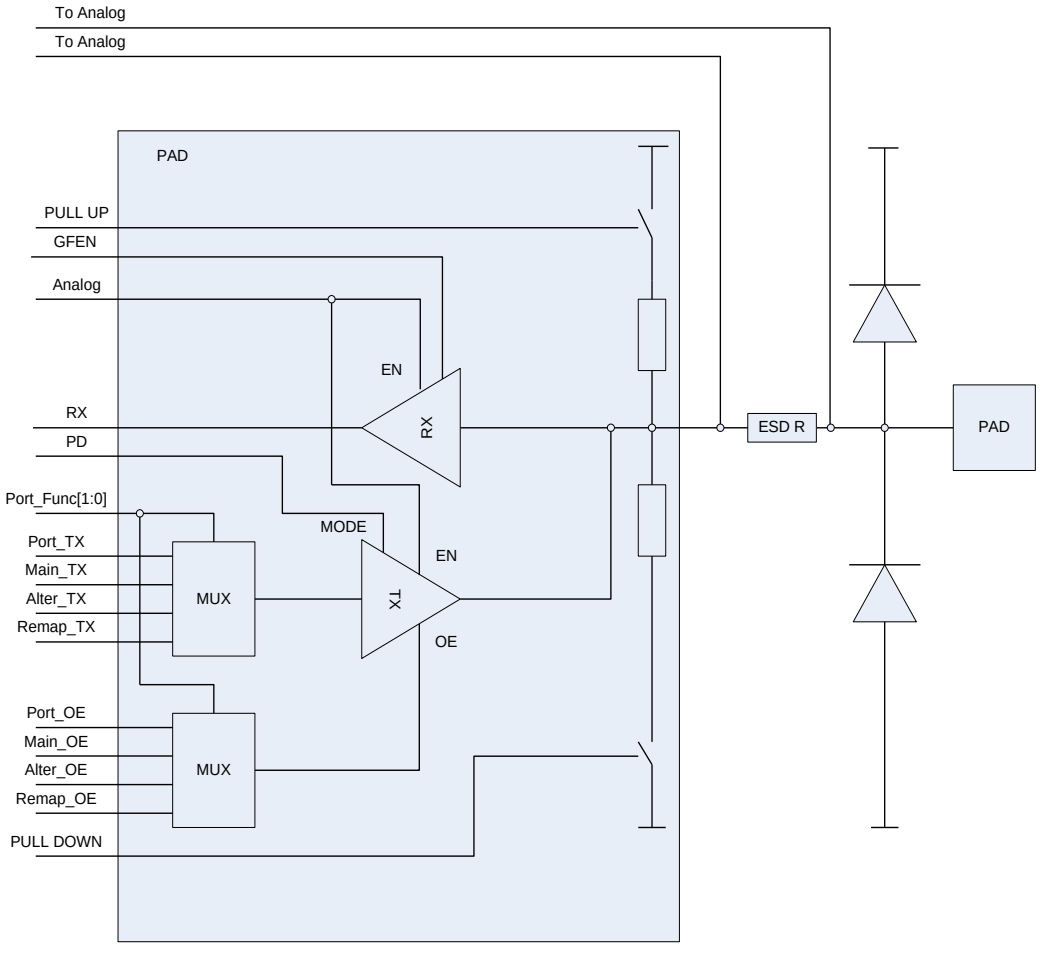


Рисунок 11–1 – Структурная схема порта ввода/вывода

11.1 Описание регистров портов ввода-вывода

Таблица 11–2 – Описание регистров портов ввода-вывода

Базовый	Название	Описание

Адрес		
0x400A_8000	MDR_PORTA	Порт A
0x400B_0000	MDR_PORTB	Порт B
0x400B_8000	MDR_PORTC	Порт C
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
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

11.1.1 MDR_PORTx->RXTX

Таблица 11–3 – Регистр RXTX

Номер	31...16	15...0
Доступ	U	R/W
Сброс	0	0
	-	**PORT RXTX[15:0]**

Таблица 11–4 – Описание бит регистра RXTX

№ 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 выводы порта и для чтения

11.1.2 MDR_PORTx->OE

Таблица 11–5 – Регистр OE

Номер	31...16	15...0
Доступ	U	R/W
Сброс	0	0
	-	**PORT OE[15:0]**

Таблица 11–6 – Описание бит регистра OE

№ 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

11.1.3 MDR_PORTx->FUNC

Таблица 11–7 – 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 11–8 – Описание бит регистра FUNC

№ 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

11.1.4 MDR_PORTx->ANALOG

Таблица 11–9 – Регистр ANALOG

Номер	31…16	15...0
Доступ	U	R/W
Сброс	0	0
	-	**ANALOG EN[15:0]**

Таблица 11–10 – Описание бит регистра ANALOG

№ 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

11.1.5 MDR_PORTx->PULL

Таблица 11–11 – Регистр PULL

Номер	31…16	15…0
Доступ	R/W	R/W
Сброс	0	0
	PULL UP[15:0]	**PULL DOWN[15:0]**

Таблица 11–12 – Описание бит регистра PULL

№ 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)

11.1.6 MDR_PORTx->PD

Таблица 11–13 – Регистр PD

Номер	31...16	15...0
Доступ	R/W	R/W
Сброс	0	0
	PORT SHM[15:0]	**PORT PD[15:0]**

Таблица 11–14 – Описание бит регистра PD

№ 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	Режим работы контроллера.

	PD[15:0]	Режим работы выхода: 0 – управляемый драйвер; 1 – открытый сток

11.1.7 MDR_PORTx->PWR

Таблица 11–15 – 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 11–16 – Описание бит регистра PORTx_PWR

№ 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

11.1.8 MDR_PORTx->GFEN

Таблица 11–17 – Регистр GFEN

Номер	31...16	15...0
Доступ	R/W	R/W
Сброс	0	0
	-	**GFEN[15:0]**

Таблица 11–18 – Описание бит регистра GFEN

№ 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

12 Детектор напряжения питания MDR_POWER

Блок детектора напряжения питания PVD предназначен для контроля питания U_{CC} и BU_{CC} при работе микроконтроллера. 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 BU_{CC} 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 микроконтроллерах второй ревизии (по адресу 0x000003FC загрузочного ПЗУ хранится значение 0x83400FDF и год выпуска не ранее 2011) при $BU_{CC}<U_{CC}$ блок определения уровня BU_{CC} не функционирует.

Таблица 12–1 – Типовые уровни напряжений детектора питания

Параметр	Не менее	Типовое	Не более
Входное напряжение, U_{CC}, В	2,0	-	3,6
Входное напряжение, BU_{CC}, 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 BU_{CC}, при PBLS = “00”, В		1,8	
Уровень срабатывания PBVD от BU_{CC}, при PBLS = “01”, В		2,2	
Уровень срабатывания PBVD от BU_{CC}, при PBLS = “10”, В		2,6	
Уровень срабатывания PBVD от BU_{CC}, при PBLS = “11”, В		3,0	

Таблица 12–2 – 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

12.1 MDR_POWER->PVDCS

Таблица 12–3 – 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]**	**PBLS [1:0]**	**PVD EN**

Номер	31…12	11	10
Доступ	U	R/W	R/W
Сброс	0	0	0
	-	**INV**	**INVB**

Таблица 12–4 – Описание бит регистра PVDCS

№ 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
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
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

13 Внешняя системная шина MDR_EBC

Внешняя системная шина позволяет работать с внешними микросхемами памяти и периферийными устройствами. Области адресного пространства микроконтроллера отведены для работы с внешней системной шиной.

Таблица 13–1 – 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 выводы портов для формирования сигналов CE.

13.1 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 микроконтроллера 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 выводы внешней системной шины ADDR, DATA и сигналы управления OE, WE, BE[3:0] и сигнал синхронизации CLOCK.

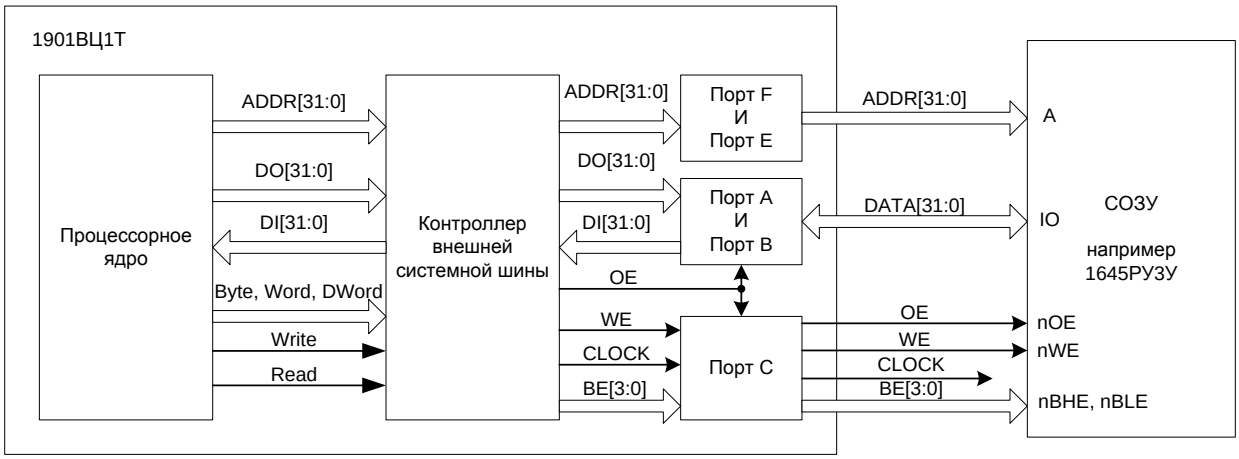


Рисунок 13–1 – Обмен по внешней системной шине

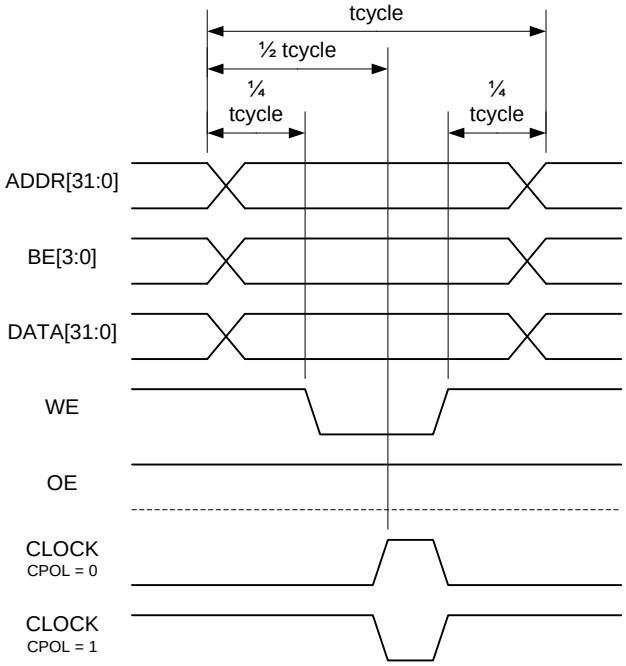


Рисунок 13–2 – Диаграмма записи

Время цикла записи tcycle задается битами WAIT_STATE[3:0]. Активный уровень сигналов WE, OE, BE[3:0] низкий. Если сигнал CLOCK не требуется, он может не использоваться.

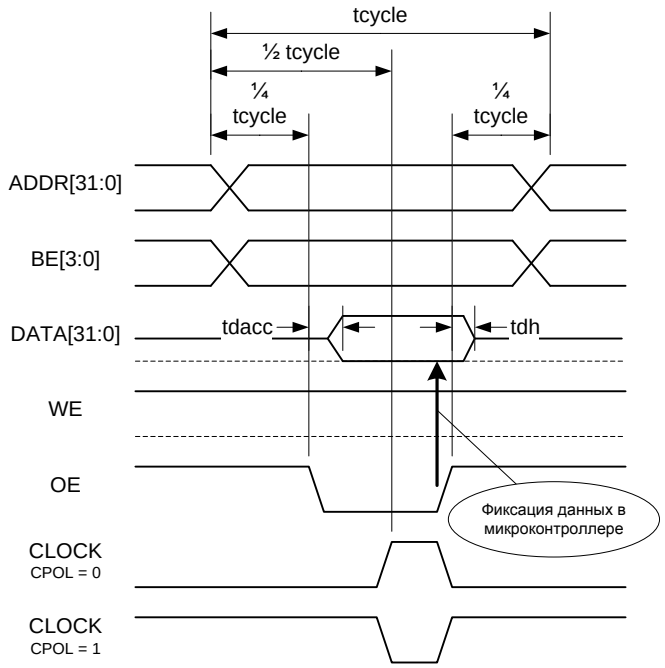


Рисунок 13–3 – Диаграмма чтения

При чтении по внешней системной шине необходимо выбрать такую длительность времени tcycle, чтобы выполнялось время скорости доступа к памяти. Время tdh для микроконтроллера равно нулю.

Таблица 13–2 – 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

13.2 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 микроконтроллера 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.

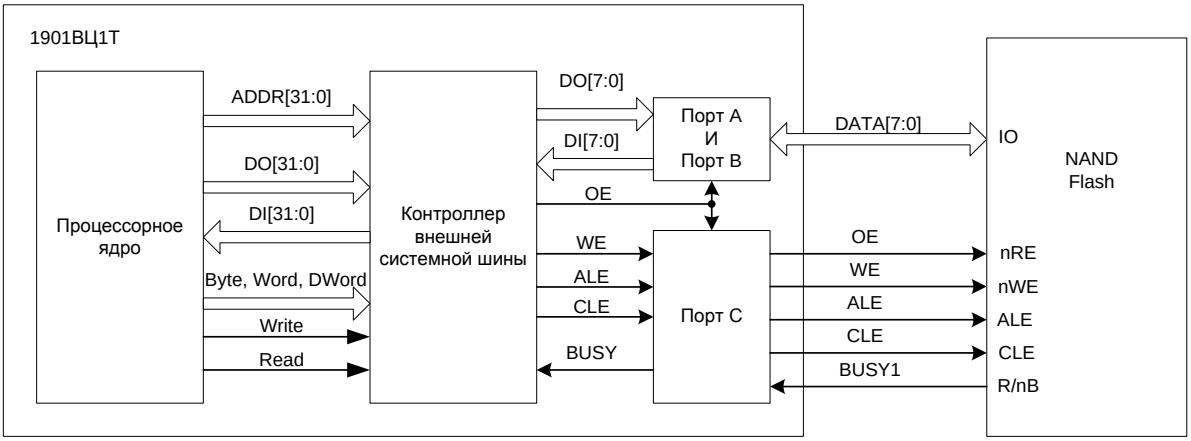


Рисунок 13–4 – Подключение внешней NAND Flash

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 13–3).

Таблица 13–3 – 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 0x90 – Read ID 0xFF – Reset 0x80 – Page Program 0x60 – Block Erase 0x70 – Read Status	Не имеет значения
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_wp = 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
```

13.3 Описание регистров блока контроллера внешней системной шины

Таблица 13–4 – 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

13.3.1 MDR_EBC->CONTROL

Таблица 13–5 – 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
Срос	0	0	0	0	0	0	1	0	0	0	0	0
	-	**WAIT_STATE[3:0]**				**-**	**BUSY**	**-**	**CPOL**	**NAND**	**RAM**	**ROM**

Таблица 13–6 – Описание бит регистра CONTROL

№ 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 AHB, необходимых для стандартного цикла записи/чтения. Сигналы OE/WE устанавливаются в момент времени ¼ WAIT_STATE, снимаются в момент времени ¾ 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

13.3.2 MDR_EBC->NAND_CYCLES

Таблица 13–7 – 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
Срос		0	0	0	0	0	0	0
	-	**t_rr**	**t_alea**	**t_whr**	**t_wp**	**t_rea**	**t_wc**	**t_rc**

Таблица 13–8 – Описание бит регистра NAND_CYCLES

№ 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

14 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 14–1). Контроллер DMA содержит 32 аппаратных/программных запроса. Аппаратные запросы реализованы от большинства периферийных устройств RISC и DSP подсистем (таблица 21–3). Каждый канал может быть сконфигурирован для обработки аппаратных или программных запросов.

Несмотря на то, что DSP ядро не имеет доступа к регистрам DMA RISC, в системе реализован специальный регистр, предоставляющий возможность задавать запросы при помощи DSP ядра. Подробнее об этом см. раздел «Организация управления DSP подсистемой».

Таблица 14–1 – 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).

14.1 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.

14.2 Термины и определения

При описании контроллера используются следующие термины:

Таблица 14–2 – 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 chnl_pri_alt_set 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 100
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 и 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 dma_done, если окончание подобного режима работы происходит после выполнения обычного цикла

14.3 Функциональное описание

Ниже (Рисунок 14–1) показана упрощенная структурная схема контроллера.

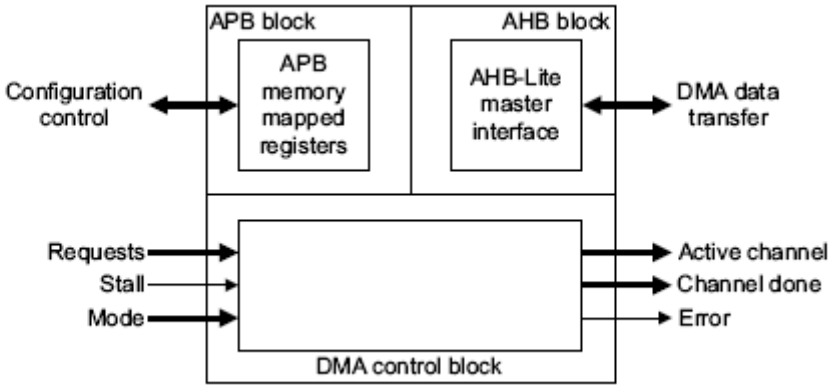


Рисунок 14–1 – Структурная схема контроллера

Контроллер состоит из следующих основных функциональных блоков:
- блок, подключенный к шине APB;
- блок, подключенный к шине AHB;
- управляющий блок DMA.

14.3.1 Распределение каналов DMA

Таблица 14–3 – Распределение каналов DMA

Номер канала	Источник req	Источник sreg	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
22	A_ADC(DSP)	-	Запрос от АЦП Аудиокодека(DSP)

23	A_DAC(DSP)	-	Запрос от ЦАП Аудиокодека(DSP)
24	Crypto(DSP)	-	Запрос от Крипто-модуля(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

14.3.2 Блок, подключенный к шине APB

Блок содержит набор регистров, позволяющих настраивать контроллер, используя ведомый APB интерфейс. Регистры занимают адресное пространство емкостью 4 кбайт.

14.3.3 Блок, подключенный к шине AHB

Контроллер содержит один блок типа «ведущий» шины DMA Bus, который позволяет, используя 32-х разрядную шину, передавать данные от источника к приемнику. Источник и приемник являются ведомыми шины AHB.

14.3.4 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.

14.3.5 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 AHB, где блоки должны «захватывать» шину или обращаться к внешней памяти. В тоже время контроллер DMA предназначен для использования в многоуровневых системах с шиной AHB, включающих встроенную память.

14.3.6 Разрядность передач данных

Контроллер интерфейса предоставляет возможность осуществлять передачу 8, 16 и 32 разрядных данных. Таблица перечисляет значения комбинаций шины HSIZE.

Таблица 14–4 – 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 логический ноль.
**) – запрещенная комбинация

Контроллер всегда использует передачи 32-х разрядными данными при обращении к управляющим данным канала. Необходимо устанавливать разрядность данных источника, соответствующую разрядности данных приемника.

14.3.7 Управление защитой данных

Контроллер позволяет устанавливать режимы защиты данных протокола AHB-Lite, определяемые шиной HPROT[3:1]. Возможен выбор следующих режимов защиты:

- кэширование;
- буферизация;
- привилегированный.

Таблица 14–5 перечисляет значения комбинаций шины HPROT.

Таблица 14–5 – Режимы защиты данных

HPROT[3] Кэширование	HPROT[2] буферизация	HPROT[1] Привилегированный	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.

14.3.8 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 14–6 – 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 «

Структура управляющих данных канала»).

14.4 Управление DMA

14.4.1 Правила обмена данными

Контроллер использует правила обмена данными, перечисленные далее в Таблица 14–7, 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];
- контроллер находится не в тестовом режиме, что выполняется установкой в состояние логического нуля разряда управления int_test_en bit[C].

Таблица 14–7 – Правила, при которых передача данных по каналам разрешена, и запросы не маскируются

Правило	Описание
1	Если dma_active[C] установлен в 0, то установка в 1 dma_req[C] или dma_sreq[C] на один или более тактов сигнала hclk, следующих или не следующих друг за другом, инициирует передачу по каналу номер C
2	Контроллер осуществляет установку в 1 только одного разряда dma_active[C]
3	Контроллер устанавливает в 1 dma_active[C] в момент начала передачи по каналу C
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 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 контролер осуществляет DMA передачи согласно правилам, представленным в Таблица 14–8.

Таблица 14–8 – 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

14.4.2 Диаграммы работы контроллера DMA

Данный раздел описывает примеры функционирования контроллера с использованием правил обмена данными, представленных ранее (Таблица 14–7):

- 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 (Рисунок 14–2 – Рисунок 14–6), подразумевают следующее:

- hready находится в состоянии 1;
- AHB «ведомый» всегда дает ответ «OKAY».

14.4.2.1 Импульсный запрос на обработку

Рисунок 14–2 показывает временную диаграмму работы контроллера DMA при получении импульсного запроса от периферии.

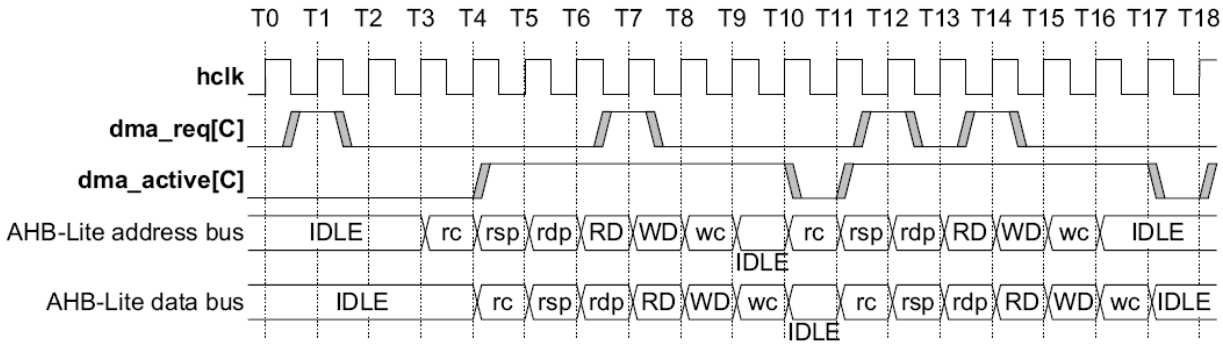


Рисунок 14–2 – Диаграмма работы при получении импульсного запроса

Пояснения к диаграмме (Рисунок 14–2) приведены ниже.

Таблица 14–9 – Пояснения к диаграмме работы при получении импульсного запроса

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)
T4	Контроллер устанавливает dma_active[C] (см. правила 2 и 3) и начинает DMA передачи по каналу C

T4-T7	Контроллер считывает управляющую 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	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
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)
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)

14.4.2.2 Запрос на обработку по уровню

Рисунок 14–3 показывает временную диаграмму работы контроллера DMA при получении от периферии запроса на обработку по уровню.

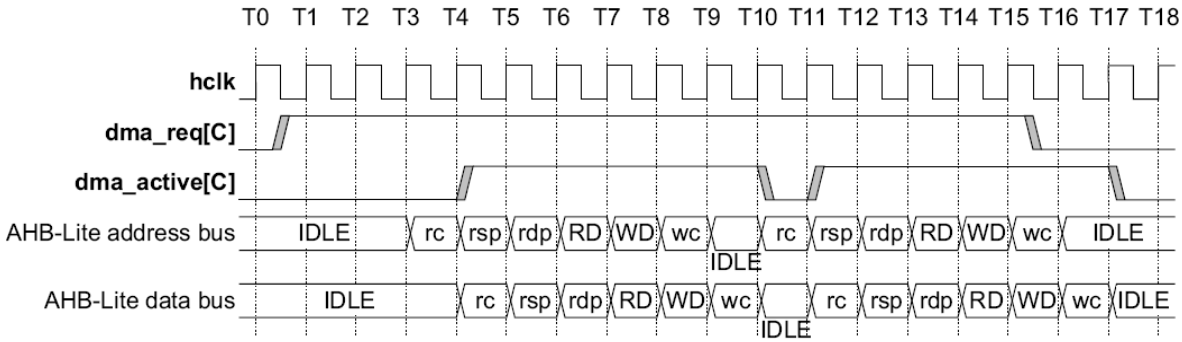


Рисунок 14–3 – Диаграмма работы при получении запроса на обработку по уровню

Пояснения к диаграмме (Рисунок 14–3) представлены ниже (Таблица 14–10).

Таблица 14–10 – Пояснения к диаграмме работы при получении запроса на обработку по уровню

T1	Контроллер обнаружил запрос на обработку по каналу C (Таблица 14–7, 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.

14.4.2.3 Флаги завершения

Рисунок 14–4 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.

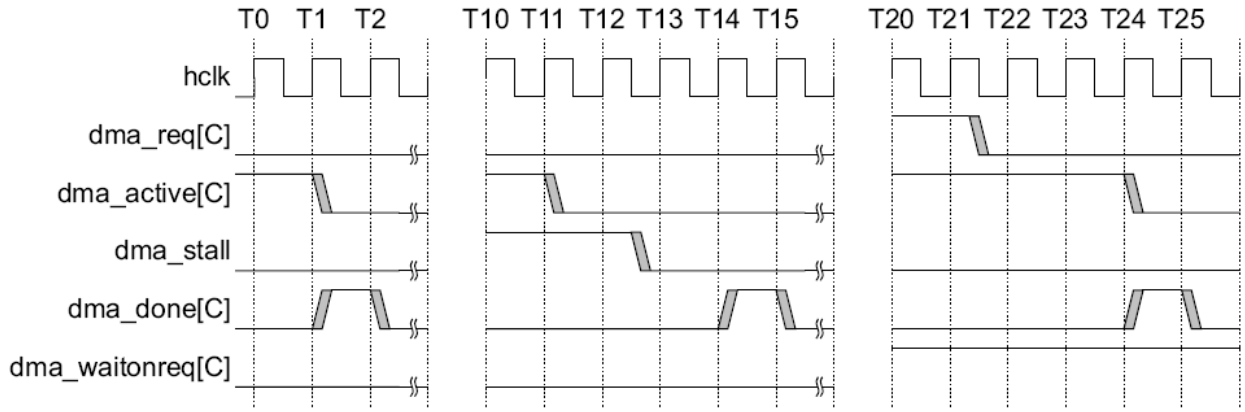


Рисунок 14–4 – Диаграммы функционирования dma_done

Пояснения к диаграмме (Рисунок 14–4), такты от T0 до T2, приведены в таблице ниже.

Таблица 14–11 – Пояснения функционирования dma_done, такты от T0 до T2

T1	Контроллер сбрасывает сигнал dma_active[C], что указывает на окончание передачи DMA (см. Таблица 14–7, 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 14–4), такты от T10 до T15, приведены в таблице ниже.

Таблица 14–12 – 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 и 12).

Пояснения к диаграмме (Рисунок 14–4), такты от T20 до T25, приведены в таблице ниже.

Таблица 14–13 – 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)

14.4.2.4 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.

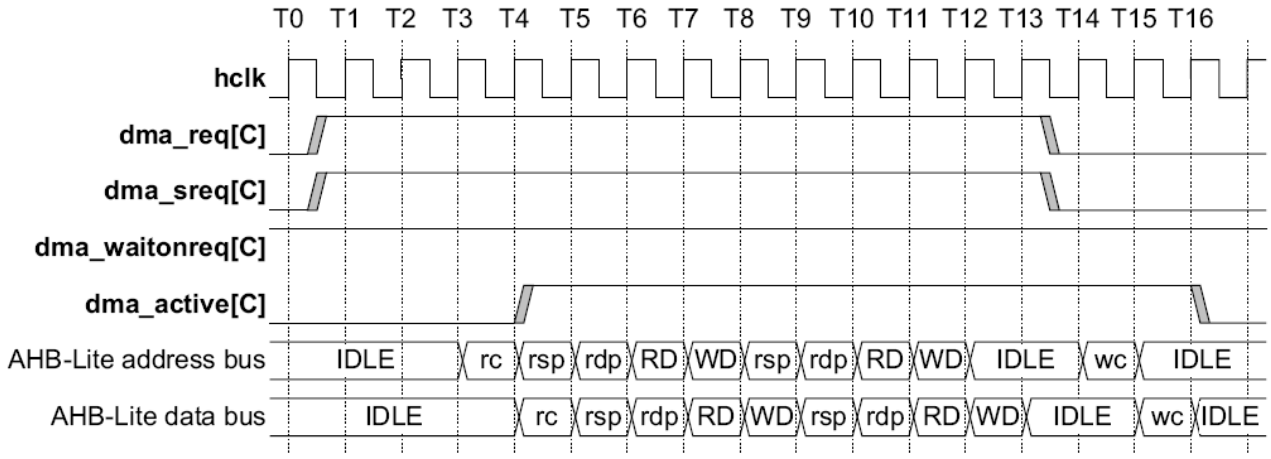


Рисунок 14–5 – Диаграмма работы контроллера DMA при использовании dma_waitonreq

Пояснения к диаграмме (Рисунок 14–5) приведены в таблице ниже.

Таблица 14–14 – 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 14–6 показывает работу контроллера DMA при установке dma_waitonreq в 1 и выполнении одиночной DMA передачи.

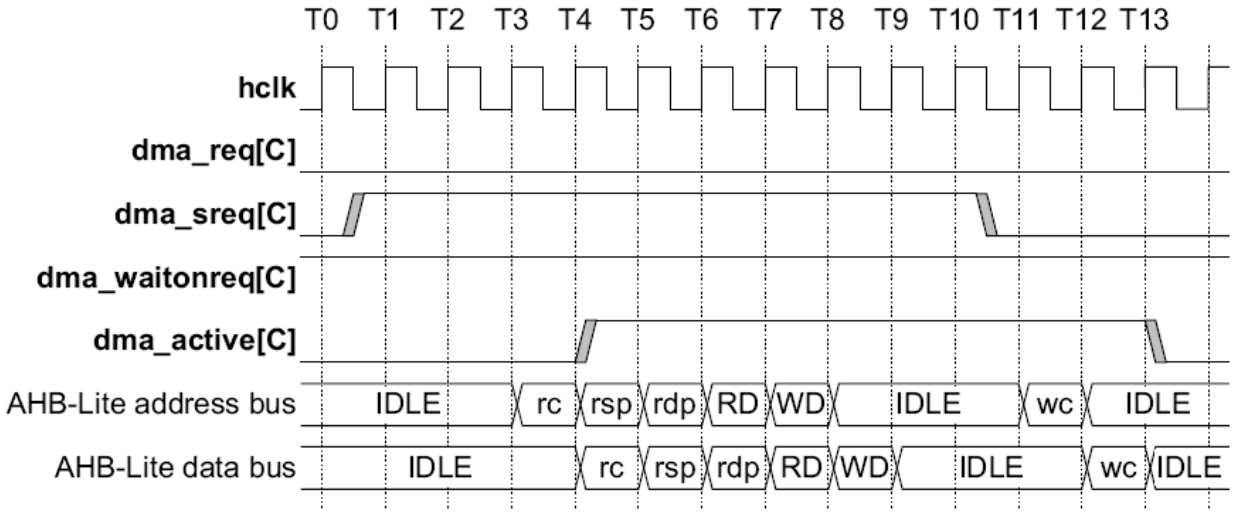


Рисунок 14–6 – Работа DMA при использовании dma_waitonreq совместно с dma_sreq

Пояснения к диаграмме (Рисунок 14–6) приведены в таблице ниже.

Таблица 14–15 – 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)

14.4.3 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 4 разряда, которые определяют количество транзакций по шине AHB до повторения арбитража. Эти разряды задают степень R числа 2; изменение R напрямую устанавливает периодичность арбитража как 2 в степени R. Для примера, если R равно 4, то арбитраж будет проводиться через каждые 16 передач DMA.

Таблица 14–16 показывает возможную периодичность арбитража.

Таблица 14–16 – Периодичность арбитража в единицах передач по шине AHB

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 разделе «Управляющие данные канала».

14.4.4 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 14–17 показывает уровень приоритета каналов DMA в порядке его уменьшения.

Таблица 14–17 – Уровень приоритета каналов DMA

Уровень приоритета в порядке его уменьшения	Номер канала	Уровень приоритета 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 14–7 иллюстрирует процесс выбора следующего канала для обслуживания.

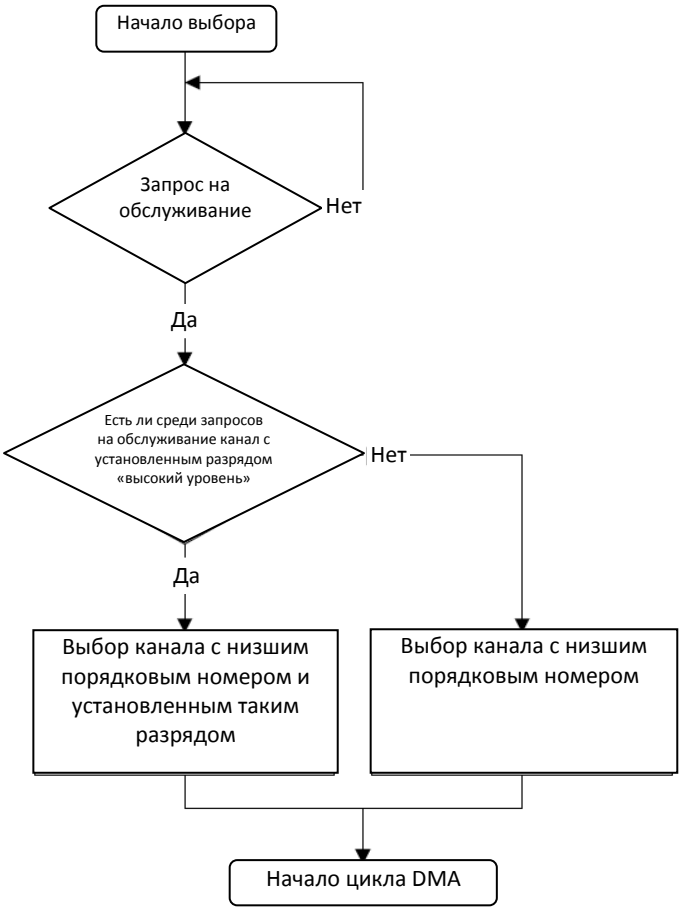


Рисунок 14–7 – Алгоритм выбора следующего канала для обслуживания

14.4.5 Типы циклов DMA

Разряды cycle_ctrl определяют, как контроллер будет выполнять циклы DMA. Описание значений этих разрядов приведено ниже.

Таблица 14–18 – 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 cycle_ctrl находятся в области памяти, отведенной под channel_cfg – см. раздел «Настройка управляющих данных канала».

Для всех типов циклов DMA повторный арбитраж происходит после 2R 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 2R 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».

14.4.5.1 Недействительный

После окончания цикла DMA контроллер устанавливает тип цикла в значение «недействительный» для предотвращения повтора выполненного цикла DMA.

14.4.5.2 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 2R 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.

14.4.5.3 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 2R передач для канала C. 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.

14.4.5.4 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 14–8 демонстрирует пример функционирования контроллера в режиме «пинг-понг».

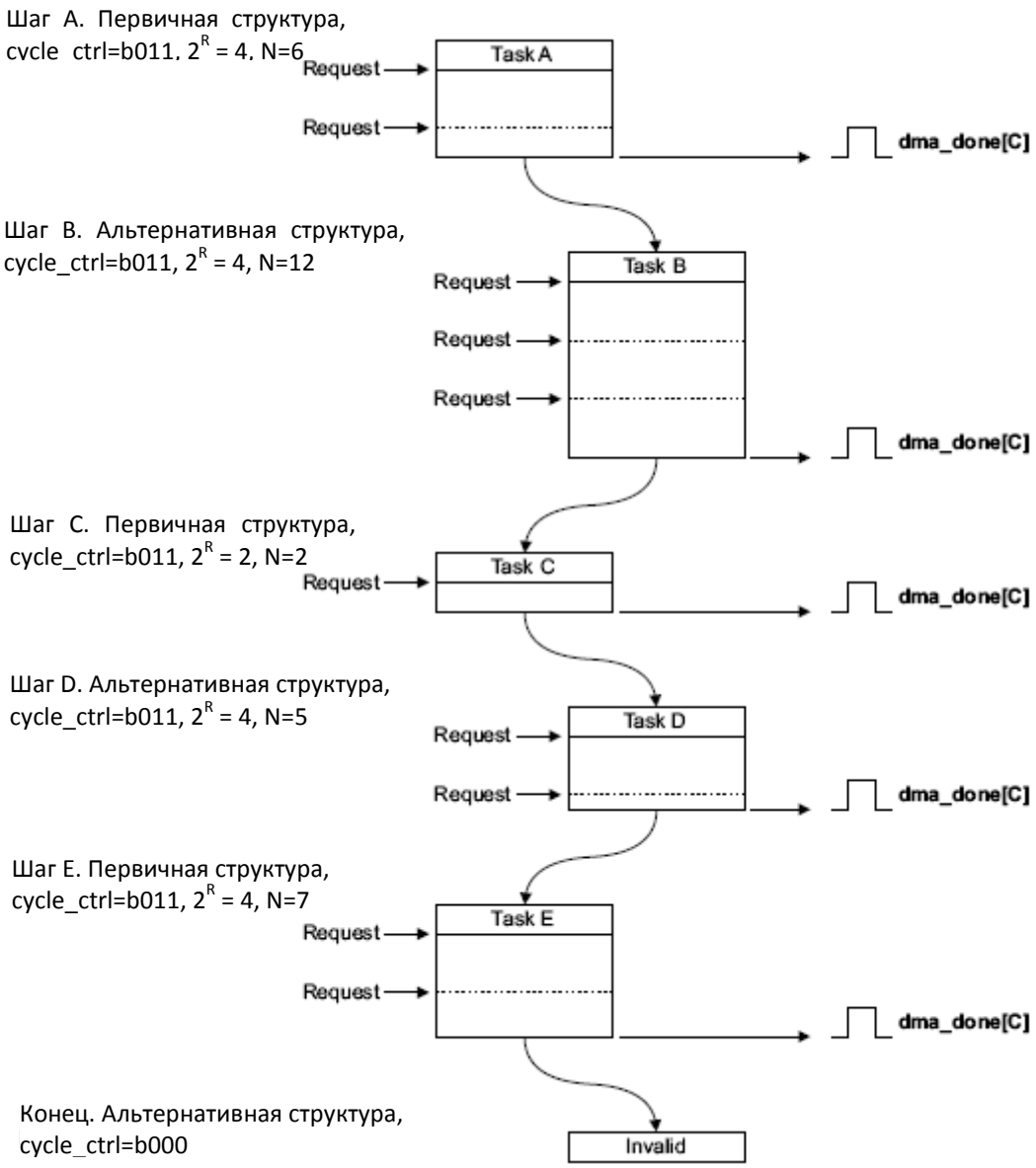


Рисунок 14–8 – Пример функционирования контроллера в режиме «пинг-понг»

Пояснения к схеме (Рисунок 14–8):

Шаг A Процессор устанавливает первичную структуру управляющих данных для шага A.

Процессор устанавливает альтернативную структуру управляющих данных для шага B. Это позволит контроллеру переключиться к шагу B незамедлительно после выполнения шага A, при условии, что контроллер не получит запрос на обработку от высокоприоритетного канала.

Контроллер получает запрос и выполняет 4 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 2 передачи DMA.

Контроллер устанавливает dma_done[C] в состояние 1 на один такт сигнала синхронизации hclk и входит в процедуру арбитража

После выполнения шага A процессор может установить первичные управляющие данные канала для шага C. Это позволит контроллеру переключиться

к шагу C незамедлительно после выполнения шага B, 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 исполняется шаг B:

Шаг B Контроллер выполняет 4 передачи DMA.
Контроллер выполняет арбитраж. После получения запроса на обработку от этого же канала контроллер продолжает цикл в ситуации отсутствия высокоприоритетных запросов.
Контроллер выполняет 4 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 4 передачи DMA.
Контроллер устанавливает dma_done[C] в состояние 1 на один такт сигнала синхронизации hclk и входит в процедуру арбитража

После выполнения шага B процессор может установить альтернативные управляющие данные канала для шага D.

После получения нового запроса на обработку от канала при условии его наивысшего приоритета исполняется шаг C:

Шаг C Контроллер выполняет 2 передачи DMA.
Контроллер устанавливает dma_done[C] в состояние 1 на один такт сигнала синхронизации hclk и входит в процедуру арбитража

После выполнения шага C процессор может установить первичные управляющие данные канала для шага E.

После получения нового запроса на обработку от канала при условии его наивысшего приоритета исполняется шаг D:

Шаг D Контроллер выполняет 4 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 исполняется шаг E:

Шаг E Контроллер выполняет 4 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 3 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 окончанию 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 E в режим «Основной цикл DM» путем установки cycle_ctrl в 3'b001.

14.4.5.5 Режим работы с памятью «исполнение с изменением конфигурации»

В данном режиме контроллер, получая начальный запрос на обработку, выполняет 4 передачи DMA, используя первичные управляющие данные. По окончании этих передач контроллер начинает цикл DMA используя альтернативные управляющие данные. Затем контроллер выполняет еще 4 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 14–19 перечисляет области памяти channel_cfg, как те, которые должны быть определены константами, так и те, значения которых определяются пользователем.

Таблица 14–19 – Channel_cfg для первичной структуры управляющих данных в режиме работы с памятью «исполнение с изменением конфигурации»

№ 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 4 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 14–9 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=b100, 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	0xXXXXXXXX
Data for Task B	0x0B000000	0x0BE00000	cycle_ctrl = b101, 2^R = 2, N = 8	0xXXXXXXXX
Data for Task C	0x0C000000	0x0CE00000	cycle_ctrl = b101, 2^R = 8, N = 5	0xXXXXXXXX
Data for Task D	0x0D000000	0x0DE00000	cycle_ctrl = b001, 2^R = 4, N = 4	0xXXXXXXXX

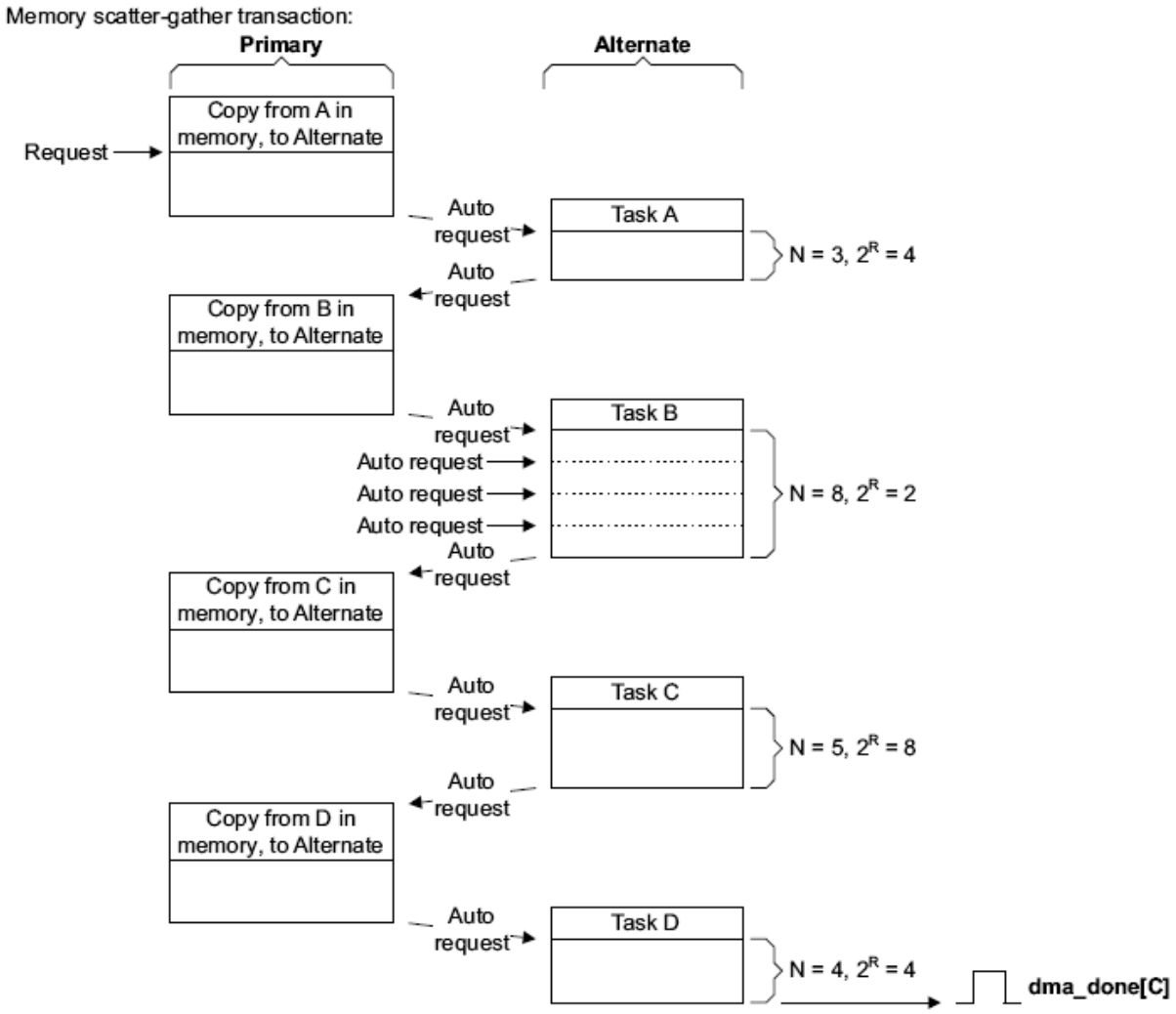


Рисунок 14–9 – Пример работы DMA в режиме «Исполнение с изменением конфигурации»

Пояснения к схеме (Рисунок 14–9):

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 4 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 A

По получению запроса на обслуживание контроллер выполняет 4 передачи DMA. Эти передачи записывают альтернативную структуру управляющих данных для шага A.

Контроллер генерирует автозапрос для канала, после чего проводит процедуру арбитража.

Шаг A

Контроллер выполняет шаг A. По окончании контроллер генерирует автозапрос для канала и проводит процедуру арбитража.

Первичная, копирование B

Контроллер выполняет 4 передачи DMA. Эти передачи записывают альтернативную структуру управляющих данных для шага B.

Контроллер генерирует автозапрос для канала, после чего проводит процедуру арбитража.

Шаг B

Контроллер выполняет шаг B. По окончании контроллер генерирует автозапрос для канала и проводит процедуру арбитража.

Первичная, копирование C

Контроллер выполняет 4 передачи DMA. Эти передачи записывают альтернативную структуру управляющих данных для шага C.

Контроллер генерирует автозапрос для канала, после чего проводит процедуру арбитража.

Шаг C

Контроллер выполняет шаг C. По окончании контроллер генерирует автозапрос для канала и проводит процедуру арбитража.

Первичная, копирование D

Контроллер выполняет 4 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.

14.4.5.6 Режим работы с периферией «исполнение с изменением конфигурации»

В данном режиме контроллер, получая начальный запрос на обработку, выполняет 4 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 4 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 14–20 перечисляет области памяти channel_cfg, как те, которые должны быть определены константами, так и те, значения которых определяются пользователем.

Таблица 14–20 – Channel_cfg для первичной структуры управляющих данных в режиме работы с периферией «Исполнение с изменением конфигурации»

№ 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 4 передачи DMA
2…0	cycle_ctrl	b’110	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
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	0xXXXXXXXX
Data for Task B	0x0B000000	0x0BE00000	cycle_ctrl = b111, 2^R = 2, N = 8	0xXXXXXXXX
Data for Task C	0x0C000000	0x0CE00000	cycle_ctrl = b111, 2^R = 8, N = 5	0xXXXXXXXX
Data for Task D	0x0D000000	0x0DE00000	cycle_ctrl = b001, 2^R = 4, N = 4	0xXXXXXXXX

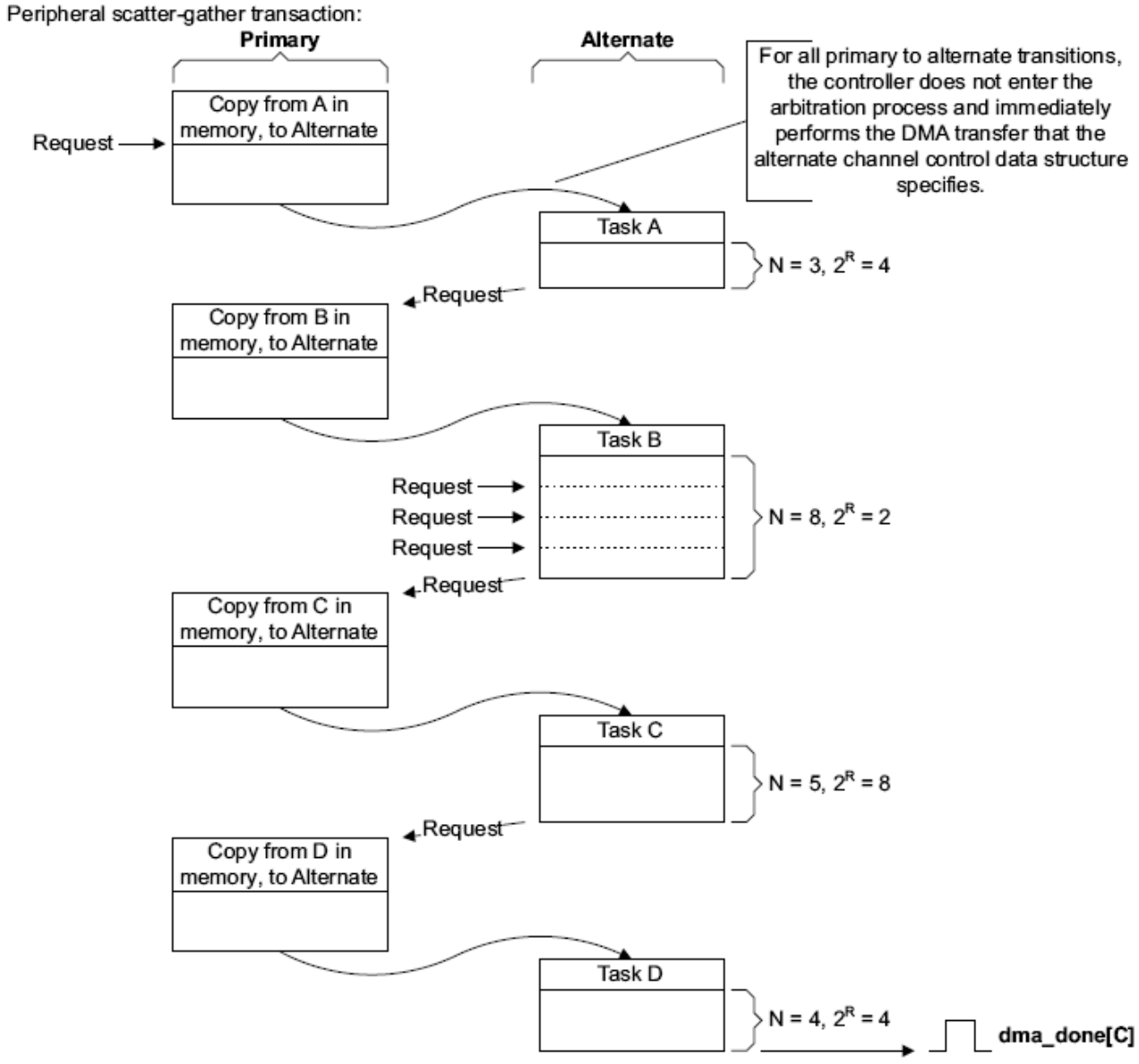


Рисунок 14–10 – Пример работы DMA в режиме с «Исполнением с изменением конфигурации»

Пояснения к схеме (Рисунок 4–10)

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 4 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 A памяти

По получению запроса на обслуживание, контроллер выполняет 4 передачи DMA. Эти передачи записывают альтернативную структуру управляющих данных для шага A.

Шаг A

Контроллер выполняет шаг A.

По окончании контроллер проводит процедуру арбитража.

Первичная, копирование из области B памяти

Контроллер выполняет 4 передачи DMA. Эти передачи записывают альтернативную структуру управляющих данных для шага B.

Шаг B

Контроллер выполняет шаг B. Для завершения задачи периферия должна установить последовательно 3 запроса.

По окончании контроллер проводит процедуру арбитража.

Первичная, копирование из области C памяти

Контроллер выполняет 4 передачи DMA. Эти передачи записывают альтернативную структуру управляющих данных для шага C.

Шаг C

Контроллер выполняет шаг C.

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 4 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.

14.4.6 Индикация ошибок

При получении контроллером по шине AHB 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.

14.5 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 14–11 показывает область памяти, необходимую контроллеру для структур управляющих данных канала, при использовании всех 32 каналов и опциональной альтернативной структуры управляющих данных.

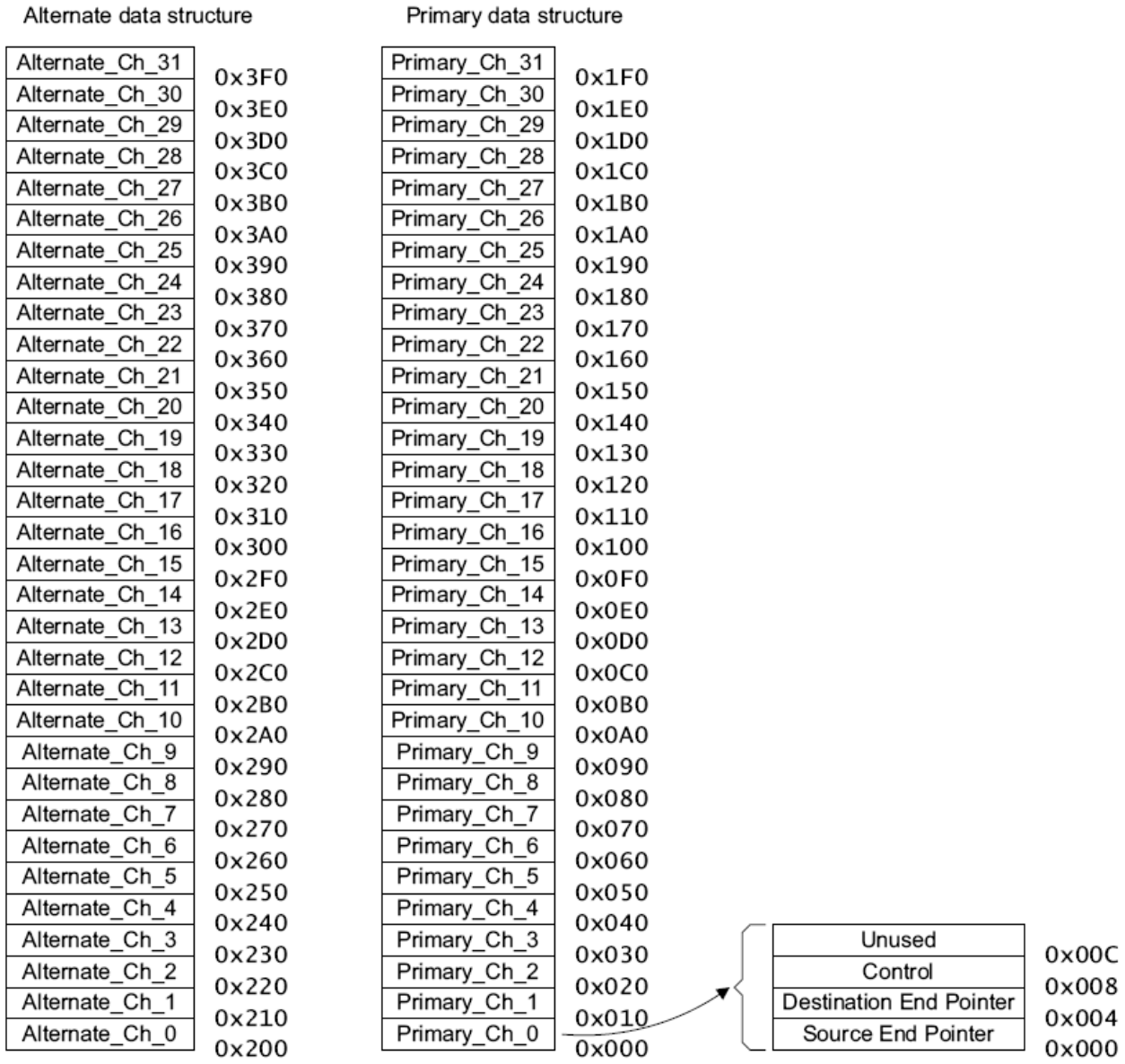


Рисунок 14–11 – Карта памяти для 32-х каналов, включая альтернативную структуру

Пример, приведенный выше (см. Рисунок 14–11), 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 14–21 перечисляет разряды адреса, обеспечивающие контроллеру доступ к различным элементам структуры управляющих данных, в зависимости от количества каналов, используемых в контроллере.

Таблица 14–21 – 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 14–12 демонстрирует пример реализации контроллера с использованием 3 каналов DMA и с альтернативной структурой управляющих данных.

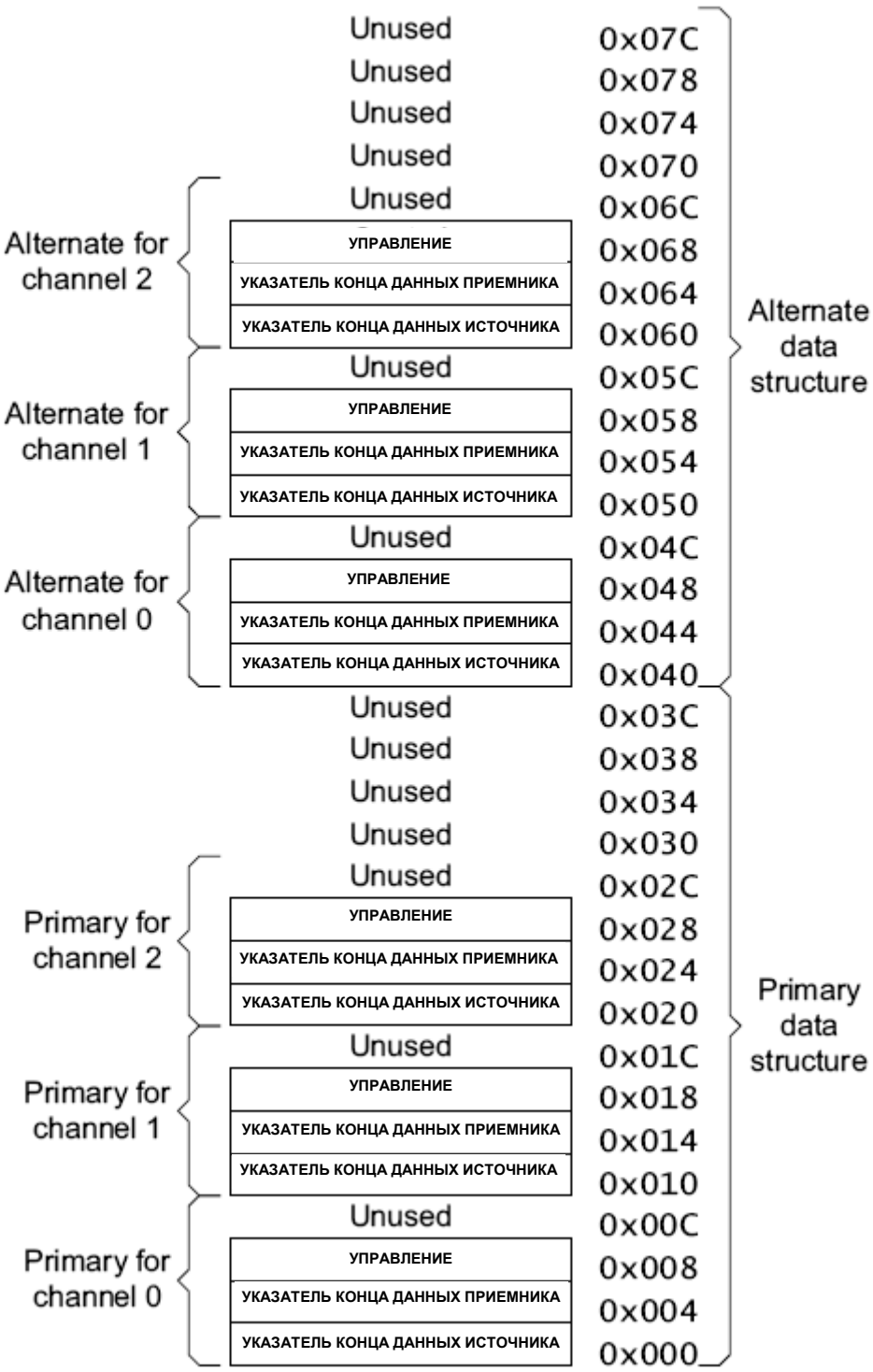


Рисунок 14–12 – Карта памяти для трех каналов DMA, включая альтернативную структуру

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 14–22 перечисляет все разрешенные значения базового адреса для первичной структуры управляющих данных в зависимости от количества каналов DMA, использованных в контроллере.

Таблица 14–22 – Разрешенные базовые адреса

Количество каналов DMA	Разрешенные значения базового адреса для первичной структуры управляющих данных
17-32	0xXXXXX000, 0xXXXXX400, 0xXXXXX800, 0xXXXXXC00

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 14–23 – Значения разрядов src_data_end_ptr

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31…0	src_data_end_ptr	Указатель последнего адреса данных источника

Перед тем, как контроллер выполнит передачу DMA, необходимо определить эту область памяти. Контроллер считывает значение этой области перед началом 2R 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 14–24 – Значения разрядов dst_data_end_ptr

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31…0	dst_data_end_ptr	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 2R 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 14–25 – 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_crtl**	**Src_prot_ctrl**	**R_power**	**n_minus_1**	**next_useburst**	**cycle_ctrl**

Таблица 14–26 объясняет назначение разрядов этой области памяти.

Таблица 14–26 – Назначение разрядов channel_cfg

№ 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
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	Задает 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	Задает 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	Задает 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. 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	Задает 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 задает 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. *Примечание* – Перед завершением цикла DMA, использующего альтернативные управляющие данные, контроллер устанавливает chnl_useburst_set[C] в значение 0, если количество оставшихся передач DMA меньше, чем 2^R. Установка next_useburst разряда определяет, будет ли контроллер дополнительно переопределять разряд chnl_useburst_set[C]. 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 next_useburst: 0 – контроллер не изменяет значение chnl_useburst_set[C]. Если chnl_useburst_set[C] установлен в 0, то для всех оставшихся циклов DMA в режиме работы с периферией «Исполнение с изменением конфигурации», контроллер отвечает на запросы по dma_req[] и dma_sreq[], при выполнении циклов DMA он использует альтернативные управляющие данные. 1 – контроллер изменяет значение chnl_useburst_set[C] в состояние 1. Поэтому для оставшихся циклов DMA в режиме работы с периферией «Исполнение с изменением конфигурации», контроллер реагирует только на запросы по dma_req[], при выполнении циклов DMA он использует альтернативные управляющие данные.
2…0	cycle_ctrl	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 cycle_ctrl поле в состояние b001 или b 010; 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 2R 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 14–27 перечисляет адреса приемника цикла DMA для 6 слов.

Таблица 14–27 – Цикла DMA для 6 слов с пословным инкрементом

Начальные значения channel_cfg перед циклом DMA				
src_size=b10, dst_inc=b10, n_minus_1=b101, cycle_ctrl=1				
DMA передачи	**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 14–28 перечисляет адреса приемника для передач DMA 12 байт с использованием «полусловного» инкремента.

Таблица 14–28 – 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 2R 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.

14.6 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 14–29 – 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
0x00C	ALT_CTRL_BASE_PTR	RO	0x000000nn**)	MDR_DMA->ALT_CTRL_BASE_PTR Регистр базового адреса альтернативных 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 управляющих данных каналов
0x034	CHNL_PRI_ALT_CLR	WO	-	MDR_DMA->CHNL_PRI_ALT_CLR Регистр сброса первичной/альтернативной структуры управляющих данных каналов
0x038	CHNL_PRIORITY_SET	R/W	0x00000000	MDR_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.

14.6.1 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 14–31 перечисляет назначение разрядов регистра.

Таблица 14–30 – 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 14–31 – Назначение разрядов регистра dma_status

№ 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
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

14.6.2 MDR_DMA->CFG

Регистр конфигурации DMA

Данный регистр имеет доступ только на запись. Регистр определяет состояние контроллера.

Таблица 14–33 перечисляет назначение разрядов регистра.

Таблица 14–32 – 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 14–33 – Назначение разрядов регистра dma_cfg

№ 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 AHB-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; 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

14.6.3 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 14–35 перечисляет назначение разрядов регистра ctrl_base_ptr.

Таблица 14–34 – Регистр базового адреса управляющих данных каналов

Номер	31...10	9...0
Доступ	R/W	U
Сброс	0	0
	ctrl_base_ptr	-

Таблица 14–35 – Назначение разрядов регистра ctrl_base_ptr

№ 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

14.6.4 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 14–37 перечисляет назначение разрядов регистра.

Таблица 14–36 – Регистр базового адреса альтернативных управляющих данных каналов

Номер	31... 0
Доступ	RO
Сброс	0
	Alt_ctrl_base_ptr

Таблица 14–37 – Назначение разрядов регистра alt_ctrl_base_ptr

№ 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

14.6.5 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 14–39 перечисляет назначение разрядов регистра.

Таблица 14–38 – Регистр статуса ожидания запроса на обработку каналов

Номер	31	**.....**	2	1	0
Номер	RO	**.....**	RO	RO	RO
Доступ	0	**.....**	0	0	0
	dma_waitonreg_status for dma_waitnreg [31]	**.....**	**dma_waitonreg_status for dma_waitnreg [2]**	**dma_waitonreg_status for dma_waitnreg [1]**	**dma_waitonreg_status for dma_waitnreg [0]**

Таблица 14–39 – Назначение разрядов регистра dma_waitonreq_status

№ 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

14.6.6 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 14–41 перечисляет назначение разрядов регистра.

Таблица 14–40 – 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 14–41 – Назначение разрядов регистра chnl_sw_request

№ 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

14.6.7 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 14–43 перечисляет назначение разрядов регистра.

Таблица 14–42 – 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 14–43 – Назначение разрядов регистра chnl_useburst_set

№ 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 состоянии этих настроек. **При чтении:** Разряд [C] =0 означает, что канал DMA C выполняет циклы DMA в ответ на запросы, полученные от dma_sreq[] идma_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.

14.6.8 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 14–45 перечисляет назначение разрядов регистра chnl_useburst_clr.

Таблица 14–44 – 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 14–45 – Назначение разрядов регистра chnl_useburst_clr

№ 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

14.6.9 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 14–47 перечисляет назначение разрядов регистра chnl_req_mask_set.

Таблица 14–46 – 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_reg_mask_set for dma_reg [31] and dma_sreg [31]	**.....**	**chnl_reg_mask_set for dma_reg [2] and dma_sreg [2]**	**chnl_reg_mask_set for dma_reg [1] and dma_sreg [1]**	**chnl_reg_mask_set for dma_reg [0] and dma_sreg [0]**

Таблица 14–47 – Назначение разрядов регистра chnl_req_mask_set

№ 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

14.6.10 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 14–49 перечисляет назначение разрядов регистра chnl_req_mask_clr.

Таблица 14–48 – Регистр очистки маскирова ния запросов на обслуживание каналов

Номер	31	**.....**	2	1	0
Доступ	WO	**.....**	WO	WO	WO
Сброс	0	**.....**	0	0	0
	chnl_reg_mask_clr for dma_reg [31] and dma_sreg [31]	**.....**	**chnl_reg_mask_clr for dma_reg [2] and dma_sreg [2]**	**chnl_reg_mask_clr for dma_reg [1] and dma_sreg [1]**	**chnl_reg_mask_clr for dma_reg [0] and dma_sreg [0]**

Таблица 14–49 – Назначение разрядов регистра chnl_req_mask_clr

№ 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

14.6.11 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 14–51 перечисляет назначение разрядов регистра chnl_enable_set.

Таблица 14–50 – Регистр установки разрешения каналов

Номер	31	**.....**	2	1	0
Доступ	WO	**.....**	WO	WO	WO
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 14–51 – Назначение разрядов регистра chnl_enable_set

№ 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

14.6.12 MDR_DMA->CHNL_ENABLE_CLR

Регистр сброса разрешения каналов

Данный регистр имеет доступ только на запись. Регистр запрещает работу каналов DMA.

Таблица 14–53 перечисляет назначение разрядов регистра chnl_enable_clr.

Таблица 14–52 – 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 14–53 – Назначение разрядов регистра chnl_enable_clr

№ 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

14.6.13 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 14–55 перечисляет назначение разрядов регистра chnl_pri_alt_set.

Таблица 14–54 – 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 14–55 – Назначение разрядов регистра chnl_pri_alt_set

№ 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 4-х 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»; – 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;

14.6.14 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 14–57 перечисляет назначение разрядов регистра chnl_pri_alt_clr.

Таблица 14–56 – 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 14–57 – Назначение разрядов регистра chnl_pri_alt_clr

№ 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 4-х 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»; – 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»

14.6.15 MDR_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 14–59 перечисляет назначение разрядов регистра chnl_priority_set.

Таблица 14–58 – 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 14–59 – Назначение разрядов регистра chnl_priority_set

№ 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 уровеньприоритета. Запись разряда соответствующего нереализованному каналу не дает никакого эффекта

14.6.16 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 14–61 перечисляет назначение разрядов регистра chnl_priority_clr.

Таблица 14–60 – 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 14–61 – Назначение разрядов регистра chnl_priority_clr

№ 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

14.6.17 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 14–63 перечисляет назначение разрядов регистра err_clr.

Таблица 14–62 – Регистр сброса флага ошибки

Номер	31…1	0
Доступ	U	R/W
Сброс	0	0
	-	**err_clr**

Таблица 14–63 – Назначение разрядов регистра err_clr

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31…1	-	Не определено. Следует записывать 0
0	chnl_priority_set	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. Для целей тестирования возможно использовать регистр err_set, чтобы установить сигнал dma_err в состояние 1. *Примечание* – При сбросе сигнала dma_err одновременно с появлением ошибки на шине AHB-Lite, то приоритет отдается ошибке и следовательно, значение регистра (и dma_err) останется неизменным (несброшенным)

15 Организация управления DSP подсистемой

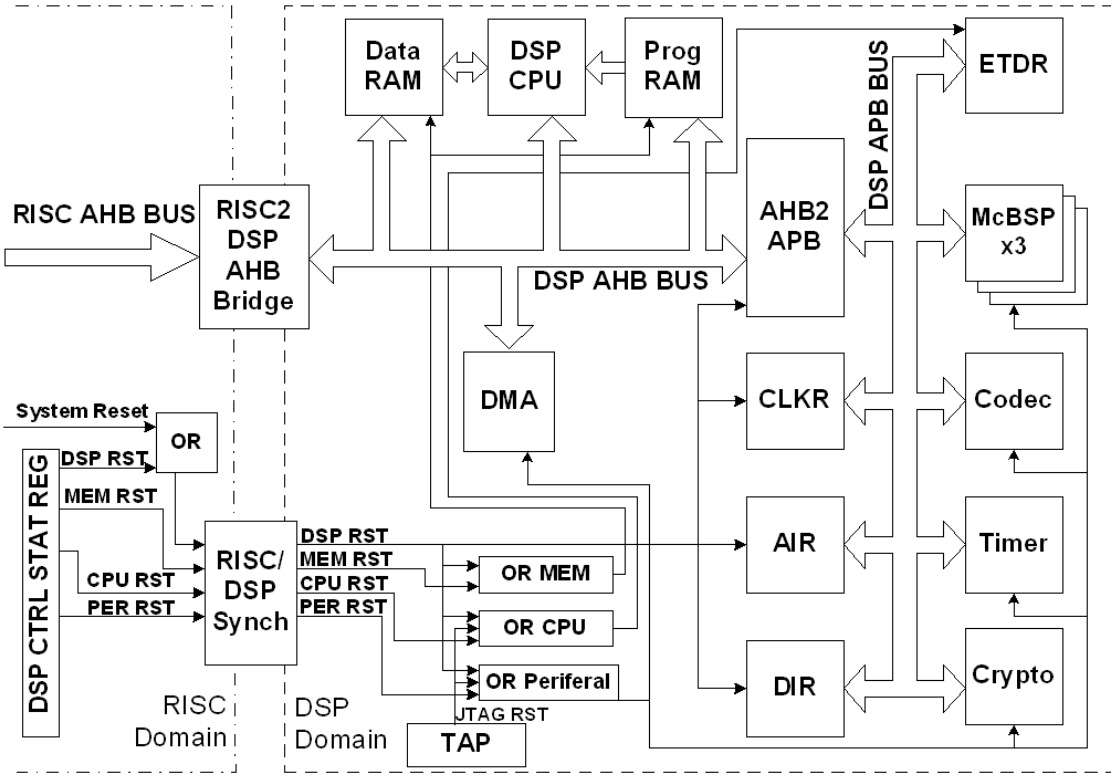


Рисунок 15–1 – Структурная схема подсистемы DSP

15.1 Средства управления подсистемой DSP

При включении питания DSP подсистема не активна (находится в сбросе, тактовая частота на подсистему не подается). Активировать DSP подсистему можно только при помощи RISC. Для этого RISC ядро обладает следущими средствами:

- Управление тактовой частотой DSP. Для управления тактовой частотой DSP в системе реализован регистр управления тактовой частой DSP DSP_CLOCK в RISC подсистеме и CLKMD в DSP подсистеме. Возможны различные варианты источника синхросигнала DSP (в т. ч. в системе реализована отдельная PLL для DSP). Возможно отключение тактовых сигналов от отдельных модулей DSP подсистемы.
- 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.

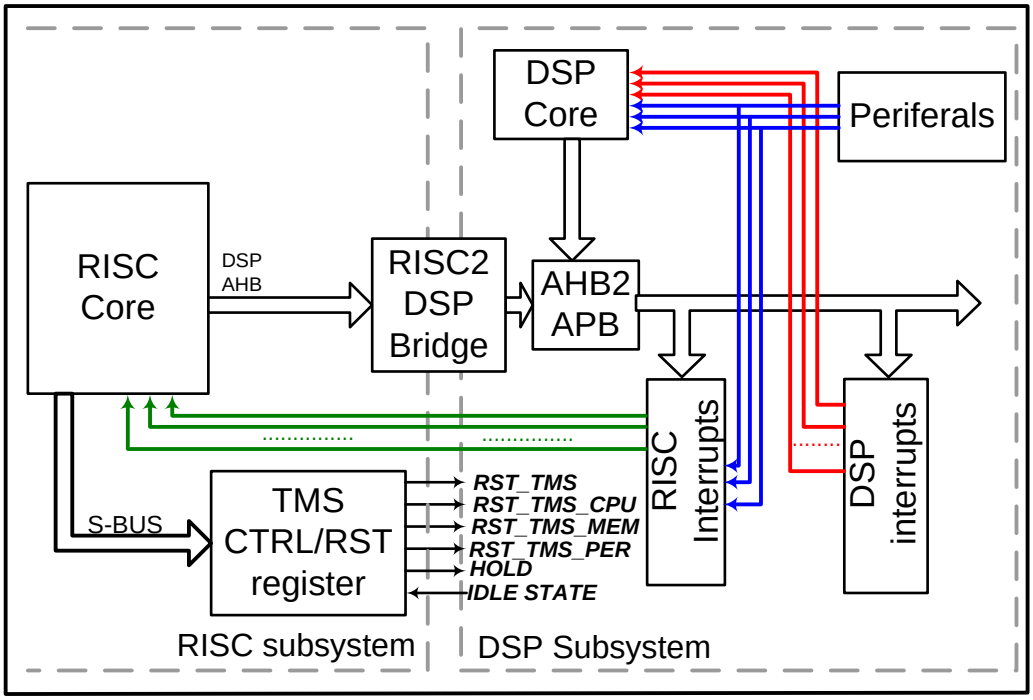


Рисунок 15–2 – Организация управления подсистемой DSP

15.2 Синхросигналы DSP подсистемы

Первичными синхросигналами для подсистемы DSP являются:

- синхросигнал CLK_DSP, который формируется в блоке контроллера тактовой частоты RISC, регистр DSP_CLOCK;
- синхросигнал TCK, который формируется аппратным 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. Т.е. в случае начала работы с TAP контроллером DSP средствами отладочного интерфейса JTAG через среду Code Composer вся система переходит на синхросигнал TCK. Если отладка DSP ядра не производится (в т.ч. в случае отладки только RISC ядра), 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, регистр ETDR. В случае отсутствия сигнала CLK_MUX ни одна часть подсистемы не активна.

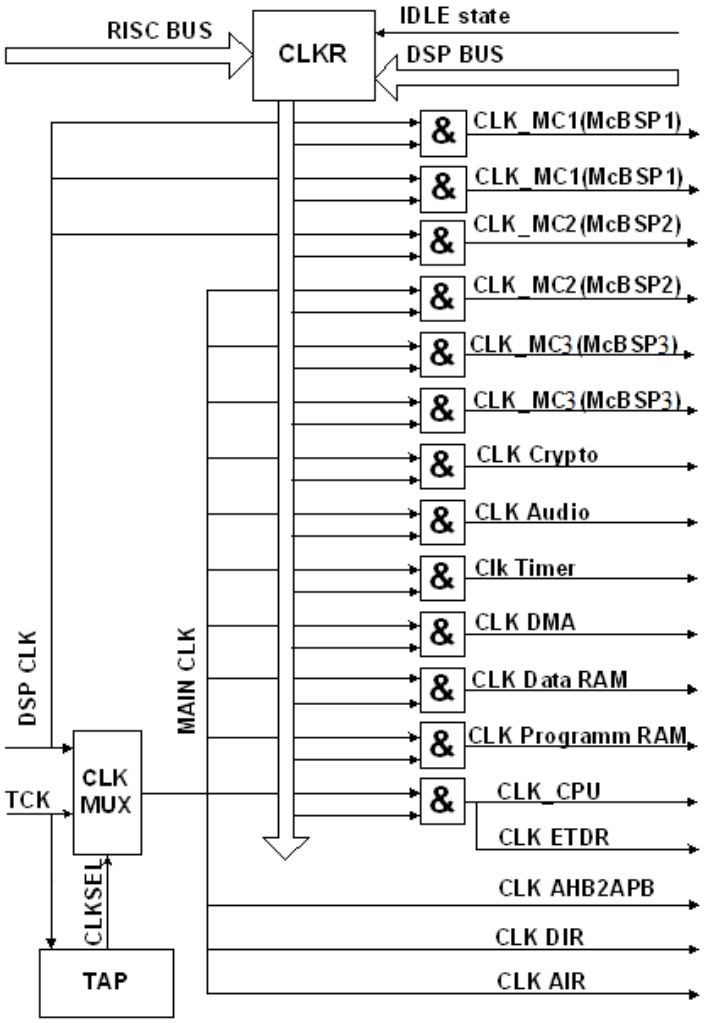


Рисунок 15–3 – Структурная схема дерева синхросигналов подсистемы DSP

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.

15.2.1 Режимы отключения синхросигналов DSP подсистемы (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 DSP ядра, 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 пониженого 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,

DSP ядро 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.

15.3 Сбросы DSP подсистемы

Управление сбросами устройств подсистемы DSP может быть осуществлено только со стороны RISC. Для этого в регистр DSP_CONTROL_STATUS включены биты кправления сбросами: RST_DSP, RST_DSP_CPU, RST_DSP_MEM, RST_DSP_PER.

Общий сброс DSP подсистемы

Бит RST_DSP в регистре DSP_CONTROL_STATUS служит для общего сброса подсистемы DSP. В началиный 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 быдут 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

Даный бит управляет сбросом ядра DSP. После подачи питания находится в активном состоянии (ядро сброшено).

Сброс памяти DSP

Даный бит 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

Даный бит управляет сбросом всех периферийных модулей подсистемы DSP. После подачи питания находится в активном состоянии (периферия сброшена).

15.4 Прерывания и запросы DMA между подсистмемами

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 состороны 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 следущее слово (Таблица 15–1 и Таблица 15–2):

1. В бит SNR вносится значение, которое тербуется 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 единицу.

Т.е. 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 следущее слово (таблица 94):

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 15–1 – 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 15–2 – 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)

15.5 Специальные возможности управления

Сигналы HOLD, HOLDA

В микроконтролере реализована возможность остановки конвейера DSP со стороны RISC. Для этого в регистр DSP_CONTROL_STATUS включены биты HOLD и HOLDA. Активный уровень обоихсигналов – единица. Для остановки конвейра установить бит HOLD. При этом конвейер процессора остановиться и бит HOLDA утановиться в 1.

Флаги BIO, XF

Так же в системе реализованы флаги BIO и XF. Состояние данных флагов может быть обработано специальными командами процессора DSP.

В 3 ревизии микросхемы 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 з-арезервированный бит XF_EN, переводящий вывод PA[15] в режим вывода флага XF.

Таким образом для прямого вывода флага XF необходимо:

- Перевести порт PA[15] микроконтроллера в режим «Переопределенной и DSP функции». Порт, при этом, будет работать в режиме внешнего прерывания.
- Установить бит XF_EN регистра «Управления и статуса DSP». Порт переключится на выход и будет транслировать состояние флага XF.

Данная функция реалищована для упрощения отладки программ DSP ядра.

15.6 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 DSP частота 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 убедится 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.

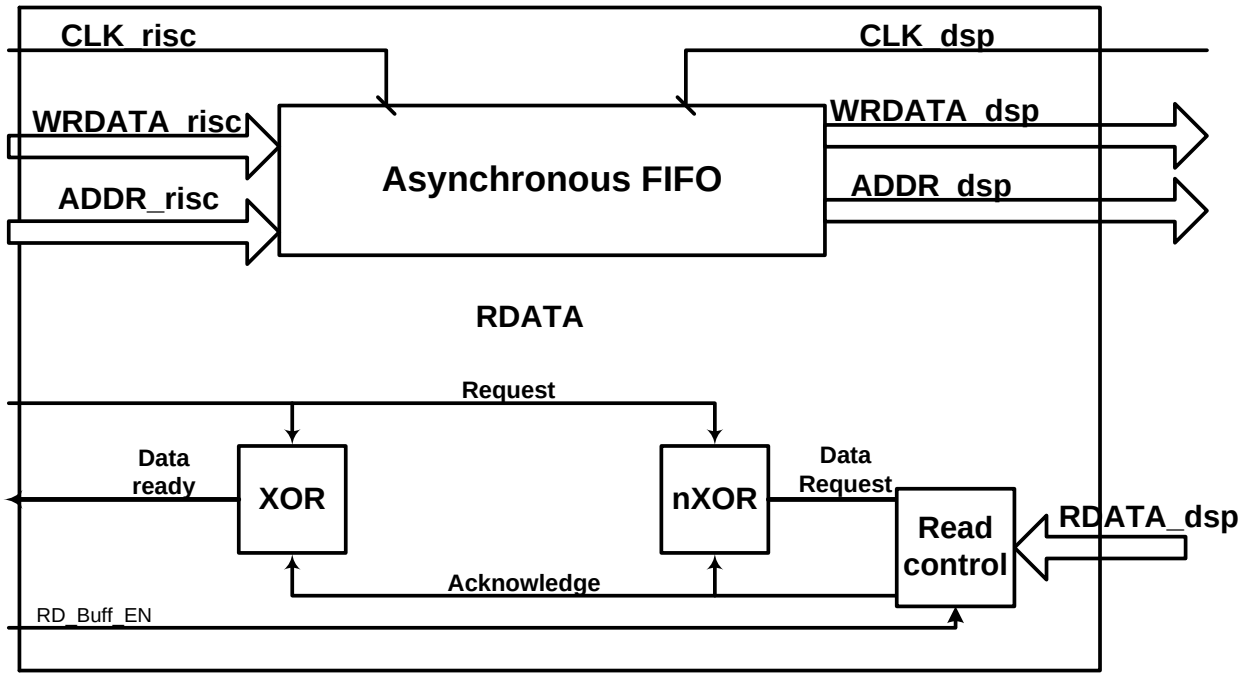


Рисунок 15–4– Структура буфера пересинхронизации RISC– DSP

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. Нерекомендуется данный режим при обращении к регистрам (т.к. автоматическое чтение некоторых регистров может изменить их значение). После подачи питания механизм предчтения данных неактивен.

15.7 Способы управления подсистемой DSP

15.7.1 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 программи данных DSP образ программного обеспечения (программу, необходимые константы, таблицу векторов прерыаний).
4. Снять сброс с ядра DSP и, при необходимости, с периферии (регистр DSP_CONTROL_STATUS).

После выполнения всех операций DSP ядро начнетсвою работу в штатном режиме, как автономный DSP процессор. Для окончания работы ядром DSP необходимо только установить сброс DSP ядра. После чего можетбыть изменен весь образ памяти DSP, либо отдельные функции. Так же может быть прекращена подача синхросигнала DSP.

15.7.2 Динамическое изменение программного обеспечения DSP ядра

Изменения программы DSP без остановки ядра может быть осуществлено как при помощи RISC так и самим DSP.
Для динамического измененипрограммы 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 расслоена память, позволяющая одновременное безконфликтное обращение двух устройств.
2. Занести в свободный участок памяти новую функцию, требуемую к исполнению.
3. Уведомить DSP ядро о готовность 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 DSP-подсистемы, либо DMA-RISC, если программа находится вне DSP-подсистемы. DSP ядро 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 DSP подсистеме. Включить каналы соответствующие запросам DSP.
2. Со стороны DSP указать в упраляющейструктуре адрес источника и назначения для данных.
3. Подать DMA запрос на контроллер DMA RISC (регистр AIRQ).
4. Периодически проверять готовность данных (регистр AIRQ).

15.7.3 Работа с периферийными модулями и памятью DSP без участия ядра DSP

В случае необходимости, RISC может использоватьвсе периферийные модули и память безучастия 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.

15.8 Регистры управления подсистемой DSP

Для управления DSP частью в системе реализован набор регистров (Таблица 15–3).

Таблица 15–3 – 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 15–4 – Регистр управления и статуса DSP

15	14	13…12	11	10	9	7…6	5	4	3	2	1	0
RD_Buff_EN	BRTRD	.	IDLE num1		HOLDA	.	BIO	HOLD	RST_DSP_ PER	RST_DSP_MEM	RST_DSP_CPU	RST_DSP

Таблица 15–5 – 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.
13	-	Зарезервировано.
12	-	Зарезервировано.
11-10	IDLE num1	Код состояния DSP. 00 – DSP подсистема 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 – Конвйер DSP остановлен. 1 – Конвейер DSP работает (Не было запроса, либо конвейер еще не остановлен).
8	XF	Флаг общего назначения DSP к RISC. Обрабатывается

		специальными командами DSP.
7	-	Зарезервировано
6	XF_EN	Разрешение вывода сигнала XF напрямую на порт PA [15] микроконтроллера
5	BIO	Флаг общего назначения от RISC к DSP. Обрабатывается 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 находятся в сбросе.
0	RST_DSP	Общий сброс DSP части устройства. 0 – DSP подсистема работает. 1 – DSP подсистема находистя в сбросе.

15.8.1 Регистр управления синхронизацией CLKMD

Таблица 15–6 – Регистр управления синхронизацией CLKMD

15	14	13	12	11	10	9	8	7	6	5	4	3	2-0
CPU	CPM	CDM	PC1	MC1	PC2	MC2	PC3	MC3	DMA	TMR	CDC	CRP	-

Таблица 15–7 – 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 15–8 – Регистр прерываний от RISC к DSP и запросовперываний 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
	ADMARQ1	**ADMARQ0**	**-**	**ANMI**	**AIRQ3**	**AIRQ2**	**AIRQ1**	**AIRQ0**

Таблица 15–9 – Описание бит регистра прерываний от RISC к DSP и запросовперываний для RISC AIRQ

№ 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 передачиданных DMA RISC
8	ADMARQ2	Бит запроса передачиданных DMA RISC
7	ADMARQ1	Бит запроса передачиданных DMA RISC
6	ADMARQ0	Бит запроса передачиданных DMA RISC
5	-	–
4	ANMI	Немаскируемоепрерываване 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

15.8.2 Регистр прерываний от DSP к RISC

Таблица 15–10 – 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 15–11 – Описание бит регистра запросов прерываний от DSP к RISC

№ 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

16 Процессорное ядро DSP

16.1 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, совмещённый 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 (ARAUs).
- Инструкции повторения одной команды и повторения блока программного кода.
- Команды с 32-разрядными длинными словами.
- Команды с чтением двух и трёх 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 трёх слов).
- 6-стадийный конвейер обработки.
- Аппаратное разрешение кофликтов конвейерной обработки.

16.2 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.

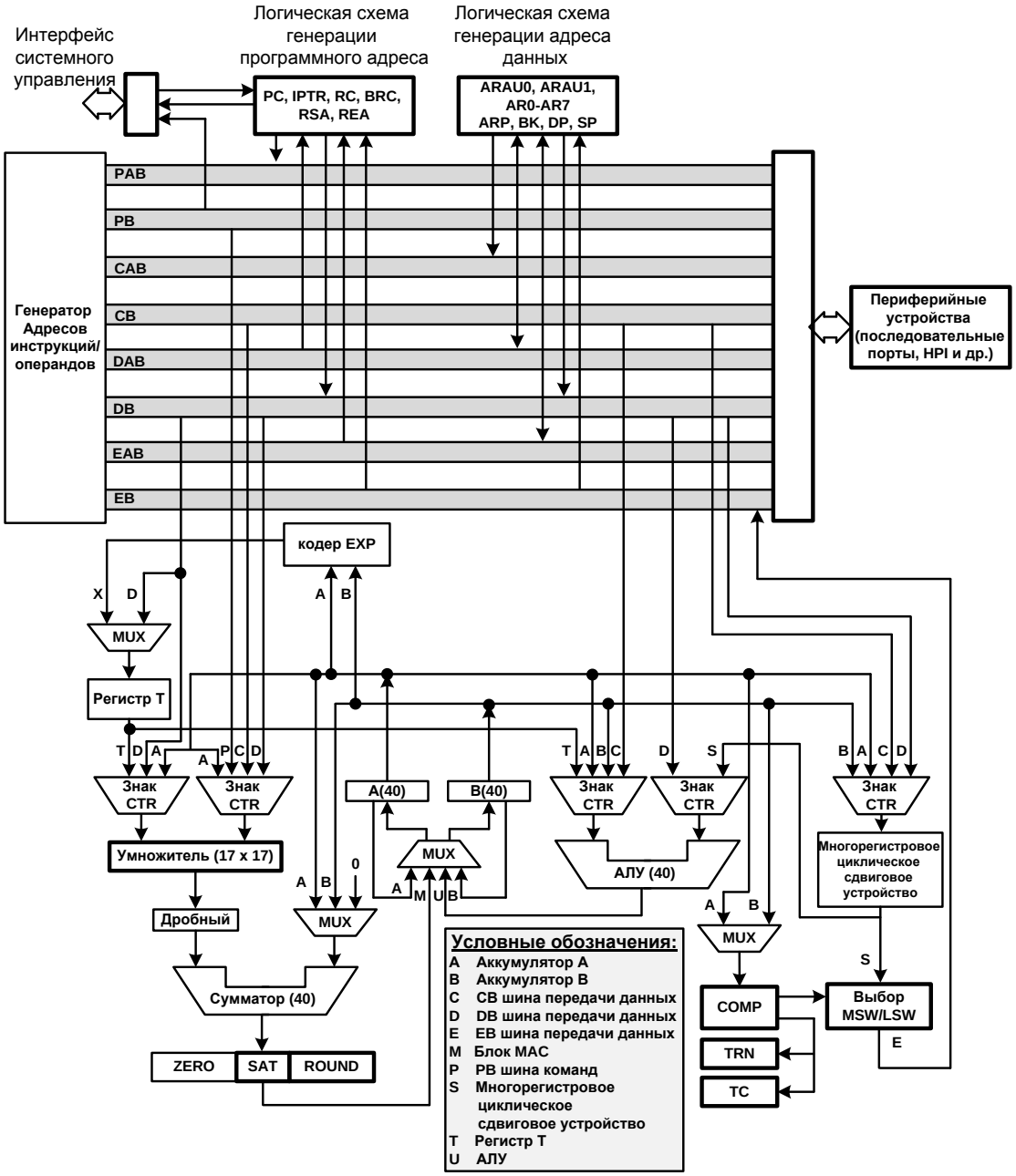


Рисунок 16–1 – Блок-схема микропроцессорного ядра

16.2.1 Шинная структура DSP

Процессороное 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 записьрезультата.

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 микроконтроллера все шины ядра преобразуются в шины стандарта AMBA. Обращение к периферийным модулям ЦОС части устройства происходит по шине APB, к памяти по шине AHB.

16.2.2 Устройство управления выполнением программ

Устройство управления выполненят следущие 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 программы включают команды переходов, вызовы подпрограмм, условные команды, повторение команд, сброс и прерывания.

16.2.3 Арифметико-логическое устройство

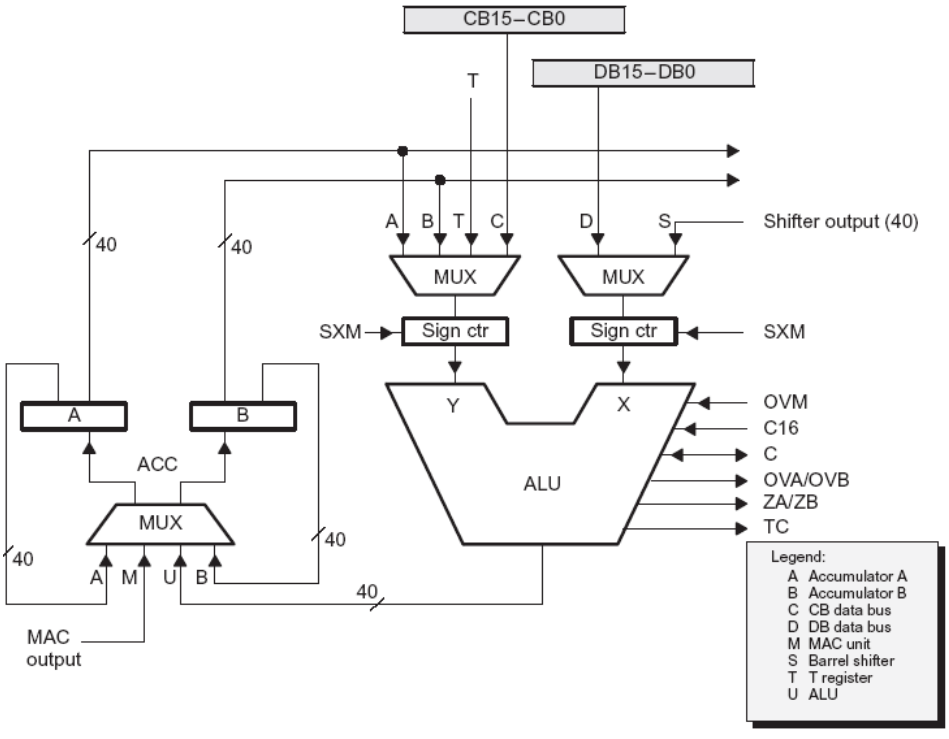


Рисунок 16–2 – Структура арифметико-логического устройства процессора DSP

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 ALU, результат обычно передается на аккумулятор назначения (аккумулятор A или B). 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 ALU - любая из трех величин:

- Значение одного из аккумуляторов (A или B).
- Операнд памяти данных из шины данных CB.
- Значение в регистре T.

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).

16.2.3.1 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 зависимоти 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 отрицательная 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.

16.2.3.2 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.

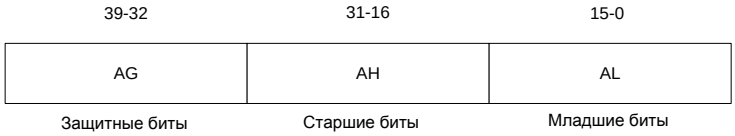
16.2.3.3 Парный 16-разрядный режим

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.

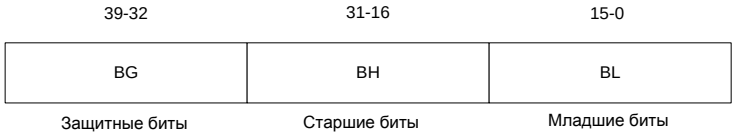
16.2.4 Аккумуляторы A и B

Аккумуляторы ACCA и ACCB, 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 A



Аккумулятор B

Рисунок 16–3 – 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.

16.2.4.1 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≤ SHIFT ≤ 15. Выполнение команды SFTA зависит от бита SXM. Когда SXM = 1 и сдвиг является отрицательной величиной, SFTA выполняет арифметический сдвиг вправо и поддерживает знак аккумулятора. Когда 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.

16.2.4.2 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.

16.2.4.3 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 B. В то же

самое время, она складывает операнды памяти Xmem и Ymem, сдвигает результат влево на 16 бит и загружает эту величину на аккумулятор A.

В инструкции LMS, аккумулятор B хранит временные результаты свертки входной последовательности и коэффициентов фильтра; аккумулятор A обновляет коэффициенты фильтра. Аккумулятор A 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 A (32-16) возводится в квадрат производится сложение с аккумулятором B. Результат сохраняется на аккумуляторе B. В то же самое время, Ymem вычитается из Xmem и разница сохраняется на аккумуляторе A. Значение, которое возводится в квадрат, есть значение аккумулятора перед вычитанием.

16.2.5 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.

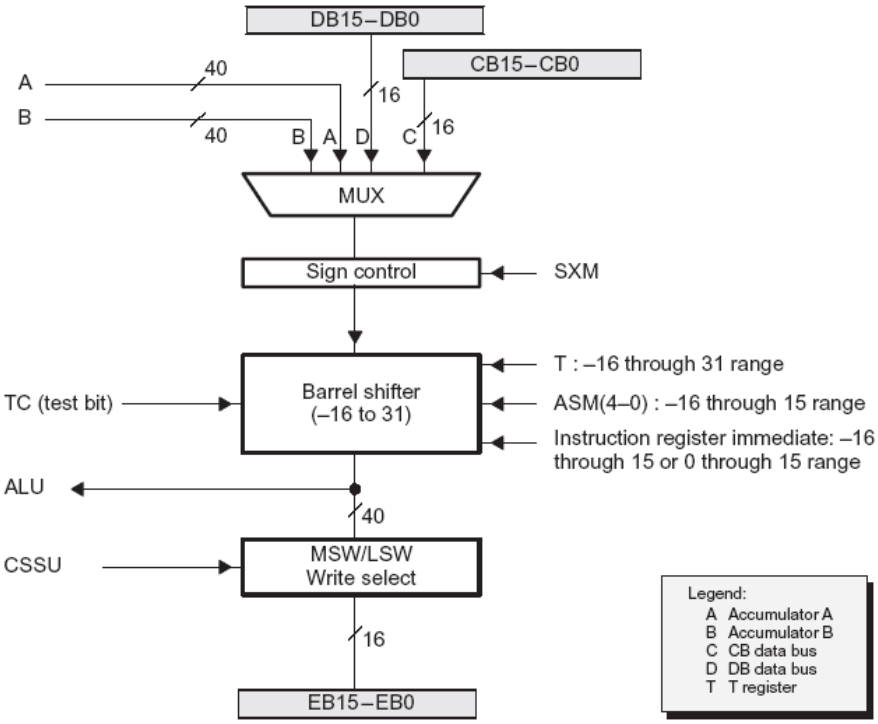


Рисунок 16–4 – Структурная схема сдвигового устройства

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 дипазоне 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)

16.2.6 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 свёртки, 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.

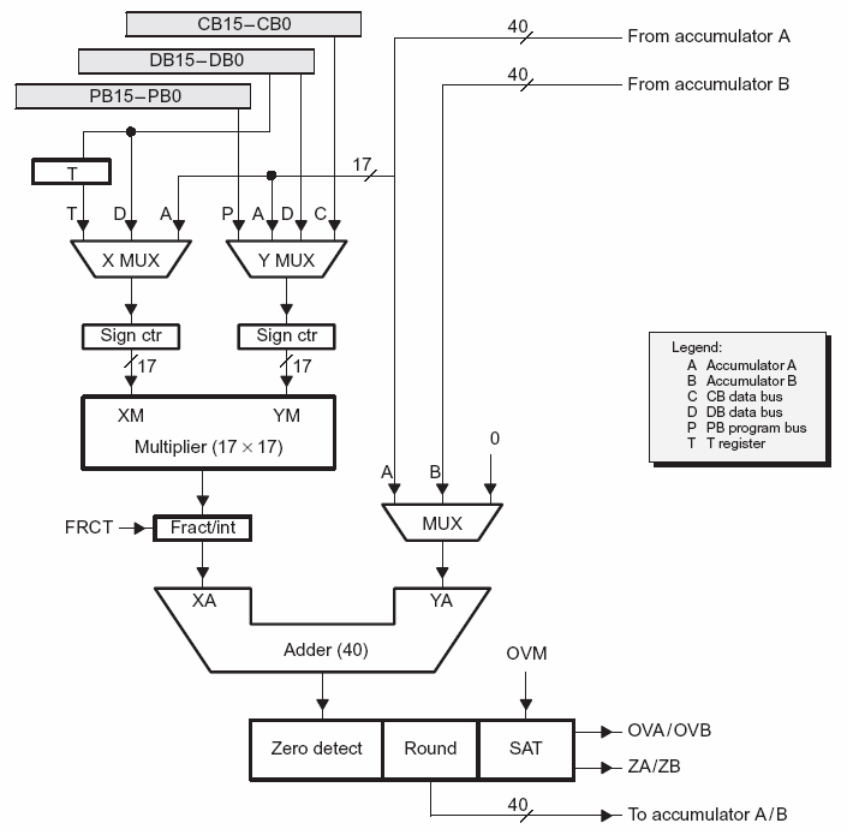


Рисунок 16–5 – 17 * 17-разрядный аппаратный умножитель

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.

16.2.6.1 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 16-1 показывает, как ввод данных в умножитель производится для некоторых инструкций. В общей сложности имеется девять используемых комбинаций ввода.

Таблица 16-1

		X Multiplexer			Y Multiplexer			
Case	Instruction Type	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, B		√				√	
7	MPYA *AR2		√					√
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.

16.2.6.2 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.

16.2.6.3 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 x 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)

16.2.7 Устройство сравнения, выбора и хранения (CSSU)

Устройство сравнения, выбора и хранения является специализированным аппаратным устройством, предназначенным для операций сложения/сравнения/выбора по Витерби.

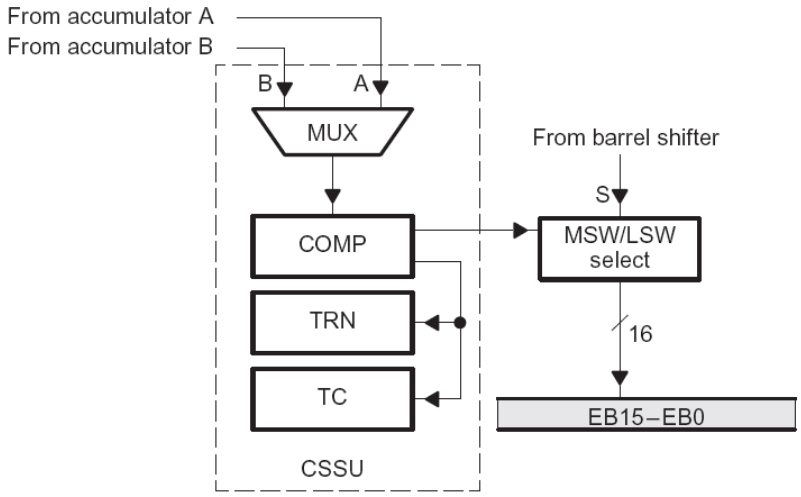


Рисунок 16–6 – Устройство сравнения, выбора и хранения (CSSU)

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 ± D1 и Met2 ± D2). Двойное сложение завершается в одном такте, если АЛУ сконфигурировано в двойном 16-разрядном режиме, установкой бита C16 в ST1. С АЛУ, сконфигурированном в двойном 16-разрядном режиме, все длинные слова (32-бит) становится двойными 16-битовыми арифметическими операндами.

Регистр T подключен к вводу АЛУ (как двойной 16-битовый операнд) и использован как локальная память для того, чтобы минимизировать обращения к памяти. Таблица 16-2 показывает инструкции, которые выполняют двойные 16-разрядные операции.

Таблица 16-2

Инструкции	Функции (двойной 16-разрядный режим)
DADD Lmem, src [, dst]	src(31–16) + Lmem(31–16) →□dst(39–16) src(15–0) + Lmem(15–0)→□dst(15–0)
DADST Lmem, dst	Lmem(31–16) + T →□dst(39–16) Lmem(15–0) – T →□dst(15–0)
DRSUB Lmem, src	Lmem(31–16) – src(31–16) →□src(39–16) Lmem(15–0) – src(15–0) →□src(15–0)
DSADT Lmem, dst	Lmem(31–16) – T→dst(39–16) Lmem(15–0) + T →□dst(15–0)
DSUB Lmem, src	src(31–16) – Lmem(31–16) →□src(39–16) src(15–0) – Lmem(15–0) →□src(15–0)
DSUBT Lmem, dst	Lmem(31–16) – T →□dst(39–16) Lmem(15–0) + T →□dst(15–0)

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, выполненную 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;
```

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.

16.2.8 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.

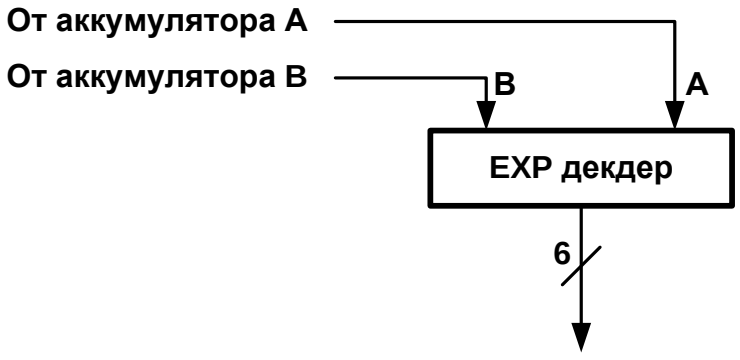


Рисунок 16–7 – Шифратор экспоненты

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 16–8 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 16–8 – Нормализация аккумулятора A

16.3 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 приёму 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 идёт 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 трёхсловная 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).

16.3.1 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 своём 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 2-х стадийной 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.

Этапы работы конвейера представлены ниже (Рисунок 16–9). 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.

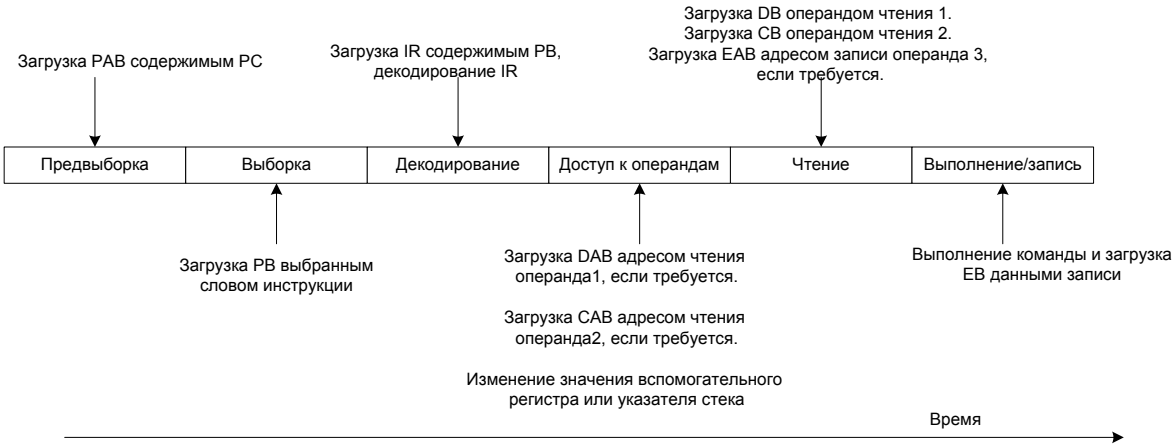


Рисунок 16–9 – Этапы работы конвейера

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 EAB. В следующем цикле операнд записывается в память, используя EB.

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 неё в этот адрес памяти.

Рисунок 16–10 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).

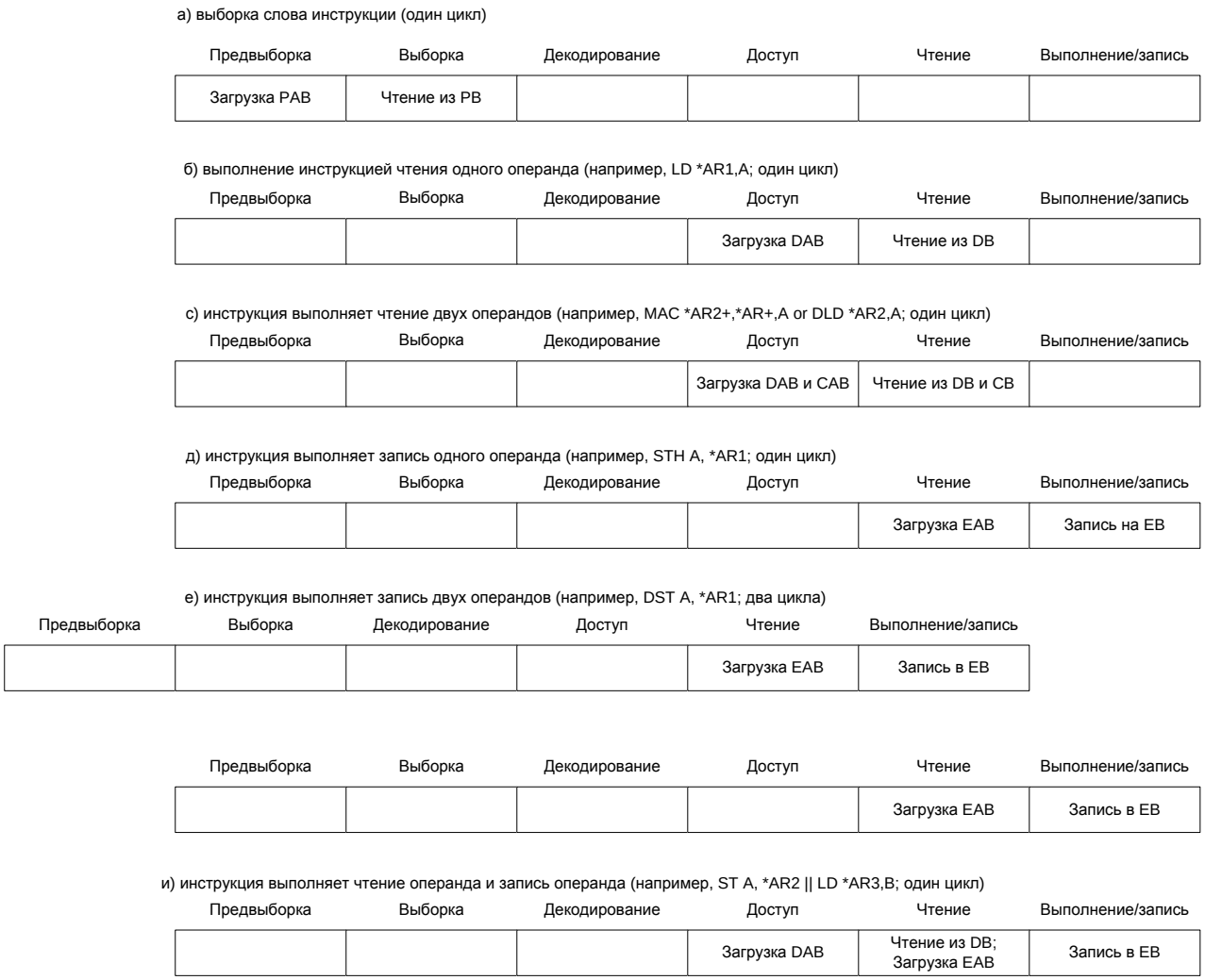


Рисунок 16–10 – Выполнение в ковейере различных фаз доступа к памяти

Далее приводятся примеры, которые демонстрируют, как конвейер отрабатывает различные типы инструкций. Все инструкции, показанные в примерах, считаются однословными и однотактными инструкциями (если не указано иное).

Конвейер изображен в этих примерах как набор чередующихся колонок, в которых каждая колонка соответствует одному слову инструкции, перемещающемуся по этапам конвейера.

Рисунок 16–11 – Работа конвейера

Каждая колонка в примере помечается слева как инструкция, операнд, многоцикловая инструкция, или очистка конвейера. Числа сверху представляют циклы инструкции. Некоторые циклы не показывают всех стадий конвейера, чтобы не загромождать рисунок.

Каждый блок в примере содержит наиболее важные действия, которые происходят на этом конвейерном этапе. Имя каждого конвейерного этапа показано выше блока, в котором это действие происходит.

Затенение представляет все выбранные инструкции и очистку конвейера, которая необходима для завершения инструкций, действия по которым показаны.

16.3.2 Инструкции перехода в конвейер

Следующие примеры показывают поведение конвейера в течение выполнения инструкции перехода (B) и задержанной инструкции перехода (BD), соответственно.

Поскольку инструкция перехода состоит из двух слов, она должна использовать, по крайней мере, два цикла, чтобы полностью выполниться. Тем не менее, стандартная инструкция перехода в действительности занимает четыре цикла.

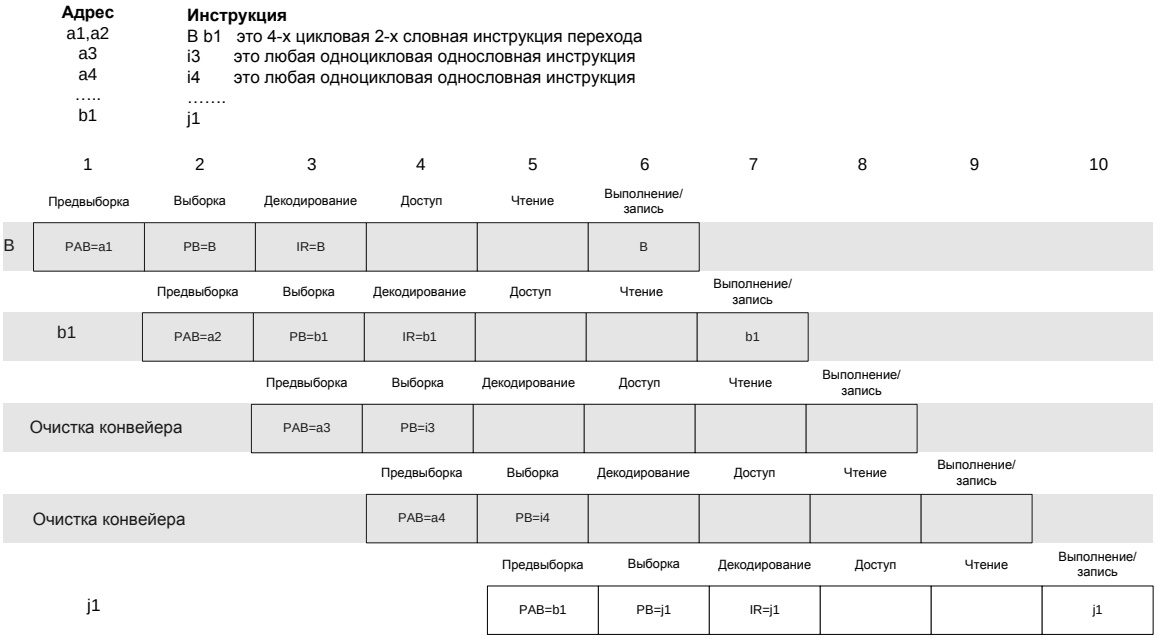


Рисунок 16–12 – Инструкция перехода в конвейере

- **Цикл 1:** PAB 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, PAB 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 4 цикла вместо двух.
- **Цикл 10: j1** завершает выполнение.

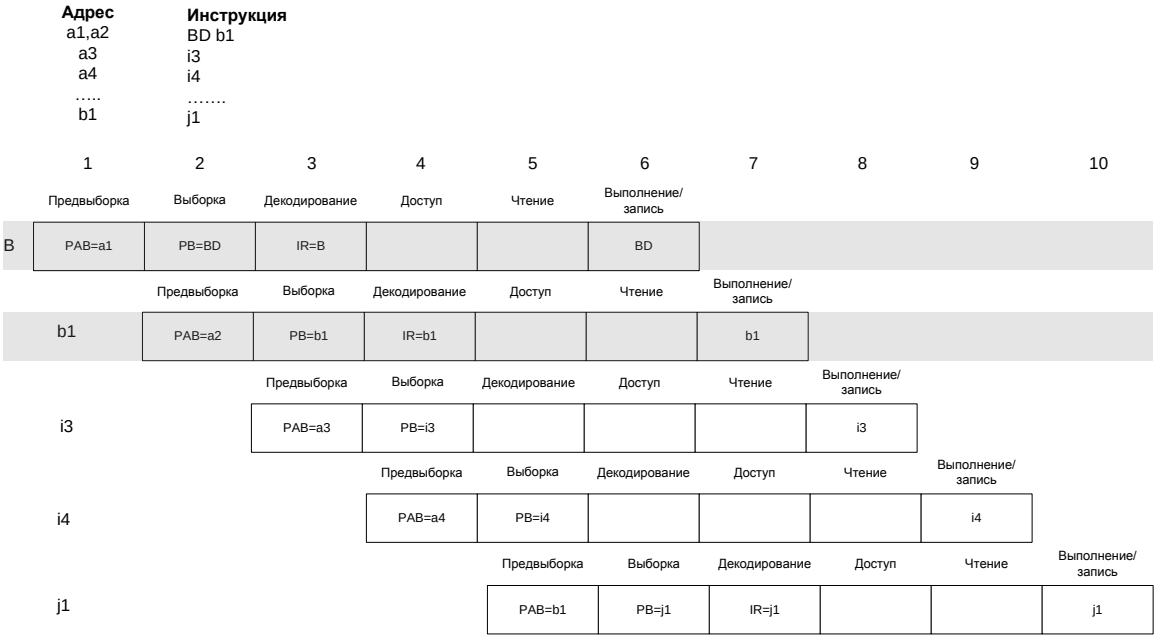


Рисунок 16–13 – Задержанная инструкция перехода в конвейере

В этом случае, конвейер ведется себя так же, 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 2-х цикловой.

16.3.3 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.

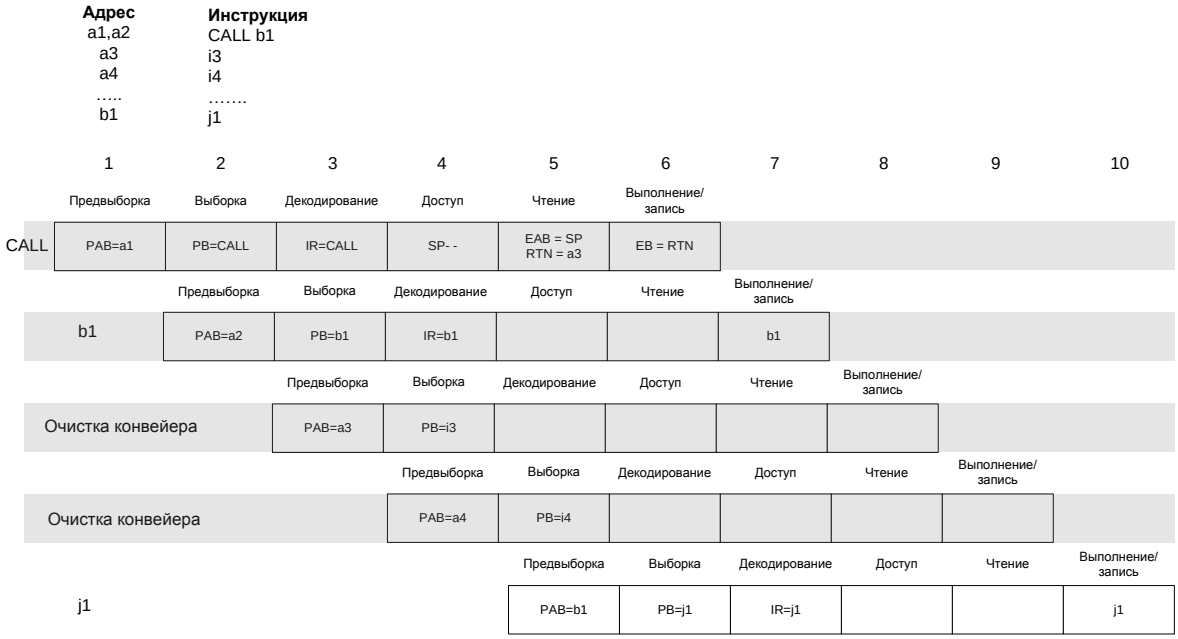


Рисунок 16–14 – Инструкция вызова в конвейере

- **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 циклов 6 и 7.
- **Циклы 8 и 9:** Эти циклы поглощены инструкцией вызова, поскольку выполнение следующих двух инструкций не разрешено.
- **Цикл 10:** j1 завершает выполнение.

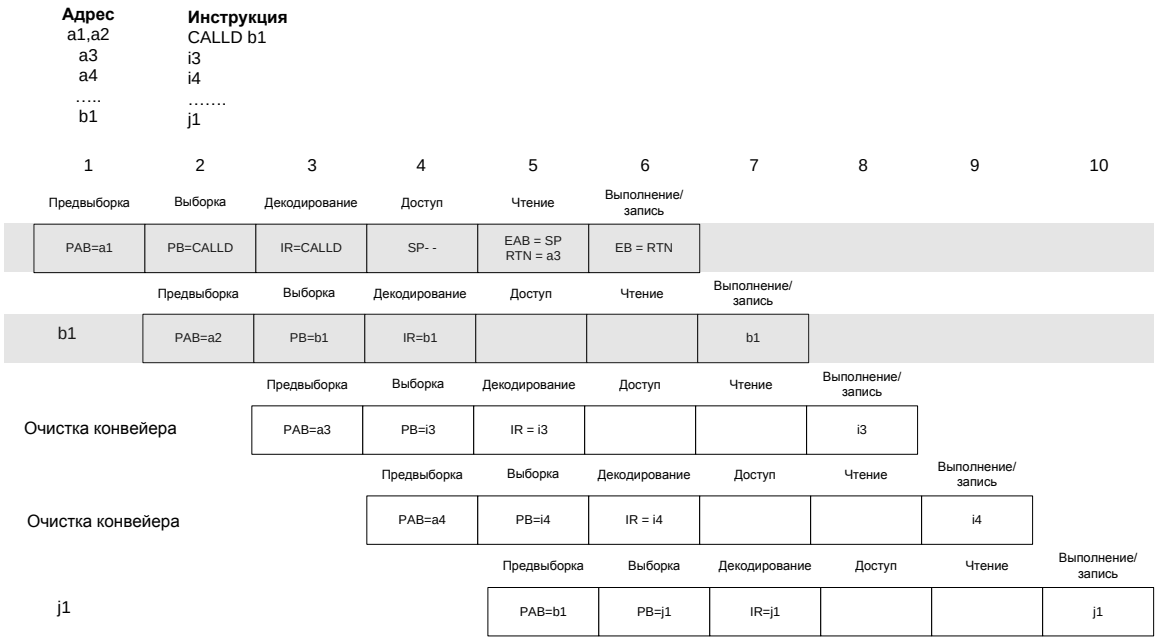


Рисунок 16–15 – Задержанная инструкция вызова в конвейере

В этом случае, конвейер ведется себя так же, как и обычная инструкция вызова. Тем не менее, в этом случае следующим двум инструкциям, i3 и i4, позволено завершить выполнение. Следовательно, только циклы 6 и 7 используются задержанной инструкцией перехода, делая задержанную инструкцию перехода 2-х цикловой.

Инструкция INTR ведется себя подобно инструкции вызова. Т.к. INTR однословная инструкция, она может вычислить векторный табличный адрес и предвыборку на один цикл раньше. Как показано на рисунке INTR использует для выполнения только три цикла.

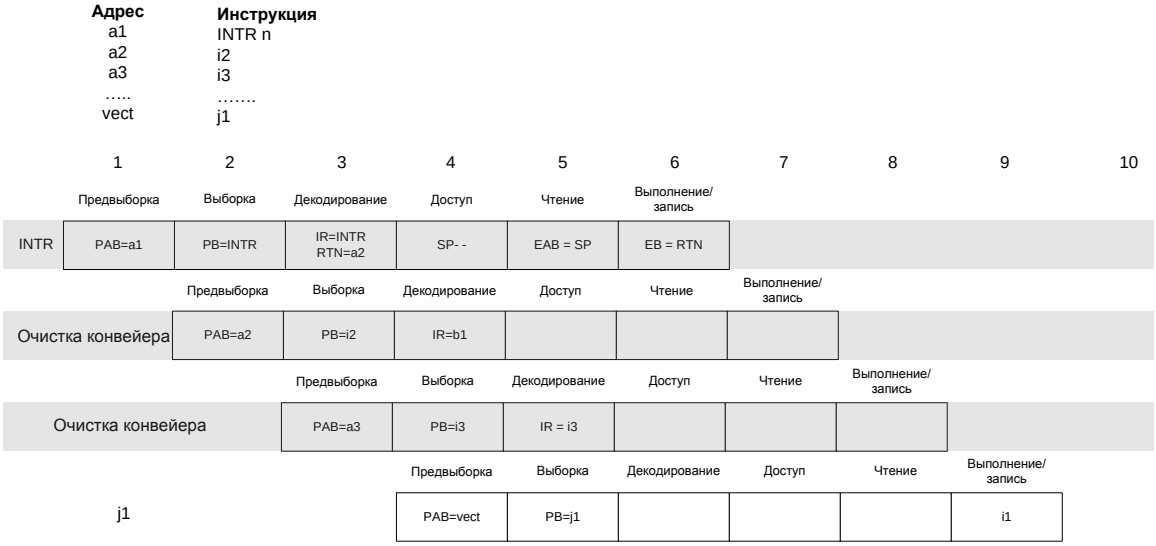


Рисунок 16–16 – Инструкция прерывания INTR в конвейере

16.3.4 Инструкции возврата в конвейере

Поскольку возврат является однословной инструкцией, Вы должны ожидать, что она использует минимум один цикл, чтобы полностью выполниться. В действительности, стандартная инструкция возврата использует пять циклов для выполнения.

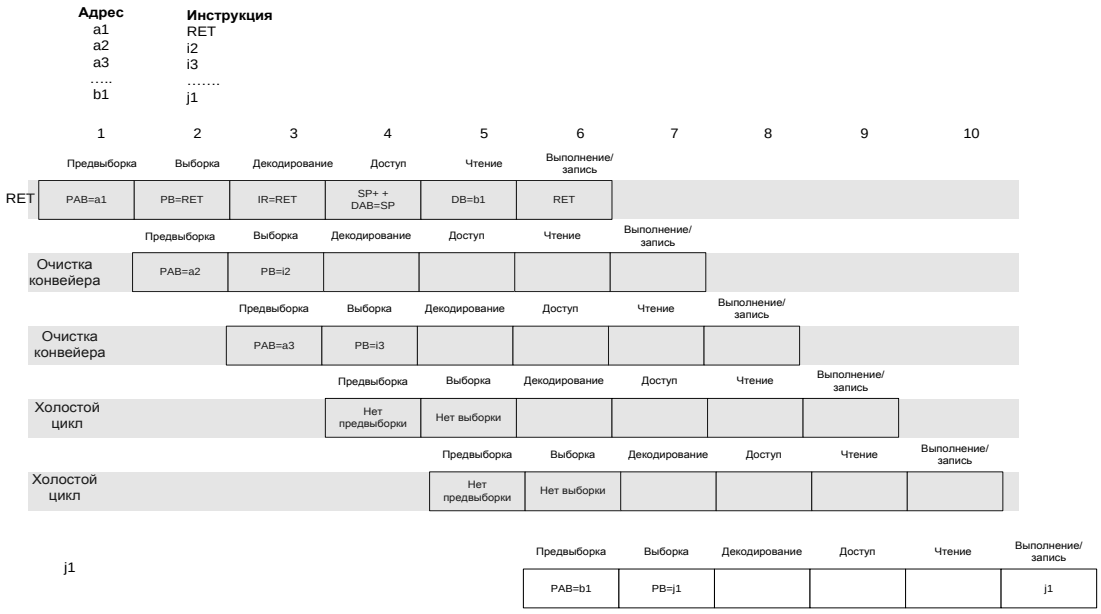


Рисунок 16–17 – Инструкция возврата в конвейере

- **Цикл 1:** PAB загружается адресом инструкции возврата.
- **Цикл 2:** код инструкции возврата выбран.
- **Цикл 3 и 4:** две дополнительные инструкции i2 и i3, выбраны. Хотя эти две инструкции и выбраны устройством, им не позволено переместиться далее на стадию декодирования и они отвергаются. В цикле 4 SP инкрементируется (представлено как SP ++) и DAB загружается содержимым SP для того, чтобы читать адрес возврата из стека.
- **Цикл 5:** верхушка стека прочитана с использованием DB.
- **Цикл 6:** инструкция возврата входит в этап конвейерного выполнения. Адрес, выбранный из стека, загружается на PAB. Это позволяет выбрать следующую инструкцию j1, по адресу возврата.
- **Цикл 7 и 8:** Эти циклы поглощены инструкцией возврата, поскольку следующие две инструкции, i3 и i4, не завершили своё выполнение.
- **Цикл 9 и 10:** Поскольку никакие инструкции не были выбраны в циклах 4 и 5, циклы 9 и 10 – холостые.
- **Цикл 11:** j1 завершает выполнение.

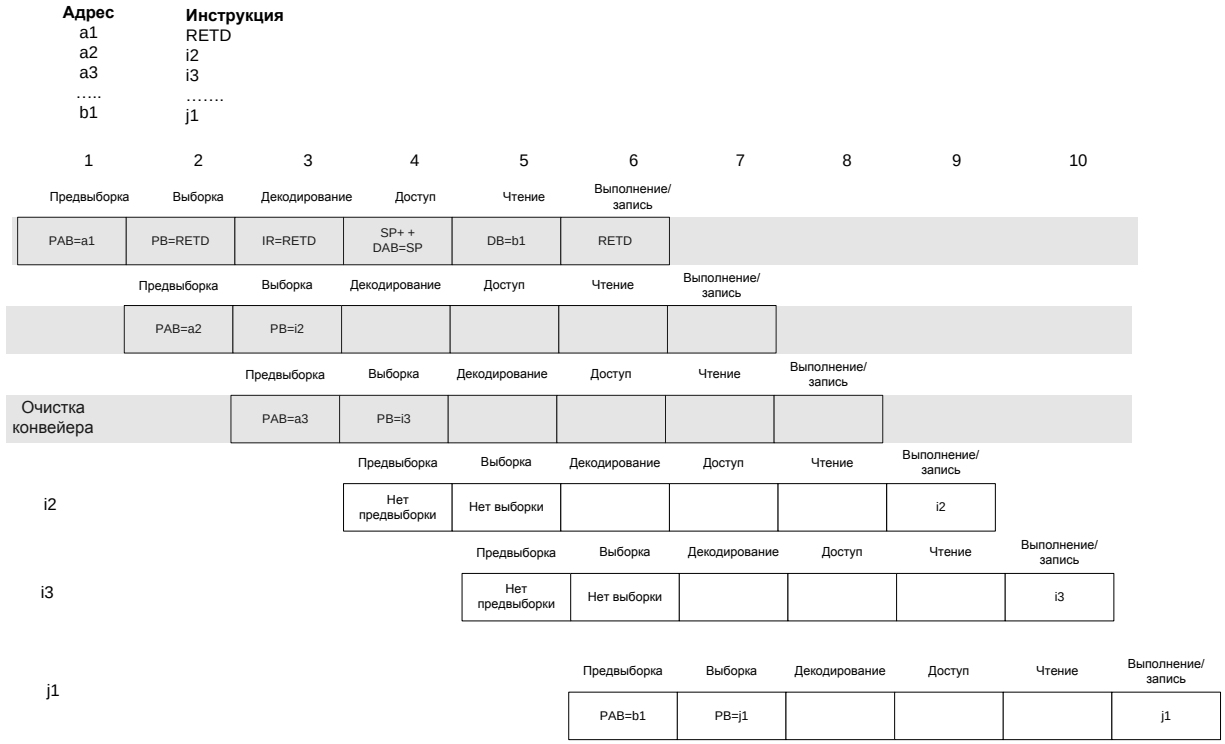


Рисунок 16–18 – Инструкция задержанного возврата в конвейере

Следующие примеры показывают конвейерное поведение для инструкции возврата с разрешением прерывания (RETE) и задержанной инструкции возврата с разрешением прерывания (RETED), соответственно. Конвейерное поведение для этих инструкций подобно тем самым обычным инструкциям возврата и задержанного возврата, соответственно, и эти инструкции требуют того же количества циклов для выполнения. Различие в том, что эти две инструкции разрешают прерывания глобально, восстанавливая бит INTM в течение выполнения этапов конвейера.

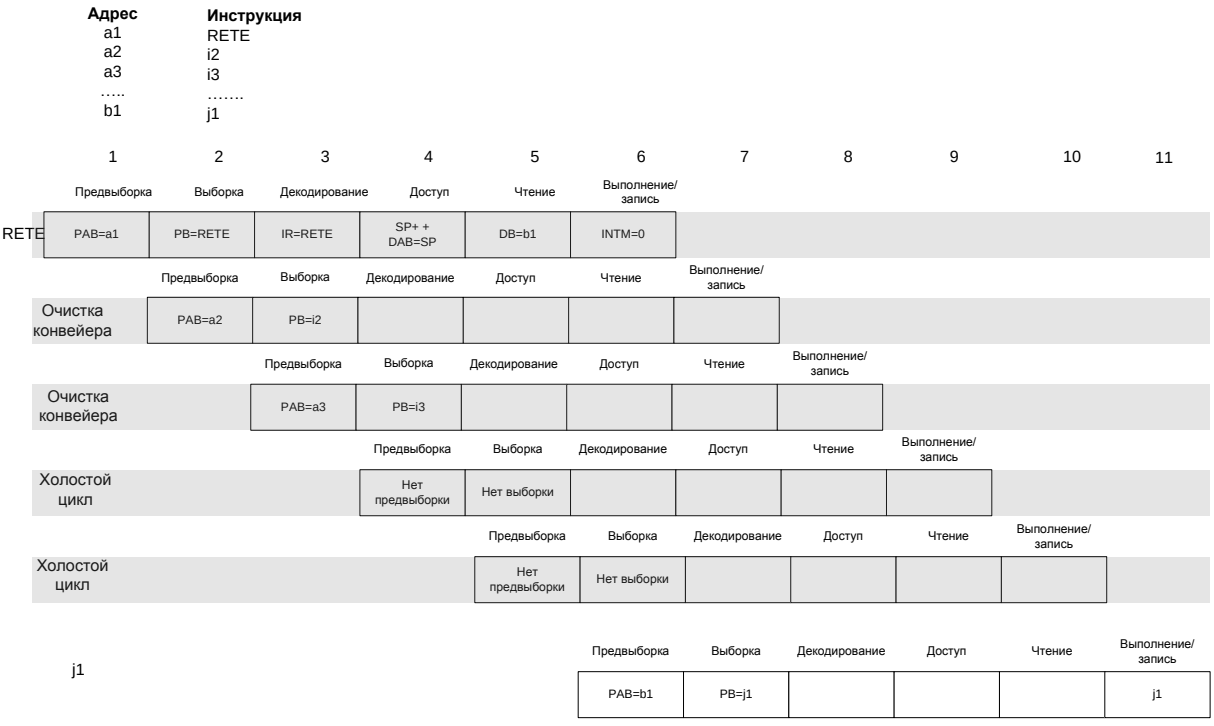


Рисунок 16–19 – Инструкция возврата с разрешением прерываний в конвейере

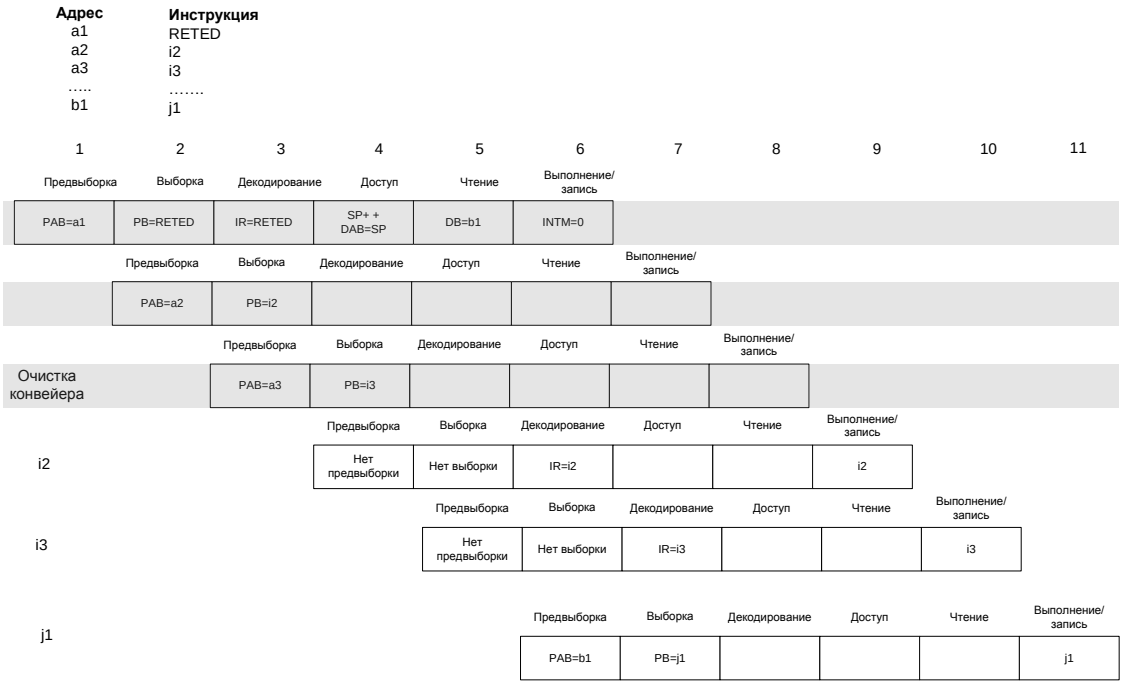


Рисунок 16–20 – Задержанная инструкция возврата с разрешением прерываний в конвейере

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.

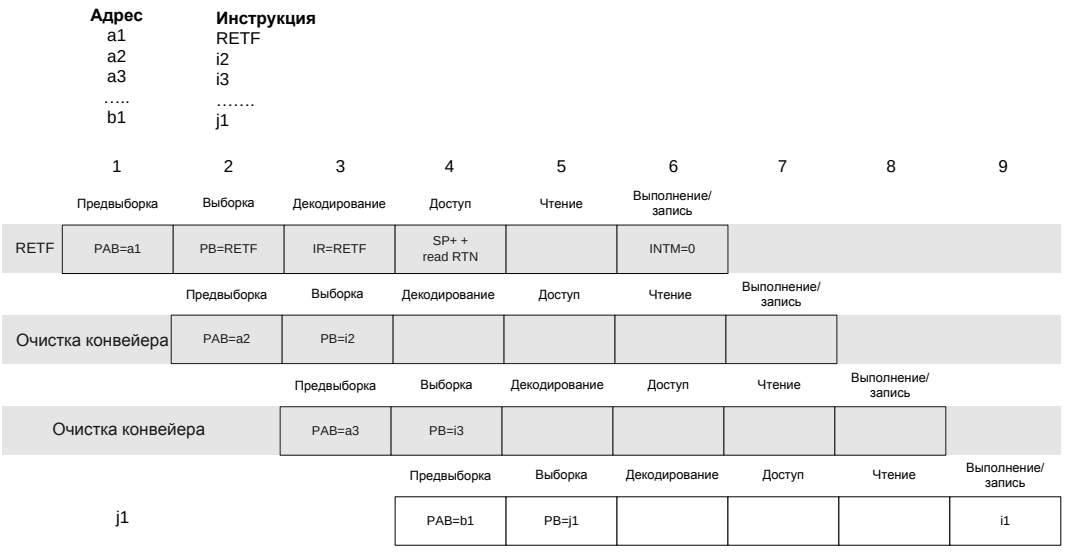


Рисунок 16–21 – Команда быстрого возврата в конвейере

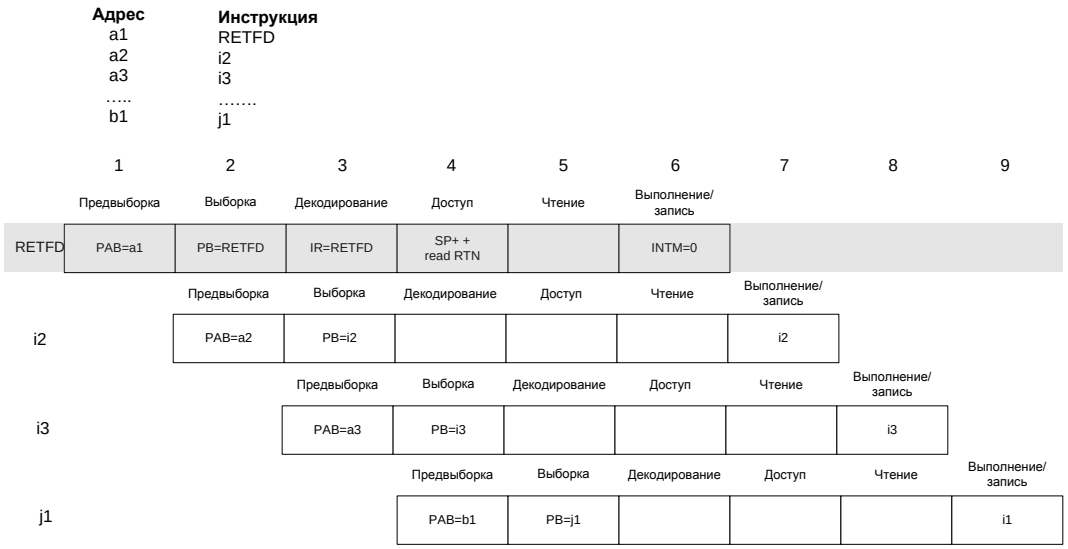


Рисунок 16–22 – Задержанная команда быстрого возврата в конвейере

16.3.5 Условная инструкция в конвейере

Поскольку XC – однословная инструкция, она использует минимально один цикл для выполнения. Пример показывает конвейерное поведение в течение выполнения XC.

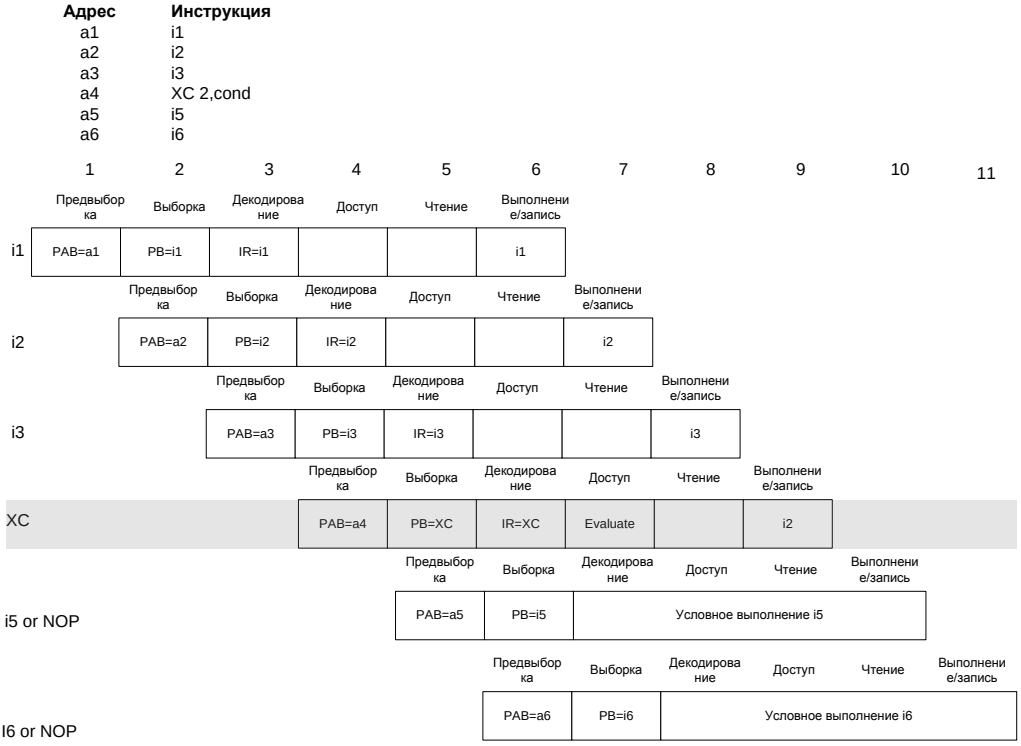


Рисунок 16–23 – Инструкция XC в конвейере

- **Цикл 4:** PAB загружается адресом инструкции XC.
- **Цикл 5:** инструкция XC выбрана.
- **Цикл 6:** инструкция декодирована.
- **Цикл 7**: когда инструкция XC перемещается на этап доступа к операндам в цикле 7 конвейера, любые условия определенные в инструкции XC 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 XC в одном цикле, ядро оценивает условия тестирования на этапе доступа. Это означает, что две однословные инструкции (или одна 2-х словная) до инструкции XC 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 XC.

16.3.6 Команды условного вызова и условного перехода в конвейере

Поскольку инструкция вызова состоит из двух слов, Вы должны ожидать, что потребуется минимум два цикла для её 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 CC, 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, PAB загружается адресом вызова (b1); в противном случае, загружается следующим инкрементированным адресом (a6).

Если оцененные условия истинны, i4 и i5 не выполняться в циклах 10 и 11. В этом случае, инструкция CC становится инструкцией 5-ти цикловой. Тем не менее, если оцененные условия ложные, команды i4 и i5 выполняться, делая инструкцию CC 3-х цикловой.

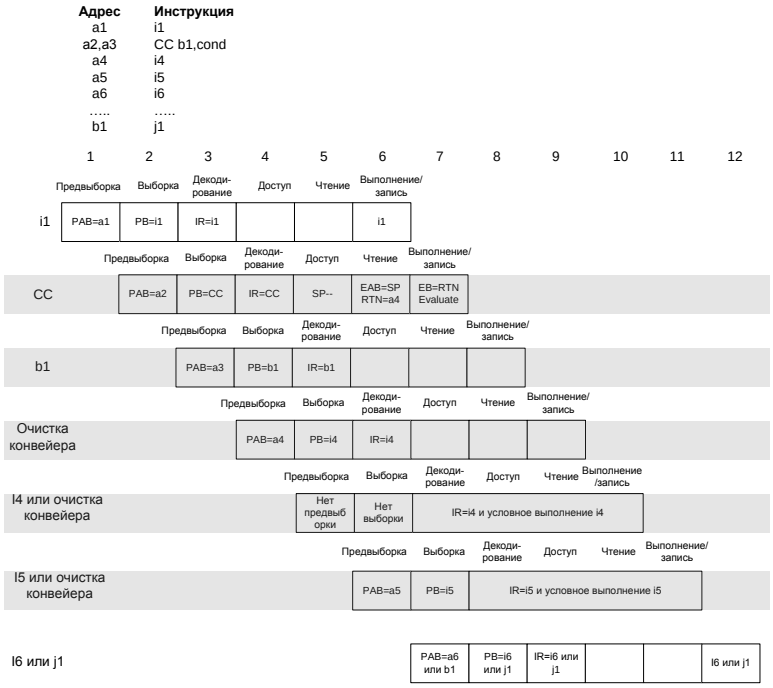


Рисунок 16–24 – Команда условного вызова (CC) в конвейере

Конвейер ведется себя так же 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 3-х цикловой.

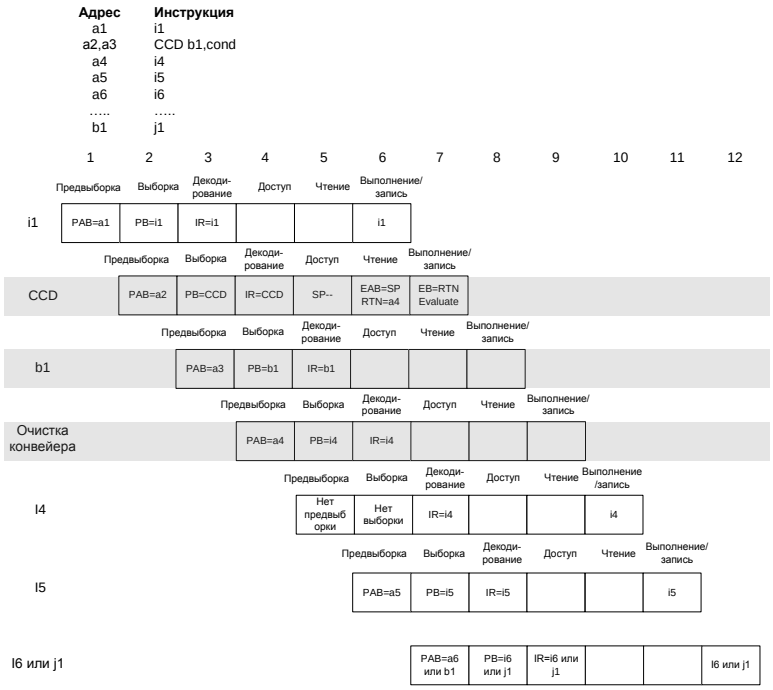


Рисунок 16–25 – Инструкция задержанного условного вызова функции (CCD) в конвейере

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.

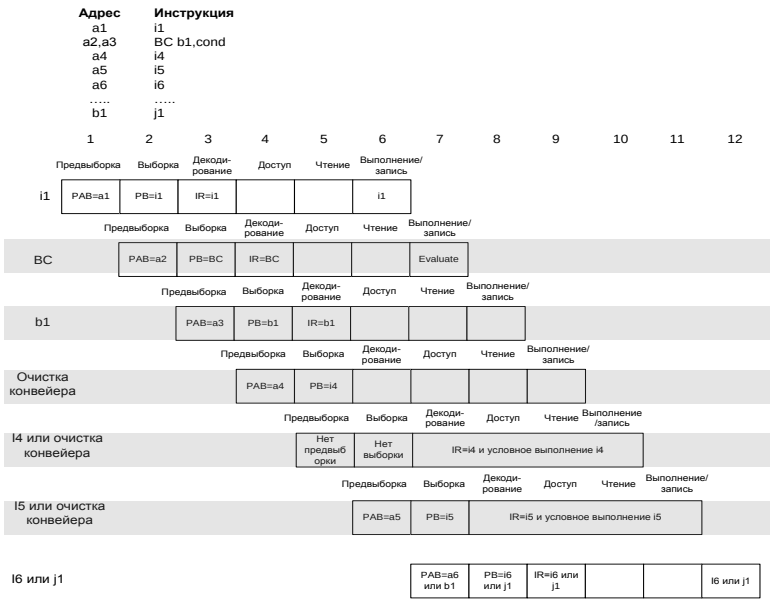


Рисунок 16–26 – Команда условного перехода (BC) в конвейере

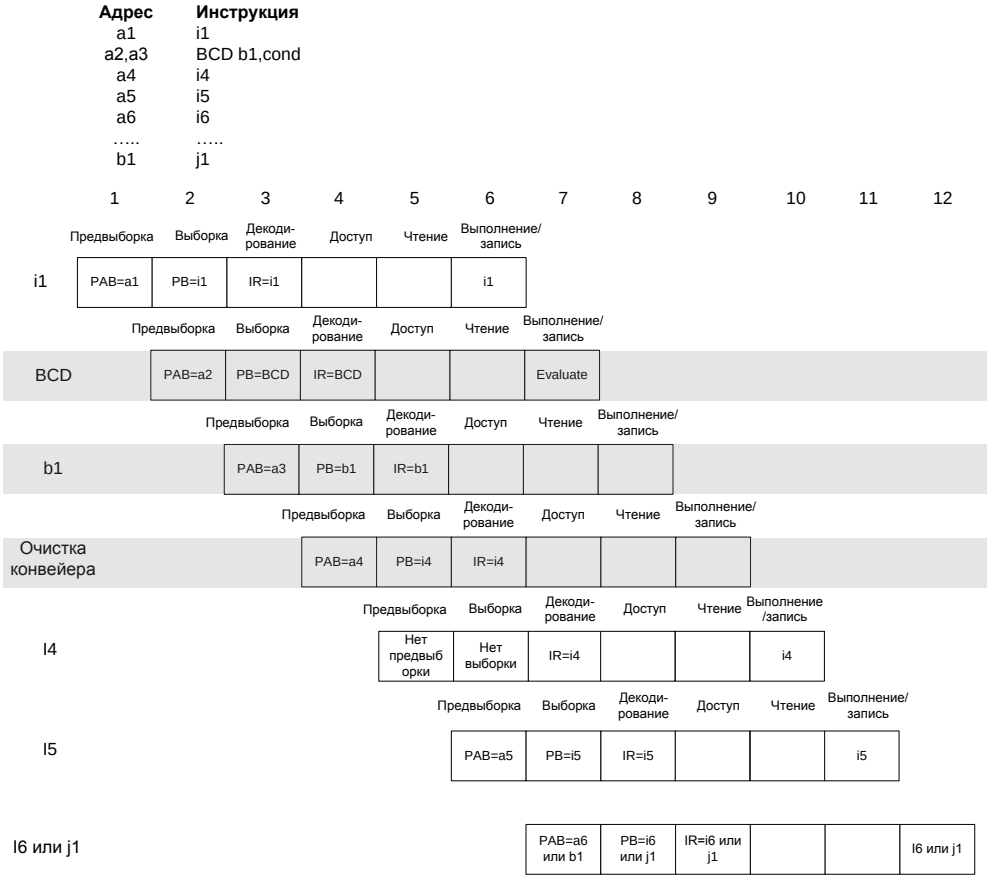


Рисунок 16–27 – Команда задержанного условного перехода (BCD) в конвейере

16.3.7 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.

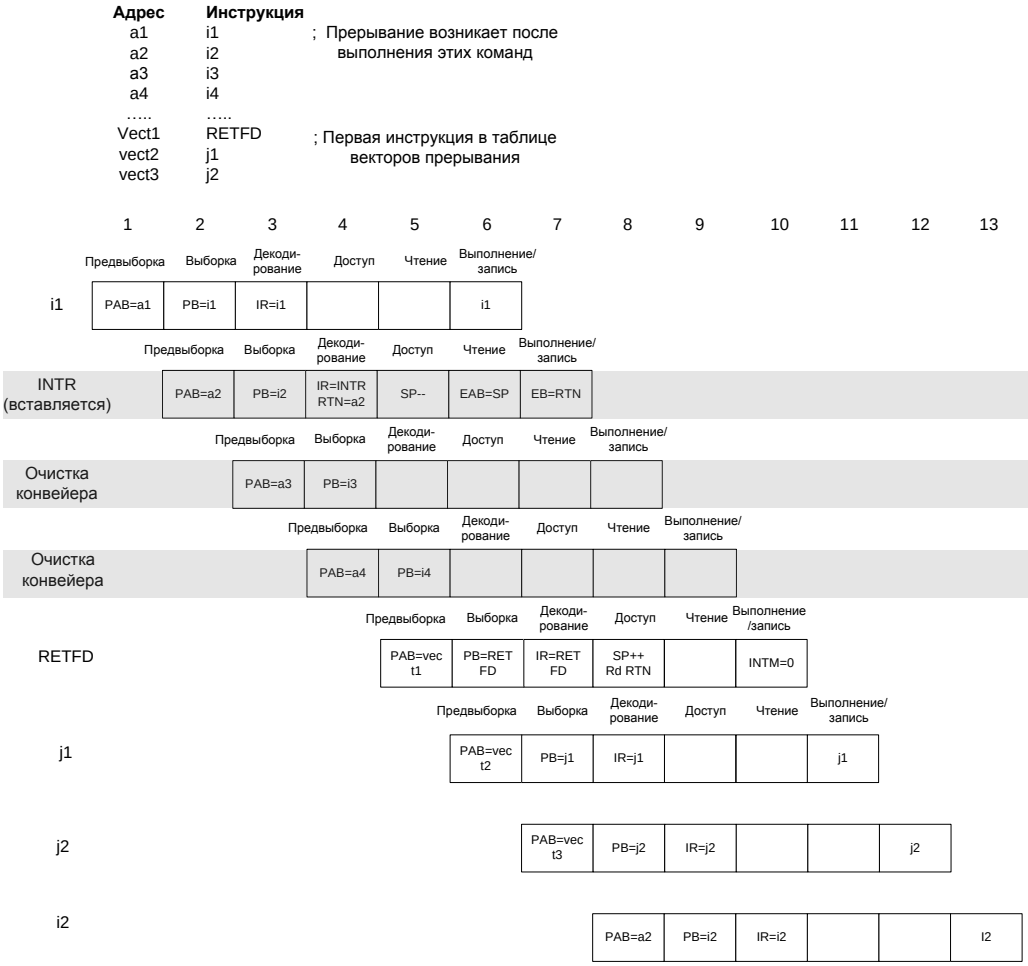


Рисунок 16–28 – Ответ конвейера на прерывание

16.4 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.

16.4.1 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 1-ом или 2-ом слове инструкции. 3-, 5-, 8- или 9-битовые значения закодированы в 1 слове команды; 16-разрядные значения закодированы во 2-ом слове команды. Длина непосредственного операнда, закодированного в команде, зависит от типа используемой команды. Таблица 16-3 даёт список команд, которые могут закодировать непосредственные операнды в своём слове команды. Таблица также дает число разрядов, которое может быть закодировано в команде.

Таблица 16-3 – 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 16–29 и Рисунок 16–30 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.

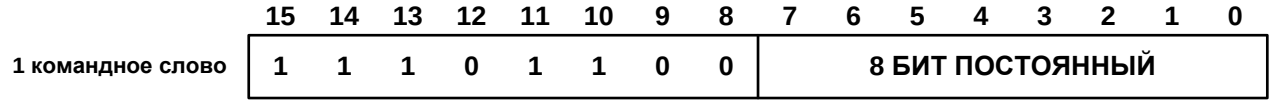


Рисунок 16–29 – Команда RPT с короткой непосредственной адресацией

Рисунок 16–30 – Команда RPT с 16-разрядной непосредственной адресацией

16.4.2 Абсолютная адресация

Это 4 типа абсолютной адресации:

- dmad адресация (data-memory address):
  - MVDK Smem, dmad
  - MVDM dmad, MMR
  - MVKD dmad, Smem
  - MVMD MMR, dmad

- pmad адресация (program-memory address):
  - FIRS Xmem, Ymem, pmad
  - MACD Smem, pmad, src
  - MACP Smem, pmad, src
  - MVDP Smem, pmad
  - MVPD pmad, Smem

- PA адресация (port address):
  - PORTR PA, Smem
  - PORTW Smem, PA

- *(lk) адресация.

dmad адресация

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, содержавшееся 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.

pmad адресация

Адресация к памяти программ (pmad-program-memory address) использует значение, определяющее адрес в пространстве программы. В синтаксисе обращения pmad использовуется 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.

PA адресация

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.

**(lk) адресация*

* (lk) адресация 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 трёхсловной. 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).

16.4.3 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 A 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 A. В режиме повторения, может использоваться приращение, чтобы увеличить адрес определённый значением аккумулятора A.

16.4.4 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 9-битовым 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 B, при условии, что правильный базовый адрес находится в DP (CPL = 0) или SP (CPL = 1), Вы написали бы:

```
ADD   SAMPLE, B
```

Младшие семь бит адреса SAMPLE сохранены в слове команды.

Рисунок 16–31 показывает формат кода операции для команд, использующих прямую адресацию.

Таблица 16-4 описывает биты команды прямой адресации.

Рисунок 16–32 иллюстрирует, как сформирован 16-разрядный адрес данных.

15-8	7	6 - 0
Код операции	**I = 0**	**dma**

Рисунок 16–31 – Формат команд прямой адресации

Таблица 16-4 – Краткое описание битов команды прямой адресации

Бит	Название	Функция
15 – 8	Код операции	Это 8-разрядное поле содержит код операции команды
7	I	I=0, режим адресации, используемый командой, является режимом прямой адресации
6 – 0	dma	Это 7-разрядное поле содержит адрес смещения памяти данных для команды

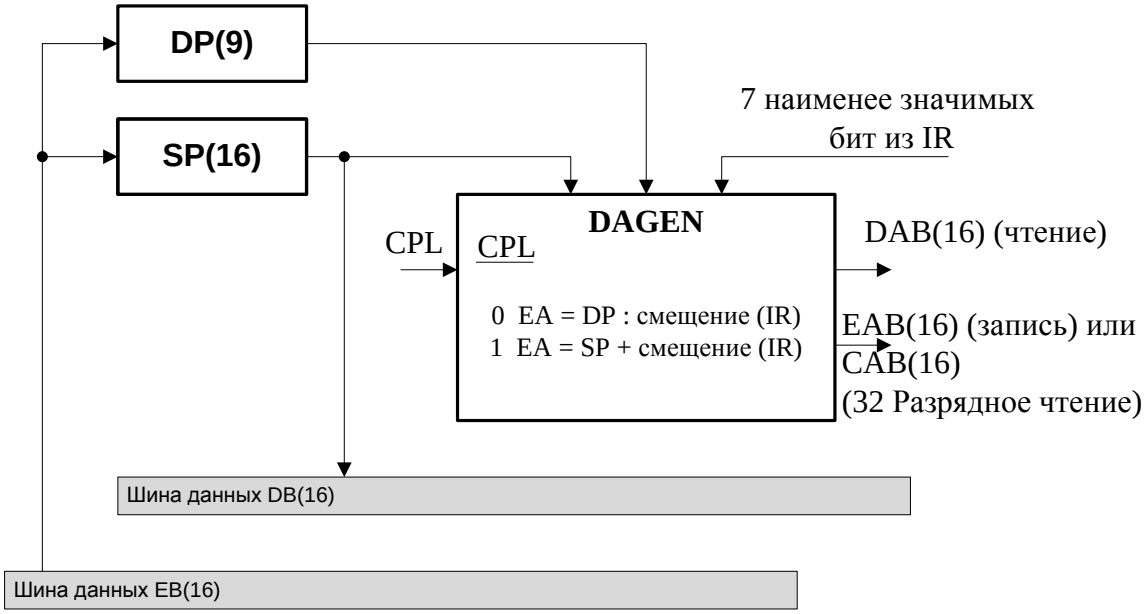


Рисунок 16–32 – Блок-схема прямой адресации

Прямая адресация со ссылкой на DP

В прямой адресации, ссылаемой на DP, 7-битовое поле dma в команде сочленяется (конкатенация) с 9-битовым значением в DP, чтобы сформировать адрес. Рисунок 16–33 показывает, как два значения составляют исполнительный адрес.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Значение из DP								**Значение из IR (dma)**							

Рисунок 16–33 – 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 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, 7-битовый dma в регистре команды складывается как положительное смещение со значением SP, чтобы сформировать эффективный 16-разрядный адрес памяти данных. Рисунок 16–34 показывает, как два значения объединяются, чтобы сформировать результирующий адрес.

Рисунок 16–34 – 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. раздел адресации через стек.

16.4.5 Косвенная Адресация

В косвенной адресации, к любому месту в пространстве данных в 64K-слов можно обратиться через 16-разрядный адрес, содержавшийся 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 16–35 показывает формат команды с косвенным обращением для единственного (Smem) операнда. Таблица 16-5 описывает биты команды.

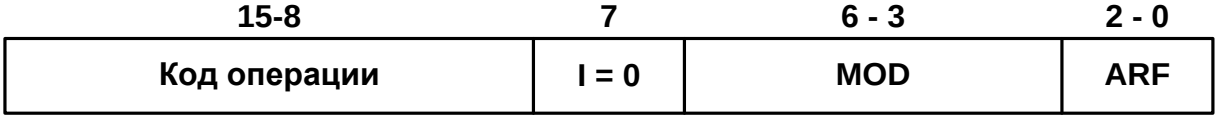


Рисунок 16–35 – Формат команд косвенной адресации для операнда памяти данных одиночного доступа

Таблица 16-5 – Косвенная адресация одного операнда

Биты	Название	Функции
15-8	Opcode	Это 8-ми битовое поле содержит код операции
7	I	При единичном значении используется косвенная адресация
6-3	MOD	Это 4-х 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 – Стандартная мода. В стандартной моде 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 устанавлен в нуль, когда DSP – в этой моде. CMPT = 1 – Мода совместимости. В моде 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 моде 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 16–36 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).

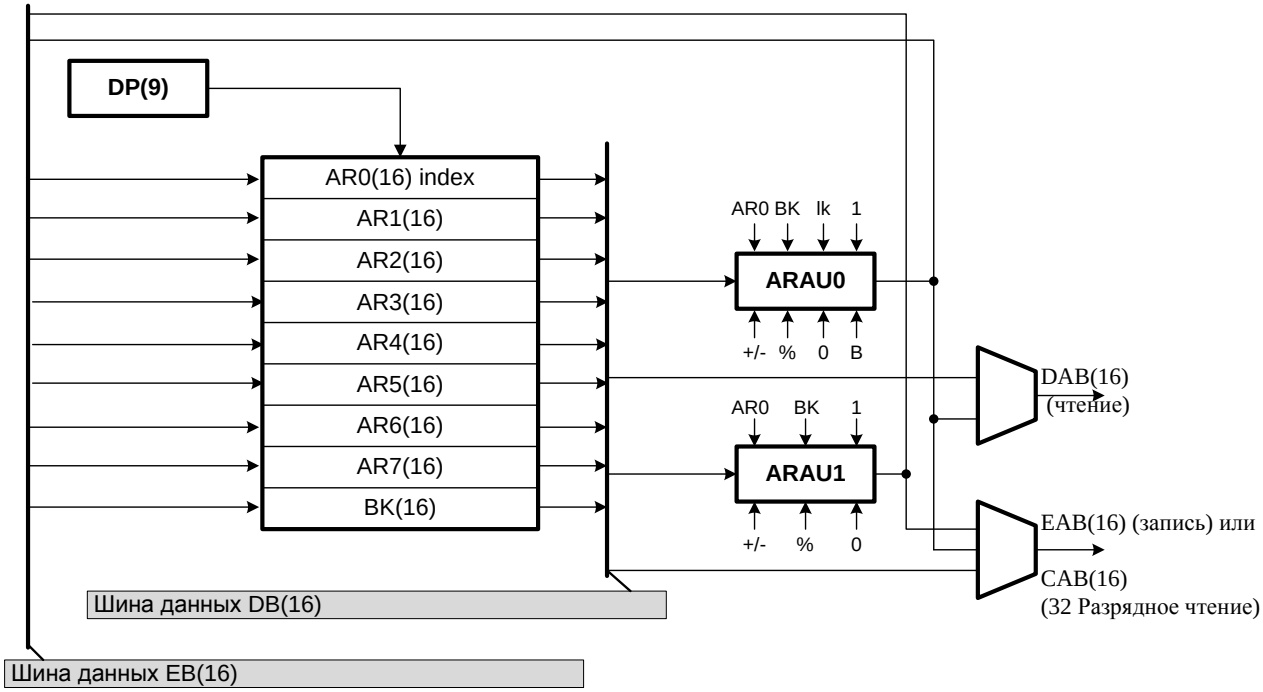


Рисунок 16–36 – Блок-схема косвенной адресации для операнда памяти данных одиночного доступа

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, 4-битовым полем в коде команды, использующем косвенную адресацию.

Таблица 16-6 дает список типов адресации одного операнда памяти данных, в соответствии со значением поля MD, синтаксис ассемблера и выполняемые действия для каждого типа.

Таблица 16-6 – Типы косвенной адресации

Поле MOD	Синтаксис операнда	Функция	Описание
0000 (0)	*ARx	addr = ARx	ARx содержит адрес памяти данных
0001 (1)	*ARx-	addr = ARx ARx = ARx -1	После доступа адрес в ARx декрементируется ‡
0010 (2)	*ARx+	addr = ARx ARx = ARx +1	После доступа адрес в ARx инкрементируется ‡
0011 (3)	*+ARx	addr = ARx + 1 ARx = ARx + 1	Адрес в ARx инкрементируется перед его использованием ‡§#
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 ‡
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 ‡
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.§
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.§
1111 (15)	*(lk)	addr = lk	Беззнаковое 16 битное длинное смещение используется как абсолютный адрес памяти данных.§¶

Примечания:

† ARx используется как адрес памяти данных, если не указано особо;

‡ Значение инкрементируемого и декрементируемого значения равно 1 для 16 битовых адресуемых слов и 2 для 32 битовых;

§ Эта мода невозможна для регистров, отображаемых в памяти;

¶ Эта мода подробнее обсуждается далее;
# Эта мода 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, инкрементируя или декрементируя 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 (BK) определяет размер кругового буфера. Круговой буфер размера R должен начаться 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 BK. Например, круговой буфер с 31 словами должен начаться в адресе, пять младших битов которого – 0 (то есть, XXXX XXXX XXX0 0000_2), и значение 31 должно быть загружено в BK. Как второй пример, круговой буфер с 32 словами должен начаться в адресе, шесть младших битов которого – 0 (то есть, XXXX XXXX XX00 0000_2), и значение 32 должно быть загружено в BK. 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 выбраного пользователем вспомогательного регистра (ARx). Конец адреса кругового буфера (EOB-end of buffer address) определяется заменой младших N бит ARx младшими N битами BK. Индекс кругового буфера – просто младшие биты N ARx, и шаг – количество, добавляемое или вычтаемое 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 ≤ index + step < BK:
        index = index + step.
Else
        if index + step ≥ 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 BK нулевое значение, циклической модификации адреса не происходит. Это особенно полезно, когда двойной операнд должен выполнить модификацию адреса, эквивалентную ARx+0.

0 иллюстрирует отношения между BK, вспомогательным регистром (ARx), началом кругового буфера, его вершиной и индекса в циклическом буфере.

0 показывает, как круговой буфер реализуется и каковы отношения между сгенерированными значениями и элементами в циклическом буфере.

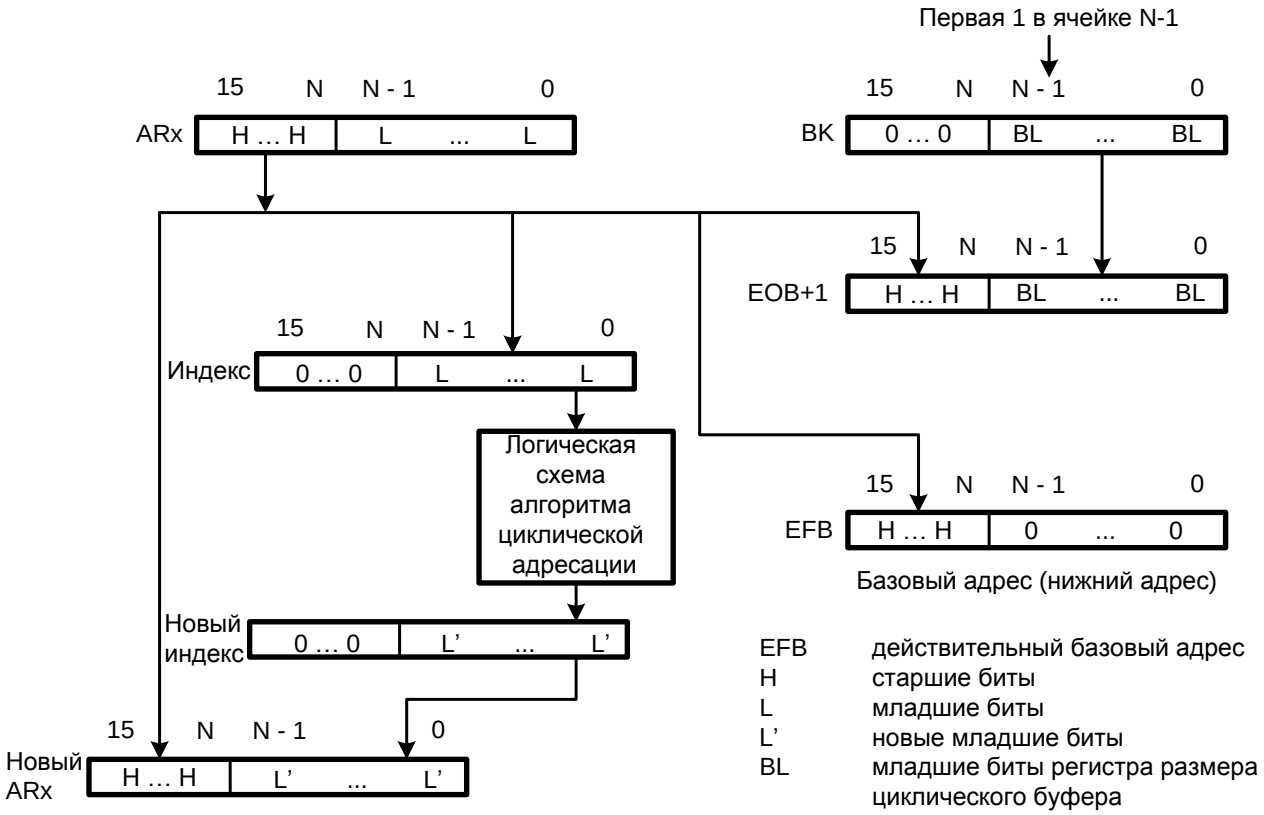


Рисунок 16–37 – Блок-схема циклической адресации

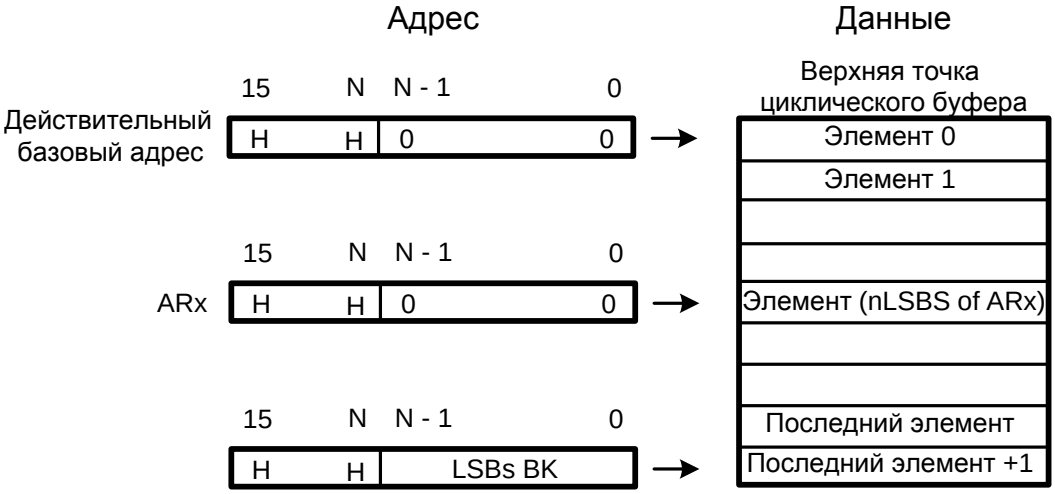


Рисунок 16–38 – Осуществление работы циклического буфера

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-реверсная 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, содержавшееся 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-реверсную адресацию, адрес генерируется бит-реверсным способом, с битом переноса, распространяющимся слева-направо, вместо нормального способа справа-налево. Синтаксис для каждого из двух режимов бит-реверсной 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 (01100000_2), и что AR0 содержит значение 00001000_2. 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 16-7 показывает соотношение битовой комбинации четырех младших бита AR2 в шаговой индексации и при использовании бит-реверсной адресации.

Таблица 16-7 – 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 16–39 показывает формат косвенной адресации команды с двойным операндом памяти данных.

Таблица 16-8 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 16-9 показывает, какие Xar и Yar выбираются в качестве вспомогательных регистров.

Рисунок 16–39 – Формат команды косвенной адресации для операндов памяти данных двойного доступа

Таблица 16-8 – Краткое описание битов команды адресации двойного операнда

Биты	Имя	Функция
15-8	Opcode	Это 8 разрядное поле содержит код операции команды
7-6	Xmod	Это 2-х разрядное 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 2-х разрядное поле определяет тип косвенной адресации для доступа к Ymem операнду
1-0	Yar	Двухразрядное поле определяет вспомогательный регистр, который содержит адрес Ymem

Таблица 16-9 – Выбор вспомогательных регистров

Xar или Yar	Вспомогательный регистр
00	AR2
01	AR3
10	AR4
11	AR5

Рисунок 16–40 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.

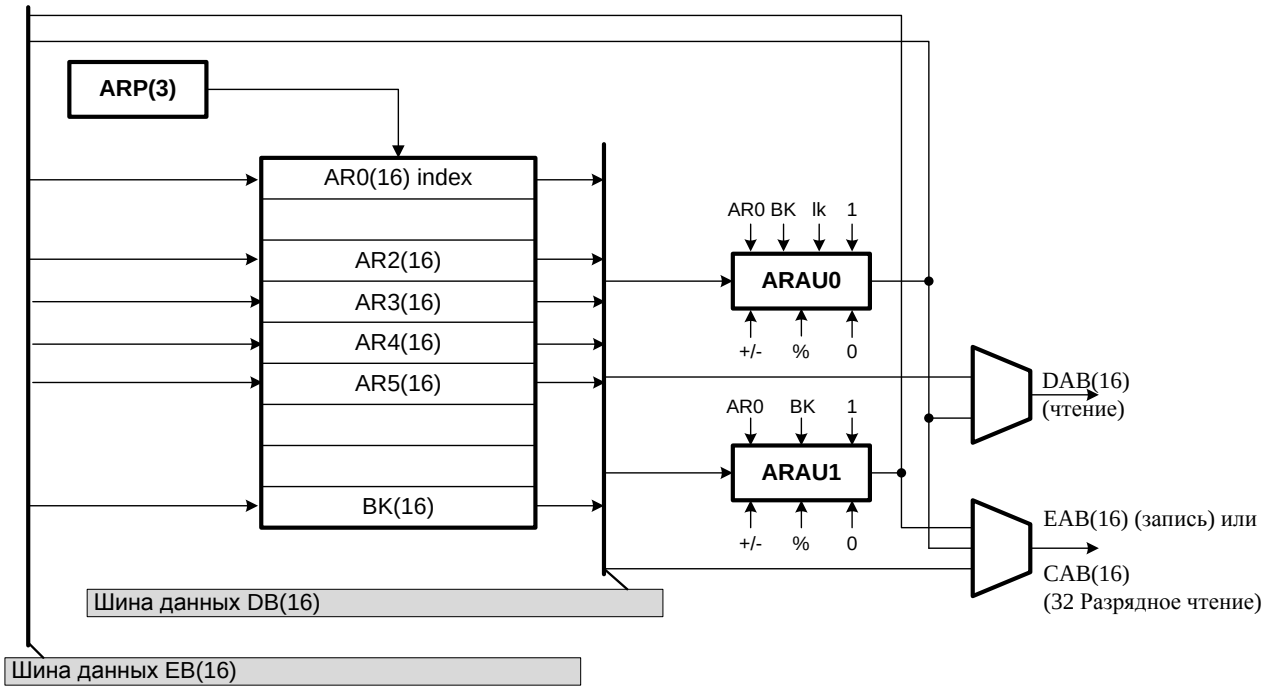


Рисунок 16–40 – Блок схема косвенной адресации для операндов памяти данных двойного доступа

Таблица 16-10 перечисляет типы двойной адресации операнда памяти данных, наряду со значением поля модификации (Xmod или Ymod), синтаксисом ассемблера и функцией для каждого типа.

Таблица 16-10 – Косвенная адресация с двойным операндом памяти данных

Xmod или Ymod	Синтаксис операнда	Функция	Описание†
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‡

†ARx используется как адрес памяти данных, если не указан другой способ действия

‡Размер циклического буфера определён в регистре размера BK

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 BK 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 осуществлён. Когда Xmod или Ymod = 2, ARx увеличивается после доступа.

Индексная адресация для двойного операнда (Xmod или Ymod = 3 и BK = 0)

Когда Xmod или Ymod = 3 и BK = 0, AR0 склаывается 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 BK =/ 0)

Когда Xmod или Ymod = 3 и BK=/ 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 выполниться в одном цикле:

- BIT Xmem, BITC
- SACCD src, Xmem, cond
- SRCCD Xmem, cond
- STRCD Xmem, cond

Пять инструкций с дополнительным сдвигом также поддерживают этот тип адресации для одного операнда и выполняются в одном цикле.

- ADD Xmem, SHFT, src
- LD Xmem, SHFT, dst
- STH src, SHFT, Xmem
- STL src, SHFT, Xmem
- SUB Xmem, SHFT, src

Режим совместимости (аналоги - TMS320C2x/C20x/C24x/C5x) (ARP)

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 16–41 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 16-11 показывает синтаксис ассемблера для ’C2x/C20x/C24x/C5x сравнивая его с данным микропроцессором.

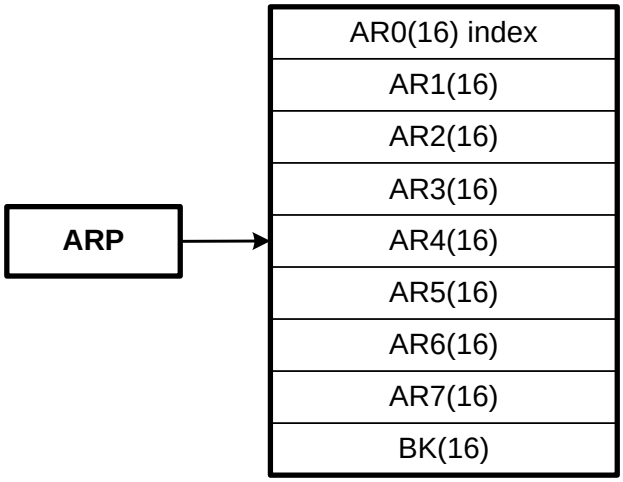


Рисунок 16–41 – Формирование индексов вспомогательных регистров протоколом разрешения адреса (ARP протоколом)

Таблица 16-11 – Сравнение синтаксисов ассемблерадля устройств TMS320C2x/C20x/C24x/C5x и ‘54x

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

Рисунок 16–42 показывает формат команды с косвенным обращением для ARP режима.

15-8	7	6 - 3	2 - 0
Код операции	**I = 1**	**MOD**	**ARF=000**

Рисунок 16–42 – Формат команды косвенной адресации в режиме сравнения

Таблица 16-12 описывает биты команды в ARP режиме.

Таблица 16-12 – Инструкции с косвенной адресацией - совместимая мода

Биты	Имя	Функция
15-8	Opcode	Это 8-ми битовое поле содержит код операции
7	I	При единичном значении используется косвенная адресация
6-3	MOD	Это 4-х битовое 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

Биты	Имя	Функция
		регистре 1. CMPT=0 – Стандартная мода. В стандартной моде 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 устанавлен в нуль, когда DSP в этой моде. CMPT=1 – Мода совместимости. В моде 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 моде совместимости

Примечания:

ARP должен всегда быть установлен в 0, когда DSP – в стандартной моде (CMPT = 0).

При сбросе как ARP так и CMPT устанавливаются в 0 принудительно.

16.4.6 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 16–43 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.

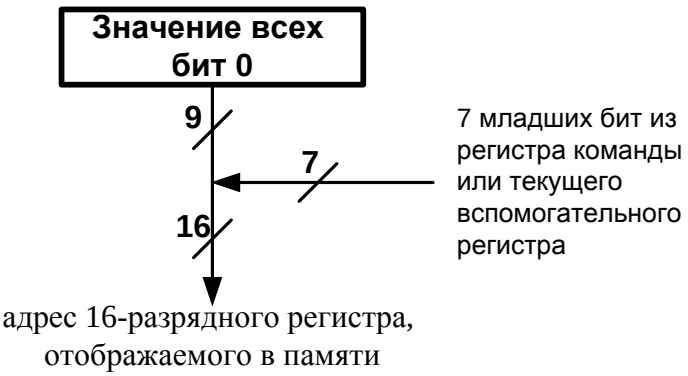


Рисунок 16–43 – Блок-схема адресации отображаемого в памяти регистра

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 моды косвенной адресации недоступны для адресации регистров отображенных в памяти:

- *ARx(lk)
- *+ARx(lk)
- *+ARx(lk)%
- *(lk)

В этих случаях ассемблер выдаёт предупреждение.

16.4.7 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 16–44 показывает пример состояния стека и SP до и после помещения X2 в стек (PSHD X2).

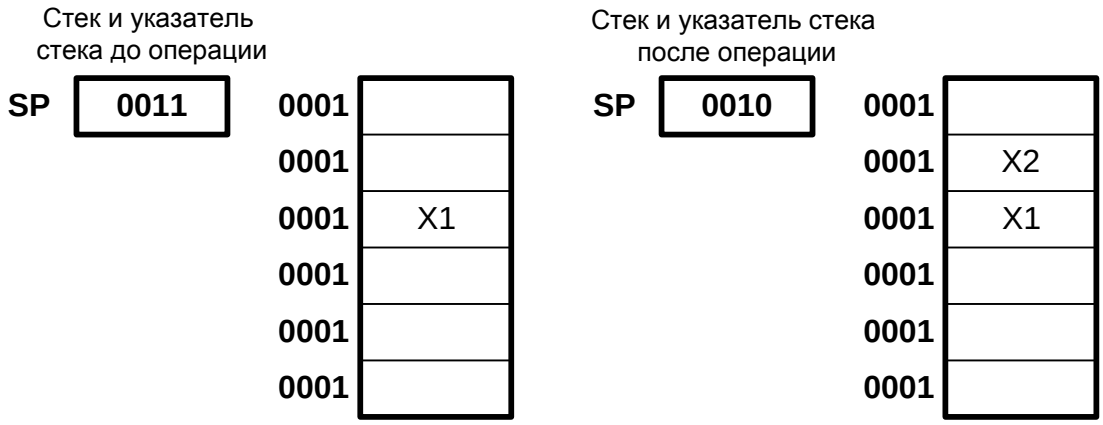


Рисунок 16–44 – Стек и указатель стека до и после операции записи в стек

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 подпрограмма 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).

16.4.8 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-битовые данным, тем не менее, требует использование специальных инструкций, которые указаны в Таблица 16-13.

Таблица 16-13 – 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 нечётному адресу(большему на 1). Если сначала доступ по нечетному адресу, то затем доступ ко второму слову будет по меньшему чётному адресу. Рисунок 16–45 показывает этот эффект.

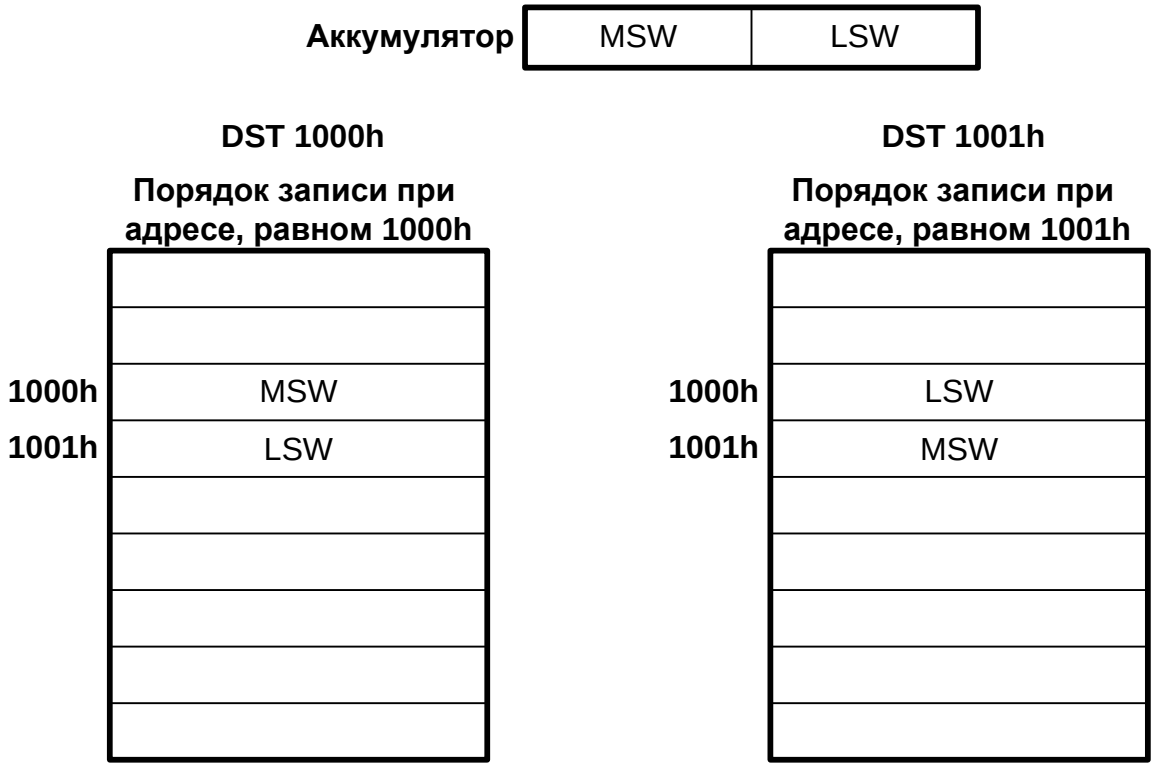


Рисунок 16–45 – Порядок слов в памяти

16.5 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 счётчиком микропроцессора.

Мода 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 (PC) 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 DSP средствами 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.

16.5.1 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 16–46):

- 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).

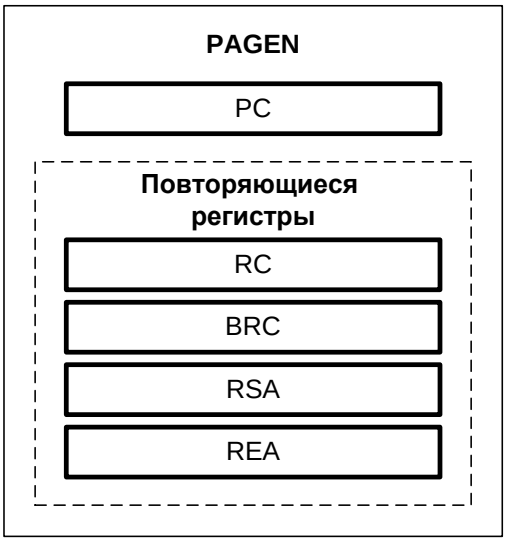


Рисунок 16–46 – Регистры логической схемы генерации адреса программ

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 неё, 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.

16.5.2 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.

PC может быть установлен несколькими способами. Таблица 16-14 показывает, что загрузка в PC зависит от того, какое действие выполняется.

Таблица 16-14 – Загрузка адреса в PC

Операция	Адрес, загружаемый в PC
Сброс	PC загружается кодом FF80h
Последовательное выполнение	PC загружается значением PC + 1
Переход	PC загружается 16-битным непосредственным значением следующим за инструкцией
Переход из аккумулятора	PC загружается младшими 16-ти битами аккумулятора A или B
Цикл повторения блока	PC загружается начальным адресом повторения(RSA), когда PC+1 эквивалентно конечному адресу повторения (REA), при условии того, что BRAF = 1
Вызов подпрограммы	PC+2 заталкивается в стек, PC загружается 16 битным операндом, непосредственно следующим после инструкции. Инструкция возврата выталкивает вершину стека обратно на PC для возврата к прерванной последовательности.
Вызов подпрограммы из аккумулятора	PC+1 вталкивается в стек, PC загружается младшими 16 битами аккумулятора A или B. Инструкция возврата выталкивает вершину стека обратно на PC для возврата к прерванной последовательности.
Аппаратное прерывание, программное прерывание или ловушка	PC+1 вталкивается в стек, PC загружается адресом соответствующим вектору ловушки. Инструкция возврата выталкивает вершину стека обратно на PC для возврата к прерванной последовательности.

16.5.3 Ветвления

Ветвления ломают последовательный поток инструкций, передавая управление в другую позицию в программной памяти. Следовательно, ветвления влияют на программный адрес сгенерированный и сохраненный в PC. 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, PC загружается определённым адресом программной памяти, и выполнение новой секции кода начинается с этого адреса. Адресом, загружаемым в PC, является второе слово инструкции перехода или младшие 16 бит аккумулятора (аккумулятор A или аккумулятор B).

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 2-х словная 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 PC (переход, вызов подпрограммы, возврат или программное прерывание).

Таблица 16-15 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 16-15 – Безусловный переход

Команда	Описание	Число циклов Не задерж./задерж.
B[D]	Загрузка PC адресом, определенным инструкцией	4/2
BACC[D]	Загрузка PC 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 16-16. Если все условия выполнены, PC 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 передаётся 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 PC (переход, вызов подпрограммы, возврат или программное прерывание).

Таблица 16-16 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 16-16 – Команды условного перехода

Инструкция	**Описание**	**Количество циклов Условие выполнено/не выполнено**	
		Не Задержанная	**Задержанная**
BC[D]	Загрузка PC адресом, определенным инструкцией, если условие, определённое 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 16-17 – Команды длинных переходов

Команды	**Описание**	**Количество циклов**
		Не задержанная / Задержанная
FB[D]	Загрузка PC и XPC адресом, определённым в инструкции	4/2
FBACC[D]	Загрузка PC и XPC адресом, определённым в 23 разрядах соответствующего аккумулятора	6/4

16.5.4 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 её 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 определённым 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 2-словная или две 1-словные 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 PC (переход, вызов процедуры, возврат или программное прерывание).

Таблица 16-18 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 16-18 – Команды безусловного перехода

Команда	Описание	Количество циклов выполнения. Не задерж./задерж.
CALL[D]	В стеке размещается адрес возврата и затем в PC загружается адрес, определённый в инструкции	4/2
CALA[D]	В стеке размещается адрес возврата и затем в PC загружается адрес, определённый 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 16-19. Если все условия выполнены, PC 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 PC, 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 передаётся в начало вызванной функции. Если условия не выполнены, эти две инструкции выполняются вместо вызова.

- Задержанный: одна 2-словная или две 1-словные 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 PC (переход, вызов из подпрограммы, возврат или программное прерывание).

Таблица 16-19 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 16-19 – Условные вызовы

Команда	Описание	Количество циклов	
		Не задержанная	Задержанная
CC[D]	Размещает адрес возврата в стеке и затем загружает в PC адрес определённый в инструкции, если условия обозначенные в команде выполнены.	5/3	3/3

16.5.5 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 (PC).

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, PC 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 2-словная или две 1-словные инструкции, следующие за инструкцией возврата выполняются. Это позволяет избежать сброса конвейера, что требует дополнительных циклов. Адрес возврата берётся 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 16-20 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 16-20 – Инструкции безусловного возврата

Инструкция	Описание	Количество циклов Не задерж./Задерж.
RET[D]	Загрузка PC адресом возврата с верхушки стека	5/3
RETE[D]	Загрузка PC адресом возврата с верхушки 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 16-21. 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 остаётся 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 16-21 показывает условную инструкцию возврата и количество циклов необходимых для выполнения этой инструкции.

Таблица 16-21 – 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 определённые в команде выполнены	5/3	3/3

16.5.6 Условные операции

Микропроцессор включает инструкции, которые выполняются только тогда, когда одно или более условий выполнены. Таблица 16-22 перечисляет условия, которые Вы можете использовать с этими инструкциями и соответствующие символы операнда.

Таблица 16-22 – 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
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 16-23). Для каждой комбинации, условия должны быть выбраны из группы 1 или группа 2 следующим образом:

- Группа 1: Вы можете выбрать одно условие из категории A и одно условие из категории B. 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 (A, B, и C). Никакие два условия могут быть из одной и той же категории. Например, Вы можете одновременно протестировать TC, C, и BIO, но Вы не можете протестировать NTC, C, и NC в одно и то же время.

Таблица 16-23 – Группирование множественных условий инструкций

Группа 1		**Группа 2**		
Категория A	Категория B	Категория A	Категория B	Категория C
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 1-цикловой условно выполняемой инструкцией (XC). Есть две формы для инструкции XC. Одна форма – условное выполнение 1-словной инструкции (XC 1, условие). Вторая форма –условное выполнение одной 2-х словной инструкции или двух 1-словных 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 должыо 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 программном способом, а аппаратно. Если команды, предшествующие XC, изменяют некоторые переменные, а те влияют на выполнение инструкции XC, 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 16-24. Условия, использованные условными инструкциями загрузки, указаны в Таблица 16-25.

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 остаётся тем же.

Условные инструкции хранения являются одноопернадными инструкциями, но они используют двухпортовую память операндов в моде косвенной адресации, чтобы разместить инструкцию в одно 16-битовое слово. Следовательно, эти инструкции выполняются в одном цикле.

Условное сохранение счётчика повторения блока (BRC) позволяет Вам сохранить индекс в цикле повторения блока.

Таблица 16-24 – Инструкции условного сохранения (условной записи)

Инструкция	Регистр процессора
SACCD	Аккумулятор A или B
STRCD	Временный регистр
SRCCD	Счётчик повторения блока

Таблица 16-25 – Условия для команд условного хранения

Операнд	Условие	Описание
AEQ	A = 0	Аккумулятор A равен нулю
BEQ	B = 0	Аккумулятор B равен нулю
ANEQ	A ≠ 0	Аккумулятор A не равен нулю
BNEQ	B ≠ 0	Аккумулятор B не равен нулю
ALT	A < 0	Аккумулятор A меньше нуля
BLT	B < 0	Аккумулятор B меньше нуля
ALEQ	A ≤ 0	Аккумулятор A меньше или равен нулю
BLEQ	B ≤ 0	Аккумулятор B меньше или равен нулю
AGT	A > 0	Аккумулятор A больше нуля
BGT	B > 0	Аккумулятор B больше нуля
AGEQ	A ≥ 0	Аккумулятор A больше или равен нулю
BGEQ	B ≥ 0	Аккумулятор B больше или равен нулю

16.5.6.1 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 HM 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 16-26) эффективно становятся одноцикловыми инструкциями после первой итерации инструкции повторения.

Таблица 16-26 – 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 мода длинного смещения или абсолютный адрес (например, *ARn(lk), *+ARn(lk), *+ARn(lk)% и *(lk)). Инструкции указанные в Таблица 16-27 не могут быть повторены использованием RPT.

Таблица 16-27 – 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

16.5.6.2 Повторение блока инструкций

Повторение блока инструкций используется для того, чтобы повторять блок кода N + 1 раз, где N – некоторая величина, загруженная в регистр-счётчик 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 2-словной инструкции или двух 1-словных инструкций, следующих за инструкцией RPTBD вместо очистки конвейера; таким образом, RPTBD эффективно выполняется в 2 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 равные, 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 счётчика 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.

16.6 Функционирование DSP ядра 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 синхпросигнала, в течении нескольких тактов бит RST_DSP_MEM должен находится 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.

16.7 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 неё 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) установливается при возникновении прерывания.
2. Подтверждение прерывания. Микропроцессор должен подтвердить приём 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 предопределённый адрес (позицию вектора прерывания) и выполняют ISR.

16.7.1 Регистры управления прерываниями ядра

Таблица 16-28 – 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 16-29 – Назначение битов регистра

Биты	Наименование	Назначение
15		
14		
13	DMA_INT	Прерывание DMA (завершение обработкиодного иззапросов) 0 – Не активно 1 – Активно
12	CRINT	Прерывание Крипто модуля 0 – Не активно 1 – Активно
11	CDINT	Прерывание Аудио кодека 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 единица в соответствующий бит IFR.
- Инструкция INTR выполнена, используя соответствующий номер прерывания.

Единица 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.

16.7.2 Регистр Маски Прерывания (IMR-interrupt mask register)

Таблица 16-30 показывает, как микропроцессор использует отображенный в памяти регистр IMR для маскирования внешних и внутренних прерываний. Если INTM = 0 в ST1, единица на любом бите IMR допускает соответствующее прерывание. Немаскируемое прерывание от RISC не включены в IMR.

Таблица 16-30 – 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

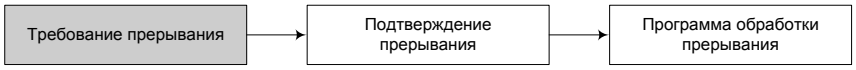
Таблица 16-31 – Назначение битов регистра

Биты	Наименование	Назначение
15	-	зарезервировано
14	-	зарезервировано
13	DMA_INTM	Маска прерывания DMA (завершение обработкиодного иззапросов) 0 – запрещено

		1 – разрешено
12	CRINTM	Маска прерывания Крипто модуля 0 – запрещено 1 – разрешено
11	CDINTM	Маска прерывания Аудио кодека 0 – запрещено 1 – разрешено
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

16.7.3 Обработка прерываний

16.7.3.1 Фаза 1: Приём требования на прерывание



Требование прерывание возникает от аппаратных устройств (в т.ч. и от RISC ядра) 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:

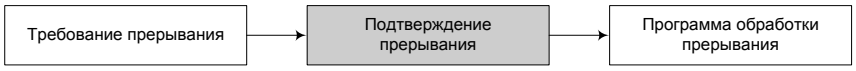
- Прерывания от RISC ядра (регистр AIRQ) INT0…INT3, NMI;
- Сброс ядра DSP (регистр DSP_CONTROL_STATUS подсистемы RISC);
- Прерывания от последовательных портов (RINT0 и XINT0 , RINT1 и XINT1, RINT2 и XINT2, RINT3 и XINT3);
- Прерывание от таймера (TINT);
- Прерывание от крипто модуля (CRINT);
- Прерывание от аудио кодека (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 моды 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.

16.7.3.2 Фаза 2: Подтверждение прерывания

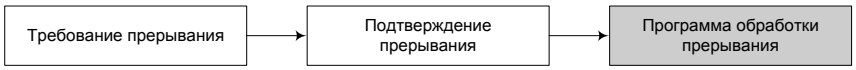


После возникновения требования прерывания от аппаратных или программных средств, CPU должно предложить ответ. Выполнение программ прерывается для немаскируемых аппаратных прерываний немедленно. Маскируемые аппаратные прерывания признаются только после того, как определенные условия будут выполнены:

- *Приоритет самый высокий.* Когда более, чем одно аппаратное прерывание требуется в одно и то же время, микропроцессор обслуживает их согласно приоритету, причём 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.

16.7.3.3 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 2-словную инструкцию или две 1-словные 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 подведёт 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. Адресы прерываний расположены отдельно, разделённые 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.

16.7.3.4 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 необходиморуководствоваться при записи контекста и восстановлении его:

1. Восстановление данных из стэка 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.

16.7.3.5 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.

16.7.3.6 Таблица веторов 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 2 позиции. Смотрите в качестве примера Рисунок 16–47: если возник INT0 и IPTR = 0001h, вектором прерывания выбирается 00C0h. Вектор прерывания для INT0 – 16 или 10h.

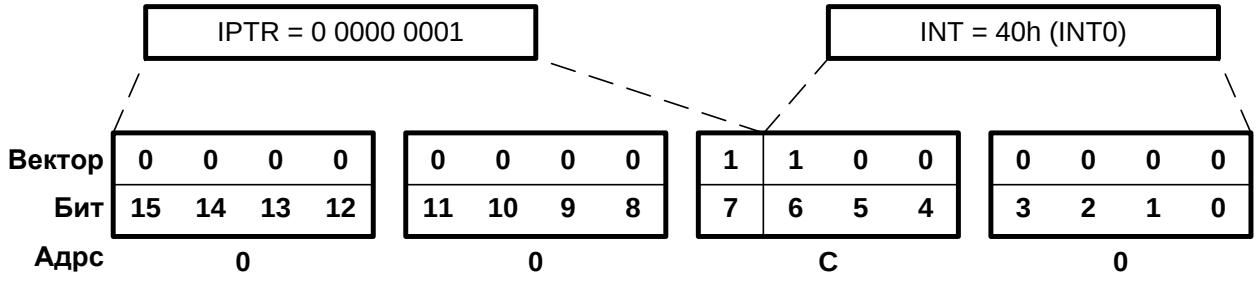


Рисунок 16–47 – Генерация адреса вестора прерываний

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 перемещён, поскольку аппаратный сброс загружает IPTR единицами. Следовательно, вектор аппаратного сброса всегда расположен по адресу FF80h в программном пространстве.

Рисунок 16–48 – Структурная схема процесса прерывания

Таблица 16-32 показывает номер прерывания, приоритет и размещение для микропроцессора.

Таблица 16-32 – Расположение и приоритеты прерываний ‘546

TRAP/INTR номер (К)	Приоритет	Название	Расположение (hex)	Фунция
0	1	$\overline{\text{RS}}$, SINTR	0	сброс (аппаратный и программный сброс)
1	2	$\overline{\text{NMI}}$, 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	$\overline{\text{INT0}}$, SINT0	40	прерывание от RISC 0 (регистр AIRQ)
17	4	$\overline{\text{INT1}}$, SINT1	44	прерывание от RISC 1 (регистр AIRQ)
18	5	$\overline{\text{INT2}}$, 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	$\overline{\text{INT3}}$, 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 крипто модуля
29	16	DMAINT	74	Прерывание DMA (DSP)
30-31	-		78-7C	зарезервировано

16.8 Регистры управления и состояния ядра

Микропроцессор имеет три регистра статуса и управления:

- Регистр состояния 0 (ST0);
- Регистр состояния 1 (ST1);
- Регистр состояния моды процессора (PMST).

ST0 и ST1 содержит состояние различных условий и мод; 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).

16.8.1 Регистры Статуса (ST0 и ST1)

Индивидуальные биты регистров ST0 и ST1 могут быть установлены или сброшены инструкциями SSBX и RSBX. Например, мода 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.

Биты ST0 показаны на Рисунок 16–49 и описаны в Таблица 16-33. Биты ST1 показаны на Рисунок 16–50 и описаны в Таблица 16-34.

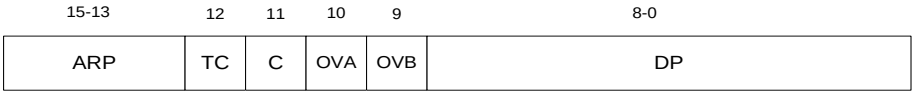


Рисунок 16–49 – Регистр состояния 0 (ST0)

Таблица 16-33 – Назначение битов регистра состояния 0 (ST0)

Биты	Обозначение	Значение по сбросу	Функция
15-13	ARP	0	Указатель вспомогательного регистра. Это 3-х битовое поле выбирает вспомогательный регистр, используемый в совместимой моде косвенной адресации одного операнда. ARP должен быть всегда установлен в нуль, когда DSP в стандартной моде (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 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 разнящиеся друг от друга.

Биты	Обозначение	Значение по сбросу	Функция
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 A. OVA устанавливается в 1, когда происходит переполнение в ALU или сумматоре умножителя и расположение для результата - аккумулятор A. 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 B. OVB устанавливается в 1, когда происходит переполнение в ALU или сумматоре умножителя и расположение для результата – аккумулятор B. 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 моды компиляции в ST1 (CPL) = 0. Поле DP может быть загружено инструкцией LD с коротким непосредственно заданным операндом или из памяти данных.

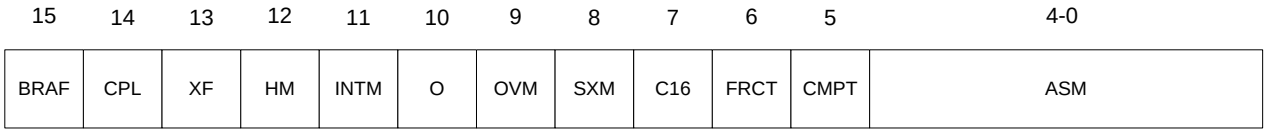


Рисунок 16–50 – Регистр состояния 1 (ST1)

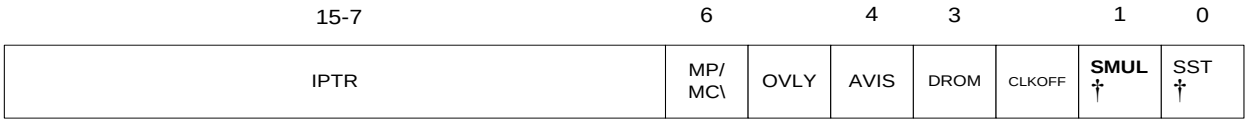
Таблица 16-34 – Назначение битов регистра состояния 1 (ST1)

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 дезактивировано. 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 мода относительной прямой адресации, использовавший выбор страничного указателя данных (DP). CPL = 1 мода 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	Мода 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 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

Биты	Обозначение	Значение по сбросу	Функция
9	OVM	0	Мода 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	Мода 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	Мода двойной 16- битовой или спаренной точности арифметических операций. C16 определяет моду арифметических действий в ALU: C16 = 0 ALU работает в моде двойной точности арифметических действий. C16 = 1 ALU работает в моде спаренных арифметических действий.
6	FRCT	0	Мода дробности. Когда FRCT в 1, выход множителя сдвигается влево на один бит, чтобы компенсировать дополнительный бит знака.
5	CMPT	0	Мода совместимости. CMPT определяет способ совместимости для ARP: CMPT = 0 ARP не обновляется в моде косвенной адресации с единственным операндом в памяти данных. ARP должен быть всегда установлен в 0, когда DSP - в этом способе. CMPT = 1 ARP обновляется в моде косвенной адресации с единственным операндом в памяти данных, кроме тех случаев, когда инструкция выбирает вспомогательный регистр 0 (AR0).
4-0	ASM	0	Мода сдвига аккумулятора. 5- битовая 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.

Регистр состояния моды процессора (PMST)

Регистр PMST загружается инструкциями работающими с регистрами, отображенными в памяти, как например, STM. Биты PMST показаны на Рисунок 16–51 и описаны в Таблица 16-35.



† В данной модели микропроцессора эти биты не поддержаны.

Рисунок 16–51 – Регистр состояния моды процессора (PMST)

Таблица 16-35 – Назначение битов регистра PMST

Биты	Обозначение	Значние 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	Мода 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 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	Мода 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 эта мода позволяет внутреннему программному адресу появляться на контактах

Биты	Обозначение	Значние по сбросу	Функция
			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 дробных чисел прежде, чем последующее сложение/вычитание потребуется в MAC или MAS инструкциях. В этой моде, 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

Биты	Обозначение	Значние по сбросу	Функция
			• 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).

17 Система команд DSP

17.1 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 так же класс команды. Здесь же размещена информация о повторяющихся и неповторяющихся командах.

17.1.1 Арифметические операции

Этот раздел включает команды арифметических операций. С таблицах ниже приведены команды следующих функциональных групп:
1. Команды сложения (Таблица 17-1);
2. Команды вычитания (Таблица 17-2);
3. Команды умножения (Таблица 17-3);
4. Команды умножения со сложением (Таблица 17-4);
5. Команды умножения с вычитанием (Таблица 17-4);
6. Команды с двойным операндом (32 бита) (Таблица 17-5);
7. Проблемно-ориентированные команды (Таблица 17-6).

Таблица 17-1 – 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 1 слово и 1 цикл при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem.

Таблица 17-2 – Команды вычитания

Синтаксис	Выражение (операция)	W †	C †	Класс
SUB Smem, src	src = src – Smem	1	1	3A, 3B
SUB Smem, TS, src	src = src – Smem << TS	1	1	3A, 3B
SUB Smem, 16, src [, dst]	dst = src – Smem << 16	1	1	3A, 3B
SUB Smem [, SHIFT], src [, dst]	dst = src – Smem << SHIFT	2	2	4A, 4B
SUB Xmem, SHFT, src	src = src – Xmem << SHFT	1	1	3A
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	3A, 3B

SUBC Smem, src	If (src – Smem << 15) _ 0
	src = (src – Smem << 15) << 1 + 1
	Else
	src = src << 1
		1	1	3A, 3B

SUBS Smem, src	src = src – uns(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 1 слово и 1 цикл при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem.

Таблица 17-3 – Команды умножения

Синтаксис	Выражение (операция)	W †	C †	Класс
MPY Smem, dst	dst = T * Smem	1	1	3A, 3B
MPYR Smem, dst	dst = rnd(T * Smem)	1	1	3A, 3B
MPY Xmem, Ymem, dst	dst = Xmem * Ymem, T = Xmem	1	1	7
MPY Smem, #lk, dst	dst = Smem * #lk , T = Smem	2	2	6A, 6B
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	3A, 3B
MPYU Smem, dst	dst = uns(T) * uns(Smem)	1	1	3A, 3B
SQUR Smem, dst	dst = Smem * Smem, T = Smem	1	1	3A, 3B
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 1 слово и 1 цикл при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem.

Таблица 17-4 – 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 1 слово и 1 цикл при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem.

Таблица 17-5 – 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 1 слово и 1 циклt при использовании длинного смещения в косвенной адресации или абсолютной адресации Lmem.

Таблица 17-6 – 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
EXP src	T = number of sign bits (src) – 8	1	1	1
FIRS Xmem, Ymem, pmad	B = B + A * pmad A = (Xmem + Ymem) << 16	2	3	8
LMS Xmem, Ymem	B = B + Xmem * Ymem A = A + Xmem << 16 + 215	1	1	7
MAX dst	dst = max(A, B)	1	1	1
MIN dst	dst = min(A, B)	1	1	1
NEG src [, dst]	dst = –src	1	1	1
NORM src [, dst]	dst = src << TS dst = norm(src, TS)	1	1	1
POLY Smem	B = Smem << 16 A = rnd(A(32–16) * T + B)	1	1	3A, 3B
RND src [, dst]	dst = src + 215	1	1	1
SAT src	saturate(src)	1	1	1
SQDST Xmem, Ymem	B = B + A(32–16) * A(32–16) A = (Xmem – Ymem) << 16	1	1	7

† 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 при использовании длинного смещения в косвенной адресации или абсолютной адресации Lmem.

17.1.2 Логические операции

Этот раздел включает команды логических операций. Таблицы ниже содержат команды следующих функциональных групп:

1. Команды И (AND) (Таблица 17-7);
2. Команды ИЛИ (OR) (Таблица 17-8);
3. Команды, исключающие ИЛИ (XOR) (Таблица 17-9);
4. Команды сдвига (Таблица 17-10);
5. Команды тестирования (Таблица 17-11).

Таблица 17-7 – Команды AND

Синтаксис	Выражение (операция)	W †	C †	Класс
AND Smem, src	src = src & Smem	1	1	3A, 3B
AND #lk [, SHFT], src [, dst]	dst = src & #lk << SHFT	2	2	2
AND #lk, 16, src [, dst]	dst = src & #lk << 16	2	2	2
AND src [, SHIFT] [, dst]	dst = dst & src << SHIFT	1	1	1
ANDM #lk, Smem	Smem = Smem & #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 1 слово и 1 цикл при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem.

Таблица 17-8 – Команды OR

Синтаксис	Выражение (операция)	W †	C †	Класс
OR Smem, src	src = src | Smem	1	1	3A, 3B
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	18A, 18B

† 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 при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem.

Таблица 17-9 – Команды XOR

Синтаксис	Выражение (операция)	W †	C †	Класс
XOR Smem, src	src = src ^ Smem	1	1	3A, 3B
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	18A, 18B

† 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 при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem.

Таблица 17-10 – Команды сдвига

Синтаксис	Выражение (операция)	W †	C †	Класс
ROL src	циклический сдвиг влево через перенос	1	1	1
ROLTC src	циклический сдвиг влево через TC	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 17-11 – 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 1 слово и 1 цикл при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem.

17.1.3 Команды управления программой

Этот раздел включает команды управления ходом программы. Таблицы ниже содержат команды следующих функциональных групп:

1. Команды перехода (Таблица 17-12);
2. Команды вызова (Таблица 17-13);
3. Команды прерывания (Таблица 17-14);
4. Команды возврата (Таблица 17-15);
5. Команды повторения (Таблица 17-16);
6. Команды управления стеком (Таблица 17-17);
7. Разные команды управления программой (Таблица 17-18).

Таблица 17-12 – Команды перехода

Синтаксис	Выражение (операция)	W †	C †	Класс
B[D] pmad	PC = pmad(15–0)	2	4/[2¶]	29A
BACC[D] src	PC = src(15–0)	1	6/[4¶]	30A
BANZ[D] pmad, Sind	if (Sind _ 0) then PC = pmad(15–0)	2	4‡/2§/[2¶]	29A
BC[D] pmad, cond [, cond [, cond]]	if (cond(s)) then PC = pmad(15–0)	2	5‡/3§/[3¶]	31A
FB[D] extpmad (в данной модели не реализовано)	PC = pmad(15–0), XPC = pmad(22–16)	2	4/[2¶]	29A
FBACC[D] src (в данной модели не реализовано)	PC = src(15–0), XPC = src(22–16)	1	6/[4¶]	30A

† количество слов (W) и циклов (C) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных.

‡ условие истинно.

§ условие ложно.

¶ задержанная команда.

Таблица 17-13 – Команды вызова

Синтаксис	Выражение (операция)	W †	C †	Класс
CALA[D] src	– –SP, PC + 1[3¶] = TOS, PC = src(15–0)	1	6/[4¶]	30B
CALL[D] pmad	– –SP, PC + 2[4¶] = TOS, PC = pmad(15–0)	2	4/[2§]	29B
CC[D] pmad, cond [, cond [, cond]]	if (cond(s)) then – –SP, PC + 2[4¶] = TOS, PC = pmad(15–0)	2	5‡/3§/[3¶]	31B
FCALA[D] src (в данной модели не реализовано)	– –SP, PC + 1 [3¶] = TOS, PC = src(15–0), XPC = src(22–16)	1	6/[4¶]	30B
FCALL[D] extpmad (в данной модели не реализовано)	– –SP, PC + 2[4¶] = TOS, PC = pmad(15–0), XPC = pmad(22–16)	2	4/[2¶]	29B

† количество слов (W) и циклов (C) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных.

‡ условие истинно.

§ условие ложно.

¶ задержанная команда.

Таблица 17-14 – 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 17-15 – Команды возврата

Синтаксис	Выражение (операция)	W †	C †	Класс
FRET[D] (в данной модели не реализовано)	XPC = TOS, ++ SP, PC = TOS, ++SP	1	6/[4¶]	34
FRETE[D] (в данной модели не реализовано)	XPC = TOS, ++ SP, PC = TOS, ++SP, INTM = 0	1	6/[4¶]	34
RC[D] cond [, cond [, cond]]	if (cond(s)) then PC = TOS, ++SP	1	5‡/3§/[3¶]	32
RET[D]	PC = TOS, ++SP	1	5/[3¶]	32
RETE[D]	PC = TOS, ++SP, INTM = 0	1	5/[3¶]	32
RETF[D]	PC = RTN, ++SP, INTM = 0	1	3/[1¶]	33

† количество слов (W) и циклов (C) соответствуют использованию расслоенной RAM для данных, когда обмен адресуется к разным блокам данных.

‡ условие истинно.

§ условие ложно.

¶ задержанная команда.

Таблица 17-16 – Команды повторения

Синтаксис	Выражение (операция)	W †	C †	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¶], REA = pmad, BRAF = 1	2	4/[2¶]	29A
RPTZ dst, #lk	Repeat single, RC = #lk, dst = 0	2	2	2

† 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 при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem.

¶ задержанная команда

Таблица 17-17 – 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 1 слово и 1 цикл при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem.

Таблица 17-18 – 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 циклt при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem.

17.1.4 Команды загрузки и сохранения

Этот раздел включает команды загрузки и сохранения. Таблицы ниже содержат команды следующих функциональных групп:

1. Команды загрузки (Таблица 17-19);
2. Команды сохранения (Таблица 17-20);
3. Команды условного сохранения (Таблица 17-21);
4. Команды параллельной загрузки и сохранения (Таблица 17-22);
5. Команды параллельной загрузки и умножения (Таблица 17-23);
6. Команды параллельного сохранения и сложения/вычитания (Таблица 17-24);
7. Команды параллельного сохранения и умножения (Таблица 17-25);
8. Разные команды типа загрузки и сохранения (Таблица 17-26).

Таблица 17-19 – 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 1 слово и 1 цикл при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem или Lmem.

Таблица 17-20 – Команды сохранения

Синтаксис	Выражение (операция)	W †	C †	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 1 слово и 1 цикл при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem или Lmem.

Таблица 17-21 – Команды условного сохранения

Синтаксис	Выражение (операция)	W †	C †	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 17-22 – Команды параллельной загрузки и сохранения

Синтаксис	Выражение (операция)	W †	C †	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 17-23 – 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 17-24 – 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 17-25 – 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 17-26 – 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 1 слово и 1 цикл при использовании длинного смещения в косвенной адресации или абсолютной адресации Smem.

17.1.5 Повторения одной команды

Процессор 1901ВЦ1Т 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 повторении (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 HM 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 17-27) становятся одноцикловыми после первой итерации команды повторения.

Таблица 17-27 – Многоцикловые команды, выполняемые за один цикл при повторении

Команда	Описание	# Циклы †
FIRS	симметричный FIR-фильтр	3
MACD	умножаются с аккумулированием и задержкой	3
MACP	умножаются с аккумулированием	3
MVDK	пересылка данные-данные	2
MVDM	пересылка данные-регистры MMR	2
MVDP	пересылка данные-программа	4
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 17-28 содержит команды, которые не могут быть повторены командами RPT или RPTZ.

Таблица 17-28 – 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

Команда	Описание
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
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

17.2 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, случае процессора 1901ВЦ1Т, 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 1901ВЦ1Т.

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, приведённых 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

1 слово, 1 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

2 слова, 2 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 3A

1 слово, 1 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 3B

2 слова, 2 цикла. Операнд чтения в памяти данных (Smem), использущий 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 4A

2 слова, 2 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 4B

3 слова, 3 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 5A

1 слово, 1 цикл. Операнд чтения в памяти данных (Smem) (с назначением в качестве приёмника 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 5B

2 слова, 2 цикла. Операнд чтения в памяти данных (Smem) с использованием косвенной адресации с длинным смещением (с назначением в качестве приёмника 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 6A

2 слова, 2 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 6B

3 слова, 3 цикла. Операнд чтения в памяти данных (Smem), использущий косвенную адресацию с длинным смещением и операнд с длинным непосредственным значением.

Мнемоника:

BITF CMPM MAC MPY

Циклы

Циклы единичного выполнения, использущие 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

Класс 7

1 слово, 1 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, 2†,3‡	1+p, 2*
	DROM	2	1,2†	1+p
	External	1+d	1+d, 2‖	2+d+p
DROM	RAM	2	1, 2†	1+p
	DROM	3	2	1+p, 2*
	External	1+d, 2‖	1+d	2+d+p
External	RAM	1+d	1+d,2†	2+d+p
	DROM	1+d, 2‖	1+d	2+d+p
	External	2+2d	2+2d	3+2d+p
MMR	RAM	1	1,2†	1+p
	DROM	2	1	1+p
	External	1+d	1+d	2+d+p

¤ Операнды в одном блоке памяти
† Операнд и код в одном блоке памяти.
‡ два операнда и код в одном блоке памяти.
‖ операнд и код в одном блоке памяти при d=0.
* два операнда в одном блоке памяти при p=0.

Циклы повторяющихся выполнений

Операнд		Программа		
Xmem	Ymem	ROM	RAM	External
RAM	RAM	n, 2n#	n, 2n#,2n+1‡	n+p, 2n (p=0)#, 2n-1+p(p≥1)#
	DROM	n+1	n,n+1†	n+p
	External	n+nd	n+nd,n+nd+1†	n+nd+1+p
DROM	RAM	n+1†	n+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†	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†	n+p
	DROM	n+1	n	n+p
	External	n+nd	n+nd	n+nd+1+p

† Операнд и код в одном блоке памяти.
‡ два операнда и код в одном блоке памяти.
# два операнда в одном блоке памяти.

Класс 8

2 слова, 3 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†,3§	3, 4†	2+p, 3+p†
		DROM	2	2, 3†	2+p, 3+p†
		External	1+d, 2+d†	2+d,3+d†	2+d+p
	DROM	RAM	2	2, 3†	2+p,3+p†
		DROM	3	2	2+p
		External	2+d	2+d	2+d+p
	External	RAM	1+d,2+d†	2+d,3+d†	2+d+p,3+d+p†
		DROM	2+d	2+d	2+d+p
		External	2+2d	2+2d	3+2d+p
DROM	RAM	RAM	2	2,3†	1+p,2+p(p=0,‡)
		DROM	3	2	2(p=0),1+p(p≠0)
		External	2(d=0),1+d(d≠0)	2(d=0&†),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+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+d,3+d†‡	2+d+p
		DROM	2(d=0),1+d(d≠0)	2(d=0&†),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+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

† Xmem(Ymem) и pmad в одном блоке памяти.
‡ Xmem и Ymem в одном блоке памяти.
§ Xmem, Ymem и pmad в одном блоке памяти.

Циклы повторяющихся выполнений (n>1)

Операнд			Программа		
pmad	Xmem	Ymem	ROM	RAM	External
RAM	RAM	RAM	n,2n†,3n§	n,2n+1†‡, 3n+1¶	n+p,2n+p†‡, 3n+p†
		DROM	n+1,2n+1†	n,2n†, 3n§†	n+p, 2n+p†
		External	n+nd, 2n(d=0)†	n+nd, 2n(d=0)†	1+n+nd+p, 2n+p(d=0) †

	DROM	RAM	1+n,1+2n†	1+n,1+2n†	n+p,2n+p†
		DROM	2n+1	2n	2n+p
		External	n+1(d=0),n+nd(d≠0)	n+nd	n+nd+p
	External	RAM	n+nd, 2n(d=0)†	n+nd, 2n(d=0)†	1+n+nd+p, 2n+p(d=0) †
		DROM	n+1(d=0),n+nd(d≠0)	n+nd	n+nd+p
		External	2n+2nd	2n+2nd	2n+2nd+p
DROM	RAM	RAM	1+n,1+2n†	1+n,1+2n†	n+p,2n+p†
		DROM	2n+1	2n	2n+p
		External	n+1(d=0),n+nd(d≠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≠0)	2n(d=0),n+nd(d≠0)	n+nd+p
	External	RAM	n+1(d=0),n+nd(d≠0)	n+nd	n+nd+p
		DROM	2n(d=0),n+nd(d≠0)	2n(d=0),n+nd(d≠0)	n+nd+p
		External	2n+2nd	2n+2nd	2n+2nd+p
External	RAM	RAM	n+nd, 2n(d=0)†	n+nd, 2n(d=0)†	1+n+nd+p, 2n+p(d=0) †
		DROM	n+1(d=0),n+nd(d≠0)	n+nd	n+nd+p
		External	2n+2nd	2n+2nd	2n+2nd+p
	DROM	RAM	n+1(d=0),n+nd(d≠0)	n+nd	n+nd+p
		DROM	2n(d=0),n+nd(d≠0)	2n(d=0),n+nd(d≠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

† Xmem и pmad в одном блоке памяти.
‡ Xmem и Ymem в одном блоке памяти.
§ Ymem и pmad в одном блоке памяти.
¶ Xmem, Ymem и pmad в одном блоке памяти.

Класс 9A

1 слово, 1 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 9B

2 слова, 2 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 10A

1 слово, 1 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 10B

2 слова, 2 цикла. Операнд записи в память данных (Smem или Xmem) использующий косвенную адресацию с длинным смещением.

Мнемоника:

CMPS ST STH STL

Циклы:

Циклы единичного выполнения, использущие 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 11A

2 слова, 2 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 11B

3 слова, 3 цикла Операнд записи в память данных (Smem) использующий косвенную адресацию с длинным смещением.

Мнемоника:

STH STL

Циклы:

Циклы единичного выполнения, использущие 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 12A

2 слова, 2 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 12B

3 слова, 3 цикла. Операнд записи в память данных (Smem), использующий косвенную адресацию с длинным смещением.

Мнемоника: **ST**

Циклы:

Циклы единичного выполнения, использущие 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 13A

1 слово, 2 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 13B

2 слова, 3 цикла. Операнд записи в памяти данных, использующий косвенную адресацию длинного слова (Lmem) с длинным смещением.

Мнемоника:

DST

Циклы:

Циклы единичного выполнения, использущие 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

1 слово, 1 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†	1,2†,3†‡	1+p,2(p=0, ‡)
	External	1+d	1+d	2+d+p
DROM	RAM	2	1,2†	1+p
	External	1+d, 2(d=0,†)	1	2+d+p
External	RAM	1+d	1+d,2(d=0,†)	2+d+p
	External	2+2d	2+2d	3+2d+p
MMR	RAM	1	1, 2†	1+p
	External	1+d	1+d	2+d+p

† Операнд и код в одном блоке памяти.
‡ два операнда в одном блоке памяти.

Циклы повторяющихся выполнений

Операнд		Программа		
Xmem	Ymem	ROM	RAM	External
RAM	RAM	n,2n#	n,n+1†,2n#, 2n+1†#	n+p,2n+p#
	External	n+nd	n+nd,n+nd+1†	n+nd+1+p

DROM	RAM	n,n+1†	n, n+1†	n+p
	External	n+nd+1	n+nd	n+nd+1+p
External	RAM	n+nd	n+nd,n+nd+1†	n+nd+1+p
	External	2n+2nd	2n+2nd	2n+2nd+p
MMR	RAM	n	n, n+1†	n+p
	External	n+nd	n+nd	n+nd+1+p

† Операнд и код в одном блоке памяти.
‡ два операнда и код в одном блоке памяти.
# два операнда в одном блоке памяти.

Класс 15

1 слово, 1 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

1 слово, 1 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†,3‡	1+p,2(p=0) #
	External	1	1, 2†	2+d+p

DROM	RAM	2	1,2†	1+p
	External	2(d=0),1+d(d≠0)	1	2+d+p
External	RAM	1+d	1+d,2+d†	2+d+p
	External	2+2d	2+2d	3+2d+p
MMR	RAM	1	1, 2†	1+p
	External	1	1	2+d+p

† Операнд и код в одном блоке памяти.
‡ два операнда и код в одном блоке памяти.
# два операнда в одном блоке памяти.

Циклы повторяющихся выполнений

Операнд		Программа		
Xmem	Stack	ROM	RAM	External
RAM	RAM	n, 2n#	n,n+1†‡,n+2‡	n+p,n+p+1#
	External	n+nd	n+nd,n+nd+1†	n+nd+1+p
DROM	RAM	n+1	n, n+1†	n+p
	External	n+nd,n+nd+1†	n+nd	n+nd+1+p
External	RAM	n+nd	n+nd,n+nd+1†	n+nd+1+p
	External	2n+2nd	2n+2nd	2n+2nd+1+p
MMR	RAM	n	n, 2n†	n+p
	External	n+nd	n+nd	n+nd+1+p

† Операнд и код в одном блоке памяти.
‡ два операнда и код в одном блоке памяти.
# два операнда в одном блоке памяти.

Класс 16B

2 слова, 2 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, использущие 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≥2)	2,1+d(d≥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 17A

1 слово, 1 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, 2†,3‡	1+p,2(p=0) #
	DROM	2	1,2†	1+p
	External	1+d	1+d, 2+d†	2+d+p
	MMR	1	1,2†	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

† Операнд и код в одном блоке памяти.
‡ два операнда и код в одном блоке памяти.
# два операнда в одном блоке памяти.

Циклы повторяющихся выполнений

Операнд		Программа		
Smem	Stack	ROM	RAM	External
RAM	RAM	n, 2n#	n, n+1†,2n#	n+p,2n+p#
	DROM	n+1	n n+1†	n+p
	External	n+nd	n+nd,n+dn+1†	n+nd+1+p
	MMR	n	n, n+1†	n+p
External	RAM	n+nd	n+nd,n+nd+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

† Операнд и код в одном блоке памяти.
‡ два операнда и код в одном блоке памяти.
# два операнда в одном блоке памяти.

Класс 17B

2 слова, 2 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†	2, 3‡	2+2p
	DROM	3	2, 3†	2+2p
	External	1+d	1+d,2+d	3+d+2p
	MMR	2	2, 3†	2+2p
External	RAM	2	2, 3†	3+d+2p
	DROM	3	2	3+d+2p
	External	2+2d	2+2d	4+2d+2p
	MMR	2	2	3+d+2p

† Операнд и код в одном блоке памяти.
‡ два операнда и код в одном блоке памяти.
# два операнда в одном блоке памяти.

Класс 18A

2 слова, 2 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 18B

3 слова, 3 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 19A

2 слова, 2 цикла. Операнд чтения в памяти данных (Smem) или операнд чтения в MMR, операнд записи в памяти данных (dmad); или операнд чтения в памяти данных (dmad), и операнд записи в памяти данных (Smem) или операнд записи в MMR.

Мнемоника:

MVDK MVDM MVKD MVMD

Циклы:

Циклы единичного выполнения

Операнд		Программа		
Smem	dmad	ROM	RAM	External
RAM	RAM	2, 3#	2, 3#, 4‡	2+2p
	External	2+d	2+d	3+d+2p
	MMR	2	2,3†	2+2p
DROM	RAM	3	2,3†	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†	2+2p
	External	2+d	2+d	3+d+2p
	MMR	2	2	2+2p

† Операнд и код в одном блоке памяти.
‡ два операнда и код в одном блоке памяти.
# два операнда в одном блоке памяти.

Циклы повторяющихся выполнений

Операнд		Программа		
Smem	dmad	ROM	RAM	External
RAM	RAM	n, 2n#	n,n+1†,2n#,2n+1‡	n+1+2p,2n+1+2p#
	External	n+nd	n+nd,n+nd+1†	n+nd+2+2p
	MMR	n	n,n+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†	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†	n+1+2p
	External	2n+2nd	2n+2nd	2n+2nd+2+2p
	MMR	n	n	n+1+2p

† Операнд и код в одном блоке памяти.
‡ два операнда и код в одном блоке памяти.
# два операнда в одном блоке памяти.

Класс 19B

2 слова, 2 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†	2, 3#, 4‡	2+2p
	External	2+d	2+d	3+d+2p
	MMR	2	2,3†	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†	2+2p
	External	2+d	2+d	3+d+2p
	MMR	2	2	2+2p

† Операнд и код в одном блоке памяти.
‡ два операнда и код в одном блоке памяти.
# два операнда в одном блоке памяти.

Класс 20A

2 слова, 4 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#	2,3#,4‡	2+2p,3+2p#
	External	2+d	2+d,3+d#	3+d+2p
DROM	RAM	3	2,3†	2+2p
	External	3+d	2+d	3+d+2p
External	RAM	2+d	2+d,3+d†	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

† Операнд и код в одном блоке памяти.
‡ два операнда и код в одном блоке памяти.
# два операнда в одном блоке памяти.

Циклы повторяющихся выполнений

Операнд		Программа		
Smem	pmad	ROM	RAM	External
RAM	RAM	2+n,3+n#	2+n,3+n#,4+n‡	n+1+2p,2n+1+2p#
	External	n+nd+1	n+nd+1,n+nd+2†	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

† Операнд и код в одном блоке памяти.
‡ два операнда и код в одном блоке памяти.
# два операнда в одном блоке памяти.

Класс 20B

3 слова, 5 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, использущие косвенную адресацию с длинным смещением

Операнд		Программа		
Smem	pmad	ROM	RAM	External
RAM	RAM	3, 4#	3,4#,5‡	3+3p,4+3p#
	External	3+d	3+d,4+d#	4+d+3p
DROM	RAM	4	3,4†	3+3p
	External	4+d	3+d	4+d+3p
External	RAM	3+d	3+d,4+d†	4+d+3p
	External	3+2d	3+2d	5+2d+3p
MMR	RAM	3	3,4‡	3+3p
	External	3+d	3+d	3+d+3p

† Операнд и код в одном блоке памяти.
‡ два операнда и код в одном блоке памяти.
# два операнда в одном блоке памяти.

Класс 21A

2 слова, 3 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#	2,3#,4‡	2+2p,3+2p#
	External	2+d	2+d,3+d‡	3+d+2p
	MMR	2	2,3†	2+2p
PROM	RAM	3	2,3†	2+2p
	External	3+d	2+d	3+d+2p
	MMR	3	2	2+2p
External	RAM	2+d	2+d,3+d†	3+d+2p
	External	2+2d	2+2d	4+2d+2p
	MMR	2+d	2+d	3+d+2p

† Операнд и код в одном блоке памяти.
‡ два операнда и код в одном блоке памяти.
# два операнда в одном блоке памяти.

Циклы повторяющихся выполнений

Операнд		Программа		
pmad	Smem	ROM	RAM	External
RAM	RAM	n+2	n+2,2n#	n+2+2p,2n+2p#
	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

† Операнд и код в одном блоке памяти.
‡ два операнда и код в одном блоке памяти.
# два операнда в одном блоке памяти.

Класс 21B

3 слова, 4 цикла. Операнд чтения в памяти программ (pmad) и операнд записи в памяти данных (Smem), использующий максимально удаленную косвенную адресацию.

Мнемоника:

MVPD

Циклы:

Циклы единичного выполнения, использущие косвенную адресацию с длинным смещением

Операнд		Программа		
pmad	Smem	ROM	RAM	External
RAM	RAM	3,4#	3,4#,5‡	3+3p,4+3p#
	External	3+d	3+d,4+d‡	4+d+3p
	MMR	3	3,4†	3+3p
PROM	RAM	4	3,4†	3+3p
	External	4+d	3+d	4+d+3p
	MMR	4	3	3+3p
External	RAM	3+d	3+d,4+d†	4+d+3p
	External	3+3d	3+3d	5+2d+3p
	MMR	3+d	3+d	3+d+3p

† Операнд и код в одном блоке памяти.
‡ два операнда и код в одном блоке памяти.
# два операнда в одном блоке памяти.

Класс 22A

2 слова, 3 цикла. Операнд чтения в памяти данных (Smem) и операнд чтения в памяти программ (pmad).

Мнемоника:

MACP

Циклы:

Циклы единичного выполнения

Операнд		Программа		
pmad	Smem	ROM	RAM	External
RAM	RAM	2,3†	2,3†,4‡	2+2p,3+2p#
	External	2+d	2+d,3+d†	3+d+2p
	MMR	2	2,3†	2+2p
PROM	RAM	3	3, 4†	2+2p
	External	3+d	3+d	3+d+2p
	MMR	3	3	2+2p
External	RAM	2+d	2+d, 3+d†	3+d+2p
	External	3+2d	3+2d	4+2d+2p
	MMR	2+d	2+d	3+d+2p

† Операнд и код в одном блоке памяти.
‡ два операнда и код в одном блоке памяти.
# два операнда в одном блоке памяти.

Циклы повторяющихся выполнений

Операнд		Программа		
pmad	Smem	ROM	RAM	External
RAM	RAM	n+2,2n+2#	n+2, n+3†,2n+2#	n+2+2p,2n+2+2p#
	External	n+nd+2	n+nd+2	n+nd+2+2p
	MMR	n+2	n+2,n+3†	n+2+2p
PROM	RAM	n+3	n+3, n+4†,	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†	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

† Операнд и код в одном блоке памяти.
# два операнда в одном блоке памяти.

Класс 22B

3 слова, 4 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, использущие косвенную адресацию с длинным смещением

Операнд		Программа		
pmad	Smem	ROM	RAM	External
RAM	RAM	3,4†	3,4†,5‡	3+3p,4+3p#
	External	3+d	3+d,4+d†	4+d+3p
	MMR	3	3,4†	3+3p
PROM	RAM	4	4, 5†	4+3p
	External	4+d	4+d	5+d+3p
	MMR	4	4	4+3p
External	RAM	3+d	3+d, 4+d†	4+d+3p
	External	4+2d	4+2d	5+2d+3p
	MMR	3+d	3+d	4+d+3p

† Операнд и код в одном блоке памяти.

Класс 23A

2 слова, 3 цикла. Операнд чтения в памяти данных (Smem), операнд записи в памяти данных (Smem), и операнд чтения в памяти программ (pmad).

Мнемоника:

MACD

Циклы:

Циклы единичного выполнения

Операнд		Программа		
pmad	Smem	ROM	RAM	External
RAM	RAM	2,3#	2,3#,4‡	2+2p,3+2p#
	External	2+d	2+d,3+d‡	3+d+2p
	MMR	2	2,3†	2+2p
PROM	RAM	3	2,3†	2+2p
	External	3+d	2+d	3+d+2p
	MMR	3	2	2+2p
External	RAM	2+d	2+d,3+d†	3+d+2p
	External	2+2d	2+2d	4+2d+2p
	MMR	2+d	2+d	3+d+2p

† Операнд и код в одном блоке памяти.
‡ два операнда и код в одном блоке памяти.
# два операнда в одном блоке памяти.

Циклы повторяющихся выполнений

Операнд		Программа		
pmad	Smem	ROM	RAM	External
RAM	RAM	n+2	n+2,2n#	n+2+2p,2n+2p#
	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

† Операнд и код в одном блоке памяти.
# два операнда в одном блоке памяти.

Класс 23B

3 слова, 4 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, использущие косвенную адресацию с длинным смещением

Операнд		Программа		
pmad	Smem	ROM	RAM	External
RAM	RAM	3,4#	3,4#,5‡	3+3p,4+3p#
	External	3+d	3+d,4+d‡	4+d+3p
	MMR	3	3,4†	3+3p
PROM	RAM	4	3,4†	3+3p
	External	4+d	3+d	4+d+3p
	MMR	4	3	3+3p
External	RAM	3+d	3+d,4+d†	4+d+3p
	External	3+3d	3+3d	5+2d+3p
	MMR	3+d	3+d	3+d+3p

† Операнд и код в одном блоке памяти.
# два операнда в одном блоке памяти.

Класс 24A

1 слово, 1 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 24B

2 слова, 2 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, использущие 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 25A

1 слово, 5 циклов. Адрес чтения в памяти программ (pmad адресуемый аккумулятором A) 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+p, 2+p#
	External	1+d	1+d	2+d+p
	MMR	1	1	1+p
PROM	RAM	2	1, 2†	1+p
	External	2	1+d	2+d+p
	MMR	2	1	1+p
External	RAM	1+d	1+d,2+d†	2+d+p
	External	2+2d	2+2d	3+2d+p
	MMR	1+d	1+d	2+d+p

† операнд и код в одном блоке памяти.
# два операнда в одном блоке памяти.

Циклы повторяющихся выполнений

Операнд		Программа		
pmad	Smem	ROM	RAM	External
RAM	RAM	n	n,n+1†	n+p, 2n+p#
	External	n+nd	n+nd	n+nd+1+p
	MMR	n	n	n+1+p
PROM	RAM	n+1	n+1, n+2†	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†	2n+nd+p
	External	2n+2nd	2n+2nd	2n+2nd+1+p
	MMR	n+nd	n+nd	n+nd+1+p

† операнд и код в одном блоке памяти.
# два операнда в одном блоке памяти.

Класс 25B

2 слова, 6 циклов. Адрес чтения в памяти программ (pmad) операнд записи в памяти данных (Smem) с использованием косвенной адресации с длинным смещением.

Мнемоника

READA

Циклы

Циклы единичного выполнения с косвенной адресацией и длинным смещением

Операнд		Программа		
pmad	Smem	ROM	RAM	External
RAM	RAM	2	2,3†	2+2p, 3+2p#
	External	2+d	2+d	3+d+2p
	MMR	2	2	2+2p
PROM	RAM	3	2, 3†	2+2p
	External	3	2+d	3+d+2p
	MMR	3	2	2+2p
External	RAM	2+d	2+d,3+d†	3+d+2p
	External	3+2d	3+2d	4+2d+2p
	MMR	2+d	2+d	3+d+2p

Класс 26A

1 слово, 5 циклов. Операнд чтения в памяти данных (Smem) адрес записи в памяти программ (pmad).

Мнемоника

WRITA

Циклы

Циклы единичного выполнения

Операнд		Программа		
Smem	pmad	ROM	RAM	External
RAM	RAM	1	1,2†,3‡	1+p, 2+p#
	External	1+d	1+d	2+d+p
DROM	RAM	2	1, 2†	1+p
	External	2+d	1+d	2+d+p
External	RAM	1+d	1+d,2+d†	2+d+p
	External	2+2d	2+2d	3+2d+p
MMR	RAM	1	1,2†	1+p, 2+p#
	External	1+d	1+d	2+d+p

† операнд и код в одном блоке памяти.
‡ два операнда и код в одном блоке памяти.
# два операнда в одном блоке памяти.

Циклы повторяющихся выполнений

Операнд		Программа		
Smem	pmad	ROM	RAM	External
RAM	RAM	n	n,2n†,2n+1‡	n+p, 2n+p#
	External	n+nd	n+nd	n+nd+1+p
DROM	RAM	n+1	n, n+1†	n+p
	External	n+1	n+nd	n+nd+1+p
External	RAM	n+nd	n+nd,n+1+d†	n+nd+1+p
	External	2n+2nd	2n+2nd	2n+2nd+1+p
MMR	RAM	n	n,n+1†	n+p, n+1+p#
	External	n+nd	n+nd	n+nd+1+p

† операнд и код в одном блоке памяти.
‡ два операнда и код в одном блоке памяти.
# два операнда в одном блоке памяти.

Класс 26B

2 слова, 6 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†,4‡	2+2p, 3+2p#
	External	2+d	2+d	3+d+2p
DROM	RAM	3	2, 3†	2+2p
	External	3+d	2+d	3+d+2p
External	RAM	2+d	2+d,3+d†	3+d+2p
	External	3+2d	3+2d	4+2d+2p
MMR	RAM	2	2,3†	2+2p, 3+2p#
	External	2+d	2+d	3+d+2p

Класс 27A

2 слова, 2 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 27B

3 слова, 3 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 28A

2 слова, 2 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 28B

3 слова, 3 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 29A

2 слова, 4 цикла, 2 цикла (с задержкой), 2 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 29B

2 слова, 4 цикла, 2 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 30A

1 слово, 6 циклов, 4 цикла (с задержкой). Операнд регистра.

Мнемоника

BACC[D] FBACC[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 30B

1 слово, 6 циклов, 4 циклов (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 31A

2 слова, 5 цикла, 3 цикла (с задержкой). Операнд в памяти программ (pmad) и операнды короткого непосредственного значения

Мнесоника

BC[D]

Циклы

Циклы единичного выполнения

	Программа		
Условие	ROM	RAM	External
Истина	5	5	5+4p
Ложь	3	3	3+2p

Циклы единичного выполнения с задержкой

	Программа		
Условие	ROM	RAM	External
Истина	3	3	3+2p
Ложь	3	3	3+2p

Класс 31B

2 слова, 5 циклов, 3 цикла (с задержкой), 3 цикла (ложное условие). Операнд в памяти программ (pmad) и операнды с коротким непосредственным значением.

Мнемоника

CC[D]

Циклы

Циклы единичного выполнения (условие истинное)

Операнд	Программа		
Stack	ROM	RAM	External
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		
Stack	ROM	RAM	External
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		
Stack	ROM	RAM	External
RAM	3	3,4†	3+2p
External	3	3	6+2p+d

† Операнд и код в одном блоке памяти.

Класс 32

1 слово, 5 циклов, 3 цикла (с задержкой), 3 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

1 слово, 3 цикла, 1 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

1 слово, 6 циклов, 4 цикла (с задержкой). Операнд отсутствует.

Мнемоника

FRET[D] FRETE[D]

Циклы

В данной модели операции не реализованы.

Класс 35

1 слово, 3 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

1 слово, 4 цикла (минимум). Операнд с коротким непосредственным значением.

Мнемоника

IDLE

Циклы

Количество циклов, необходимое для выполнения этой команды, зависит от периода бездействия

17.3 Команды на языке ассемблера

Этот раздел содержит подробную информацию о наборе команд для DSP ядра микроконтроллера. Набор команд включает команды обработки сигналов, требующие большого объема вычислений и команды общего назначения, такие как мультиобработка и высокоскоростное управление.

17.3.1 Обозначения и сокращения

В разделе перечислены сокращения и обозначения, используемые в обзоре набора команд и в описании отдельных команд.

Таблица 17-29 – 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 ≤ x ≤ 7)
ARP	Указатель вспомогательного регистра (3-битное поле в ST0), указывает текущий AR.
ASM	5-разрядное поле режима сдвига аккумулятора в ST1 (–16 ≤ ASM ≤ 15)
B	Аккумулятор B
BRAF	Флаг активности повторения блока команд в ST1
BRC	Счетчик повторения блока программного кода
BITC	4-разрядное значение, определяющее бит ячейки памяти данных, тестируемый командой контроля битов (0 ≤ BITC≤ 15)
C16	16-разрядный бит режима двухсловной/двойной точности арифметики в ST1
C	Бит переноса в ST0
CC	2-разрядный код условия (0 ≤ CC ≤ 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 ≤ dmad ≤ 65 535)
Dmem	Операнд в памяти данных
DP	9-разрядное поле указателя страницы памяти данных в ST0 (0 ≤ DP ≤ 511)
dst	Аккумулятор приемник (A или B)
dst_	Противоположный аккумулятор: если dst = A, то dst_ = B если dst = B, то dst_ = A

Обозначение	Содержание
EAB	E адресная шина
EAR	EAB адресный регистр
extpmad	Непосредственный 23- разрядный адрес памяти программ
FRCT	Бит режима дробных чиселв 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	3-разрядное непосредственное значение (0 ≤ k3 ≤ 7)
k5	5-разрядное непосредственное значение (–16 ≤ k5 ≤ 15)
k9	9-разрядное непосредственное значение (0 ≤ k9 ≤ 511)
lk	16-битное длинное непосредственное значение
Lmem	32-разрядный операнд памяти данных, использующий адресацию длинного слова
mmr, MMR	Регистр, отображаемый на память
MMRx, MMRy	Регистр, отображаемый на память, AR0–AR7 или SP
n	Число слов, полученных а 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 ≤ PA ≤ 65 535)
PAR	Регистр адреса программы
PC	Счетчик программ
pmad	Непосредственный 16-разрядный адрес памяти программ (0 ≤ pmad ≤ 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	4-битное значение, которое определяет номер разряда регистра состояния, изменяемого в командах RSBX, SSBX, и XC (0 ≤ SBIT ≤ 15)
SHFT	4-разрядное значение сдвига (0 ≤ SHFT ≤ 15)
SHIFT	5-разрядное значение сдвига (0 ≤ SHFT ≤ 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 ≤ TS ≤ 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 17-30 – Обозначения и сокращения в коде операции

Символ	Значение
A	Бит адреса памяти данных
ARX	3-разрядное значение, которое определяет вспомогательный регистр
BITC	4-разрядный код номера разряда
CC	2-разрядный код состояния
CCCC CCCC	8-разрядный код состояния
COND	4-разрядный 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	4-битное значение, определяющее один из девяти отображаемых на память регистров (0 ≤ MMRX ≤ 8)
MMRY	4-битное значение, определяющее один из девяти отображаемых на память регистров (0 ≤ MMRY ≤ 8)
N	Одиночный бит
NN	2-битное значение, определяющее тип прерывания
R	Бит выбора округления (rnd) R = 0 команда выполняется без округления R = 1 округление результата
S	Бит аккумулятора источника (src) S = 0 аккумулятор A S = 1 аккумулятор B
SBIT	4-разрядный номер разряда статусного регистра

Символ	Значение
SHFT	4-разрядное значение сдвига (0 ≤ SHFT ≤ 15)
SHIFT	5-разрядное значение сдвига (–16 ≤ SHIFT ≤ 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 17-31 – 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 → y	Значение x присваивается регистру или ячейке y. Например: (Smem) → 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.
<< nn	Сдвиг влево битов nn (отрицательный или положительный)
||	Параллельная команда
\\	Циклически сдвинуть влево
//	Циклически сдвинуть вправо
$\overline{x}$	Логическая инверсия x (в двоичной системе)
| x |	Абсолютное значение x
AAh	Показывает, что AA представляет шестнадцатеричное число

Таблица 17-32 – Операторы, используемые в наборе команд

Символ	Операторы	Вычисление
+ – ~	унарный плюс, минус, обратный код	справа налево

Символ	Операторы	Вычисление
* / %	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

17.3.1.1 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 ≤ SHIFT ≤ 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

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

Код операции разделяет различные поля битов для формирования каждой из команд. См. раздел об определении символы кода операции.

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 1901ВЦ1Т 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.

17.3.2 ABDST Xmem, Ymem

Операнды Xmem, Ymem: операнды в 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	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

A	FF	ABCD	0000
B	00	0000	0000
AR3			0100
AR4			0200
FRCT			0

Память данных

0100h	0055
0200h	00AA

После исполнения

A	FF	FFAB	0000
B	00	0000	5433
AR3			0101
AR4			0201
FRCT			0

Память данных

0100h	0055
0200h	00AA

17.3.3 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	FF	FFFF	FFCB	-53
B	FF	FFFF	FC18	-1000	00	0000	0035	+53

Пример 2 ABS A

	Перед исполнением			После исполнения		
A	03	1234	5678	00	7FFF	FFFF
OVM			1			1

Пример 3 ABS A

	Перед исполнением			После исполнения		
A	03	1234	5678	00	7FFF	FFFF
OVM			0			0

17.3.4 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 ≤ lk ≤ 32 767
–16 ≤ SHIFT ≤ 15
0 ≤ SHFT ≤ 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

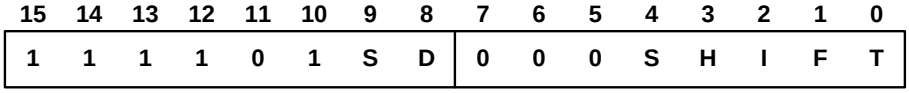
6:

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

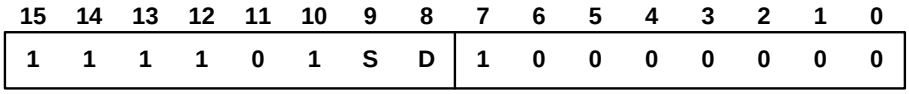
7:

8:

9:



10:



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 3A.
Синтаксисы 1, 2, и 3: класс 3.
Синтаксис 4: класс 4A.
Синтаксис 4: класс 4B.
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**	**00 0540 1200**
C	**1**	**01**
AR3	**0100**	**0101**
SXM	**1**	**1**
Память данных		
0100h	**1500**	**1500**

Пример 2 ADD A, –8, B

	Перед исполнением	После исполнения
A	00 0000 1200	00 0000 1200
B	00 0000 1800	00 0000 1812
C	1	0

Пример 3 ADD #4568, 8, A, B

	Перед исполнением	После исполнения
A	00 0000 1200	00 0000 1200
B	00 0000 1800	00 0045 7A00
C	1	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.

17.3.5 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	00 0000 0013	00 0000 0018
C	1	0
AR2	0100	0105
Память данных		
0105h	0004	0004

17.3.6 ADDM #lk, Smem

Операнды Smem: операнд памяти данных одиночного доступа.
–32 768 ≤ lk ≤ 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+

	Перед командой	После команды
AR4	0100	0101
Память данных		
0100h	0004	132F

Пример 2 ADDM 0FFF8h, *AR4+

	Перед исполнением	После исполнения
OVM	1	1
SXM	1	1
AR4	0100	0101
Память данных		
0100h	8007	8000

17.3.7 ADDS 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 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

	Перед исполнением	После исполнения
B	00 0000 0003	00 0000 F009
C	x	0
AR2	0100	00FF
Память данных		
0104h	F006	F006

17.3.8 AND

1: **AND** Smem, src
2: **AND** #lk [, SHFT], src [, dst]
3: **AND** #lk, 16, src [, dst]
4: **AND** src [, SHIFT], [, dst]

Операнды

Smem: операнд в памяти данных одиночного доступа
src: A (аккумулятор A)
B (аккумулятор B)
–16 ≤ SHIFT ≤ 15
0 ≤ SHFT ≤ 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	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	0	1	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	0	1	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	0	0	S	H	I	F	T

Выполнение

1: (Smem) AND (src) → src
2: lk << SHFT AND (src)→ dst
3: lk << 16 AND (src)→ dst
4: (dst) AND (src) << SHIFT → dst

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 41 слово.
Синтаксис 2 и 32 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 3A.
Синтаксис 1: класс 3B.
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

17.3.9 ANDM #lk, Smem

Операнды Smem: операнд в 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	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	0101
Память данных		
0100h	0444	0044

Пример 2 ANDM #0101h, 4; DP = 0

	Перед исполнением	После исполнения
Память данных		
0004h	00 0000 0100	00 0000 0100

17.3.10 B[D] pmad

Операнды 0 ≤ pmad ≤ 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 B 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.

17.3.11 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.

17.3.12 BANZ[D] pmad, Sind

Операнды Sind: операнд косвенной адресации одиночного доступа.
0 ≤ pmad ≤ 65 535

Код операции

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.

17.3.13 BC[D] pmad, cond [, cond [, cond]]

Операнды

0 ≤ pmad ≤ 65 535

В таблице ниже приводятся условия для данной команды (операнд условия).

Операнд условия	Описание	Код условия	Операнд условия	Описание	Код условия
BIO	$\overline{\text{BIO}}$ low	0000 0011	NBIO	$\overline{\text{BIO}}$ 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) ≠ 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
0	1	1	1	1	0	Z	0	C	C	C	C	C	C	C	C
16 битная константа															

Выполнение

If (cond(s))
Then
pmad → PC
Else
(PC) + 2 → 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 задержаный (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 A или B); 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 A, B, или C); 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 A	Категория B	Категория A	Категория B	Категория C
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 31A.

Пример 1 BC 2000h, AGT

	Перед выполнением				После выполнения			
A	**00**	**0000**	**0053**		**A**	**00**	**0000**	**0053**
PC			**1000**		**PC**			**2000**

Пример 2 BC 2000h, AGT

	Перед выполнением				После выполнения			
A	**FF**	**FFFF**	**FFFF**		**A**	**FF**	**FFFF**	**FFFF**
PC			**1000**		**PC**			**1002**

Пример 3

BCD 1000h, BOV
ANDM 4444h, *AR1+

	Перед выполнением		После выполнения
PC	3000	PC	1000
OVB	1	OVB	1

После того, как в ячейку памяти добавляется (AND) значение 4444h, переход осуществляется при выполнении условия (OVB). 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

17.3.14 BIT Xmem, BITC

Операнды Xmem: операнд в памяти данных двойного доступа
0 ≤ BITC ≤ 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, соответсвующие каждому биту в памяти данных.
Код бита соответствует BITC , а адрес бита соответсвует (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	0101
TC	0	1
Память данных		
0100h	7688	7688

17.3.15 BITF Smem, #lk

Операнды Smem: операнд в 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	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	0
DP	004	004
Память данных		
0205h	5400	5400

Пример 2 BITF 5, 0800h

	Перед исполнением	После исполнения
TC	x	0
DP	004	004
Память данных		
0205h	0F7F	0F7F

17.3.16 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 соответсвует (15 – T(3–0)). Код бита соответсвует 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 3A.
Класс 3B.

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

17.3.17 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 двухслолвная 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 30B.

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	00 0000 2000
PC	0025	2000
SP	1111	1110
Память данных		
1110h	4567	0028

После того, как содержимое ячейки памяти логически умножается (AND) со значением 4444h, программа продолжает выполнение с адреса 2000h.

17.3.18 CALL[D] pmad

Операнды 0 ≤ pmad ≤ 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 задержики.

(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).

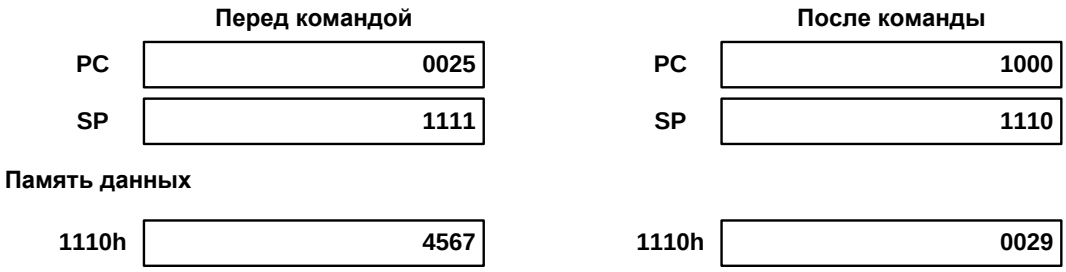
Классы Класс 29B.

Пример 1 CALL 3333h

	Перед командой	После команды
PC	0025	3333
SP	1111	1110
Память данных		
1110h	4567	0027

Пример 2

```
CALLD 1000h
ANDM #4444h, *AR1+
```



После того, как в ячейка памяти логически умножается (AND) со значением 4444h, программа продолжает выполнение с адреса 1000h.

17.3.19 CC[D] pmad, cond [, cond [, cond]]

Операнды

0 ≤ pmad ≤ 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) ≠ 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	0	Z	1	C	C	C	C	C	C	C	C
16 битная константа															

Выполнение

Без задержки:

```
If (cond(s))
Then
        (SP) - 1 → SP
        (PC) + 2 → TOS
        pmad → PC
Else
        (PC) + 2 → PC
```

С задержкой:

```
If (cond(s))
Then
        (SP) - 1 → SP
        (PC) + 4 → TOS
        pmad → PC
Else
        (PC) + 2 → 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 pmad. 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 A или B); 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 A, B, или C); 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 A	Категория B	Категория A	Категория B	Категория C
EQ NEQ LT LEQ GT GEO	OV NOV	TC NTC	C NC	BIO NBIO

Примечание – Данная команда не повторяемая.

Слова 2 слова.

Циклы 5 циклов (истинное условие).
3 цикла (ложное условие).
3 цикла (с задержкой).

Классы Класс 31B.

Пример 1 CC 2222h, AGT

	Перед командой	После команды
A	00 0000 3000	00 0000 3000
PC	0025	2222
SP	1111	1110
Память данных		
1110h	4567	0027

Пример 2

CCD 1000h, BOV
ANDM 4444h, *AR1+

	Перед командой	После команды
PC	0025	1000
OVB	1	1
SP	1111	1110
Память данных		
1110h	4567	0029

После того, как ячейка памяти логически умножается (AND) на значение 4444h, программа продолжает выполнение с ячейки 1000h.

17.3.20 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 $(\overline{src}) \rightarrow dst$

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		
A	FC	DFFA	AEAA
B	00	0000	7899

	После исполнения		
A	FC	DFFA	AEAA
B	03	2005	5155

17.3.21 CMPM Smem, #lk

Операнды Smem: операнд памяти данных одиночного доступа.
–32 768 ≤ lk ≤ 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 → TC
Else
0 → TC

Статусные биты Команда влияет на значение TC.

Описание Эта команда сравнивает 16-разрядный операнд памяти данных одиночного доступа Smem с 16-разрядной константой constant lk . если они равны, TC принимает значение 1. В противном случае, значение TC 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 CMPM *AR4+, 0404h

	Перед исполнением		После исполнения
TC	1	TC	1000
AR4	0100	AR4	1110
Память данных			
0100h	4444	0100h	4444

17.3.22 CMPR CC, ARx

Операнды 0 ≤ CC ≤ 3
ARxAR0–AR7

Код операции

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 → TC
Else
0 → 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.

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

17.3.23 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 10A.
Класс 10B.

Пример

CMPS A, *AR4+

	Перед исполнением	После исполнения
A	00 2345 7899	00 2345 7899
TC	0	1
AR4	0100	0101
TRN	4444	8889
Память данных		
0100h	0000	7899

17.3.24 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:

1. 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.
2. 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 9A.
Класс 9B.

Пример 1 DADD *AR3+, A, B

	Перед исполнением	После исполнения
A	00 5678 8933	00 5678 8933
B	00 0000 0000	00 6BAC BD89
C16	0	0
AR3	0100	0201†
Память данных		
0100h	1534	1534
0101h	3456	3456

† 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	00 5678 3933
B	00 0000 0000	00 6BAC 6D89
C16	1	1
AR3	0100	00FE†
Память данных		
0100h	1534	1534
0101h	3456	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	00 5678 3933
B	00 0000 0000	00 8ACE 4E67
C16	0	0
AR3	0101	00FF†
Память данных		
0100h	1534	1534
0101h	3456	3456

† данная команда является командой с длинным операндом, и потому после выполнения AR3 уменьшается на 2.

17.3.25 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) → dst(39–16)
(Lmem(15–0)) - (T) → dst(15–0)
Else
(Lmem) + ((T) + (T) << 16) → 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:

1. Если C16 = 0, команда выполняется в режиме двойной точности. Lmem добавляется к 32-разрядному значению, состоящему из содержимого T сочленённому с содержимым 16 битов того же регистра T сдвинутого влево на 16 разрядов (T <<16 + T). Результат сохраняется в dst.
2. 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	00 3879 579B
T	2345	2345
C16	0	0
AR3	0100	AR3† 0102

Память данных

0100h	1534	1534
0101h	3456	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	00 3879 579B
T	2345	2345
C16	0	0
AR3	0100	AR3† 0102

Память данных

0100h	1534	1534
0101h	3456	3456

† данная команда использует длинный операнд и после выполнения AR3 увеличивается на 2.

17.3.26 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	0100
Память данных		
0100h	6CAC	6CAC
0101h	0000	6CAC

17.3.27 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:

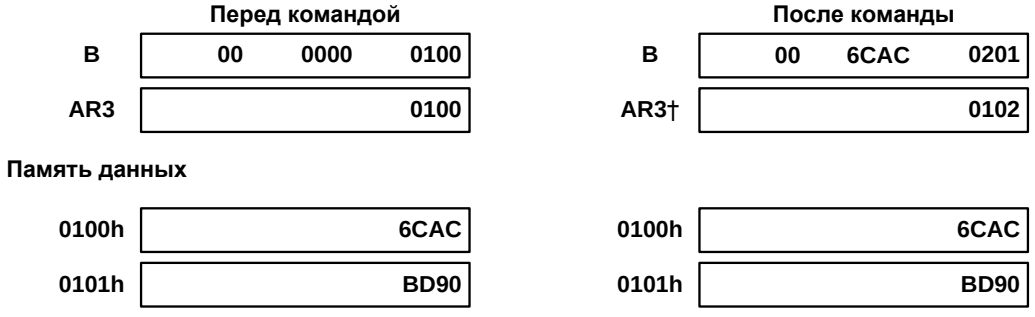
1. Если C16 = 0, команда выполняется в режиме двойной точности. Lmem загружается в dst.
2. Если C16 = 1, команды выполняется в двойном 16-разрядном режиме. 16 самых старших битов Lmem загружаются в старшие 24 бита dst. В то же время, 16 самых младших битов Lmem загружаются в 16 младших битов dst.

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 9A.
Класс 9B.

Пример DLD *AR3+, B



† данная команда является командой с чтением длинного операнда, потому после выполнения AR3 увеличивается на 2.

17.3.28 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:

1. Если C16 = 0, команда выполняется в режиме двойной точности. Содержимое src (32 бита) вычитается из Lmem. Результат сохраняется в src.
2. 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	BEBB	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			1		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.

17.3.29 DSADT 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	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:

1. Если C16 = 0, команда выполняется в режиме двойной точности. 32-разрядное значение, состоящее из содержимого T, сочленённому с содержимым того же регистра T сдвинутого влево на 16 разрядов (T <<16 + T) вычитается из Lmem. Результат сохраняется в dst.
2. 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 9A.
Класс 9B.

Пример 1 DSADT *AR3+, A

	Перед исполнением	После исполнения
A	00 0000 0000	FF F1EF 1111
T	2345	2345
C	0	0
C16	0	0
AR3	0100	AR3† 0102
Память данных		
0100h	1534	0100h 1534
0101h	3456	0101h 3456

† 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	FF F1EF 1111
T	2345	2345
C	0	1
C16	1	1
AR3	0100	AR3† 00FE
Память данных		
0100h	1534	0100h 1534
0101h	3456	0101h 3456

† команда является командой обработки длинного операнда и после выполнения AR3 уменьшается на 2.

17.3.30 DST src, Lmem

Операнды src: A (аккумулятор A).
B (аккумулятор B).
Lmemоперанд 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 13A.
Класс 13B.

Пример 1 DST B, *AR3+

	Перед командой	После команды
B	00 6CAC BD90	00 6CAC BD90
AR3	0100	AR3† 0102

Память данных

0100h	0000	6CAC
0101h	0000	BD90

† данная команда является командой обработки длинного операнда и после выполнения AR3 увеличивается на 2.

Пример 2 DST B, *AR3–

	Перед командой	После команды
B	00 6CAC BD90	00 6CAC BD90
AR3	0101	AR3† 00FF

Память данных

0100h	0000	BD90
0101h	0000	6CAC

† данная команда является командой обработки длинного операнда и после выполнения AR3 увеличивается на 2.

17.3.31 DSUB 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	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:

1. если C16 = 0, команда выполняется в режиме двойной точности. Lmem вычитается из содержимого src.
2. 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	00 4144 54DD
C16	0	0
AR3	0100	AR3† 0102
Память данных		
0100h	1534	1534
0101h	3456	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			1	C			1
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 увеличивается на 2.

17.3.32 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) → 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).
Инструкция влияет на C и OVdst.

Описание Данная команда вычитает содержимое T из 32-разрядного операнда Lmem и

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 DSUBT *AR3+, A

A	00	0000	0000	FF	F1EF	1111
T			2345			2345
C16			0			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	FF	F1EF	1111
T			2345			2345
C16			1			1
AR3			0100			00FE
Память данных						
0100h			1534			1534
0101h			3456			3456

† поскольку данная команда является командой с длинным операндом, после выполнения AR3 уменьшается на 2.

17.3.33 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	FF	FFFF	FFCB	-53
T			FC18				0019	25

Пример 2 EXP B

	Перед исполнением			После исполнения			
B	07	8543	2105	07	8543	2105	
T			FFFC			FFFC	-4

† значение аккумулятора B среди «сторожевых» битов имеет значимые разряды, что приводит к отрицательному показателю.

17.3.34 FB[D] extpmad

Операнды 0 ≤ extpmad ≤ 7F 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	2000
XPC	00	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	1000
XPC	00	7F

После того, как в операнд логически умножается (AND) на значение 4444h, программа продолжает выполнение с ячейки 1000h на странице 7Fh

17.3.35 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-разрадный 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 цикла.
4 цикла (с задержкой).

Классы Класс 30A.

Пример 1 FBACC A

	Перед командой	После команды
A	00 0001 3000	00 0001 3000
PC	1000	3000
XPC	00	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	00 007F 2000
XPC	01	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.

17.3.36 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-разрадный 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 цикла.
4 цикла (с задержкой).

Классы Класс 30B.

Пример 1 FCALA A

	Перед выполнением	После выполнения
A	00 007F 3000	00 007F 3000
PC	0025	3000
XPC	00	7F
SP	1111	110F

Память данных

1110h	4567	0026
110Fh	4567	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.

17.3.37 FCALL[D] extpmad

Операнды 0 ≤ extpmad ≤ 7F 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	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 29B.

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.

17.3.38 FIRS Xmem, Ymem, pmad

Операнды Xmem, Ymem: операнды памяти данных двойного доступа.
0 ≤ pmad ≤ 65 535

Код операции

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 увелисивается 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	00 00FF 0000
B	00 0000 0000	00 0008 762C
FRCT	0	0
AR3	0100	0101
AR4	0200	0201
Память данных		
0100h	0055	0055
0200h	00AA	00AA
Память программ		
COEFFS	1234	1534

17.3.39 FRAME K

Операнды –128 ≤ K ≤ 127

Код операции

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 → 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

17.3.40 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 цикла.
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

17.3.41 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 цикла.
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

17.3.42 IDLE K

Операнды $1 \le K \le 3$

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	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:

1. **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.
2. **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.
3. **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.

17.3.43 INTR K

Операнды $0 \le K \le 31$

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 → SP
(PC) + 1 → TOS
Вектор прерывания определяемый K → PC
1 → INTM

Статусные биты На команду влияет INTM и IFR.

Описание Данная команда осуществляет передачу управления на программу обработки прерывания по вектору, определяемому значением K. Эта команда позволяет использовать прикладную программу по обработке прерывания. Перечень прерываний и соответствующих значений K представлен в Приложении В.
Во время выполнения команды PC увеличивается на 1 и записывается в TOS. Таким образом, вектор прерывания, определенный значением K, 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

17.3.44 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 ≤ K ≤ 255
–32 768 ≤ lk ≤ 32 767
–16 ≤ SHIFT ≤ 15
0 ≤ SHFT≤ 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 различный синтаксис в определенных условиях.

1. Синтаксис 4: если SHIFT = 0, код операции команды ассемблируется по синтаксису 1.
2. Синтаксис 4: если 0 < SHIFT ≤ 15 и режим косвенной адерсации 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 7если SHFT = 0 и 0 ≤ lk ≤ 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 3A.
Синтаксисы 1, 2 и 3: Класс 3B.
Синтаксис 4: Класс 4A.
Синтаксис 4: Класс 4B.
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	00 0000 FEDC
SXM	0	0
AR1	0200	0200
Память данных		
0200h	FEDC	FEDC

Пример 2 LD *AR1, A

	Перед командой	После команды
A	00 0000 0000	FF FFFF FEDC
SXM	1	1
AR1	0200	0200
Память данных		
0200h	FEDC	FEDC

Пример 3 LD *AR1, TS, B

	Перед командой	После команды
B	00 0000 0000	FF FFFE DC00
SXM	1	1
AR1	0200	0200
T	8	8
Память данных		
0200h	FEDC	FEDC

Пример 4 LD *AR3+, 16, A

	Перед командой	После команды
B	00 0000 0000	FF FEDC 0000
SXM	1	1
AR3	0300	0301
Память данных		
0300h	FEDC	FEDC

Пример 5 LD #248, B

	Перед командой	После команды
B	00 0000 0000	00 0000 00F8
SXM	1	1

Пример 6 LD A, 8, B

	Перед командой	После команды
A	00 7FFD 0040	00 7FF0 0040
B	00 0000 FFFF	7F FD00 4000
OVB	0	1
SXM	1	1
Память данных		
0200h	FEDC	FEDC

17.3.45 LDM MMR, dst

Операнды MMR: отображаемый в памяти регистр.
dst: A (аккумулятор).
B (аккумулятор).

Код операции

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 DP и старших девяти бит ARx. Данная команда не зависит от значения SXM.

Слова 1 слова.

Циклы 1 цикл.

Классы Класс 3A.

Пример 1 LDM AR4, A

	Перед командой			После команды		
A	00	0000	1111	00	0000	FFFF
AR4			FFFF			FFFF

Пример 2 LDM 060h, B

	Перед командой			После команды		
B	00	0000	0000	00	0000	1234

Память данных

0060h			1234	0060h		1234

17.3.46 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			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
||MACR *AR5+, B
```

	Перед командой	После команды
A	00 0000 1000	00 1234 0000
B	00 0000 1111	00 010D 0000
T	0400	0400
FRCT	0	0
AR4	0100	0101
AR5	0200	0201
Память данных		
0100h	1234	1234
0200h	4321	4321

17.3.47 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

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
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, вычитает результат умножения из dst_ и сохраняет результат в dst_.
При использовании индекса R, эта команда округляет результат умножения с вычитанем 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	00 1234 0000
B	00 0000 1111	FF FEF3 8D11
T	0400	0400
FRCT	0	0
AR4	0100	0101
AR5	0200	0201

Память данных

0100h	1234	1234
0200h	4321	4321

Пример 2

```
LD *AR4+, A
||MASR *AR5+, B
```

	Перед командой	После команды
A	00 0000 1000	00 1234 0000
B	00 0000 1111	FF FEF4 0000
T	0400	0400
FRCT	0	0
AR4	0100	0101
AR5	0200	0201

Память данных

0100h	1234	1234
0200h	4321	4321

17.3.48 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	После команды
A	00 0000 0000	00 FEDC 8000
SXM	0	0
AR1	0200	0200
Память данных		
0200h	FEDC	FEDC

17.3.49 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.

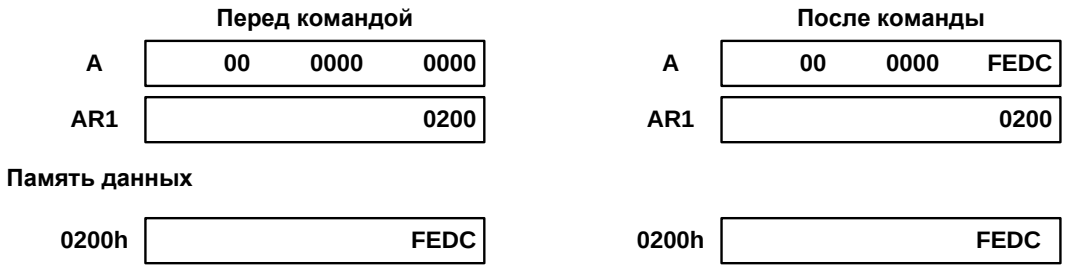
Описание
Данная команда загружает значение Smem в младшую часть dst (биты 15–0). Сторожевые биты и старшая часть dst (биты 39–16) сбрасываются в 0. Данные обрабатываются как 16-разрядное число без знака. Знак не расширяется независимо от статуса бита SXM.

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 3A.
Класс 3B.

Пример
LDU *AR1, A



17.3.50 LMS 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	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	00 77CD 0888
B	00 0000 0100	00 0000 3972
FRCT	0	0
AR3	0100	0101
AR4	0200	0201

Память данных

0100h	0055	0055
0200h	00AA	00AA

17.3.51 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 24A.
Класс 24B.

Пример LTD *AR3

	Перед командой	После команды
T	0000	6CAC
AR3	0100	0100

Память данных

0100h	6CAC	6CAC
0101h	xxxx	6CAC

17.3.52 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 ≤ lk ≤ 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) × (T) + (src) → src
2: (Xmem) × (Ymem) + (src) → dst
(Xmem) → T
3: (T) × lk + (src) → dst
4: (Smem) × lk + (src) → dst
(Smem) → 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 3A.
Синтаксис 1: Класс 3B.
Синтаксис 2: Класс 7.
Синтаксис 3: Класс 2.
Синтаксис 4: Класс 6A.
Синтаксис 4: Класс 6B.

Пример 1 MAC *AR5+, A

	Перед командой	После команды
A	00 0000 1000	00 0048 E000
T	0400	0400
FRCT	0	0
AR5	0100	0101
Память данных		
0100h	1234	1234

Пример 2 MAC #345h, A, B

	Перед командой	После команды
A	00 0000 1000	00 0000 1000
B	00 0000 0000	00 001A 3800
T	0400	4000
FRCT	1	1

Пример 3 MAC *AR5+, #1234h, A

	Перед командой	После команды
A	00 0000 1000	00 0626 1060
T	0000	5678
FRCT	0	0
AR5	0100	0101
Память данных		
0100h	5678	5678

Пример 4 MAC *AR5+, *AR6+,A, B

	Перед командой	После команды
A	00 0000 1000	00 0000 1000
B	00 0000 0004	00 0C4C 10C0
T	0008	5678
FRCT	1	1
AR5	0100	0101
AR6	0200	0201

Память данных

0100h	5678	5678
0200h	1234	1234

Пример 5 MACR *AR5+, A

	Перед командой	После команды
A	00 0000 1000	00 0049 0000
T	0400	0400
FRCT	0	0
AR5	0100	0101

Память данных

0100h	1234	1234

Пример 6 MACR *AR5+, *AR6+,A, B

	Перед командой	После команды
A	00 0000 1000	00 0000 1000
B	00 0000 0004	00 0C4C 0000
T	0008	5678
FRCT	1	1
AR5	0100	0101
AR6	0200	0201

Память данных

0100h	5678	5678
0200h	1234	1234

17.3.53 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 A (биты 32–16) на значение операнда Smem или на содержимое T, добавляет результат к аккумулятору B (синтаксис 1) или к src. Результат сохраняется в аккумулятор B (синтаксис 1) или в dst или src если не указан dst. A(32–16) используется в множителе в качестве 17-разрядного операнда.
При использовании индекса R, эта команда округляет результат умножения аккумулятора A 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 3A.
Синтаксисы 1 и 2: Класс 3B.
Синтаксисы 3 и 4: Класс 1.

Пример 1 MACA *AR5+

	Перед командой	После команды
A	00 1234 0000	00 1234 0000
B	00 0000 0000	00 0626 0060
T	0400	5678
FRCT	0	0
AR5	0100	0101

Память данных

0100h	5678	5678

Пример 2 MACA T, B, B

	Перед командой	После команды
A	00 1234 0000	00 1234 0000
B	00 0002 0000	00 009D 4BA0
T	0444	0444
FRCT	1	1

Пример 3 MACAR *AR5+, B

	Перед командой	После команды
A	00 1234 0000	00 1234 0000
B	00 0000 0000	00 0626 0000
T	0400	5678
FRCT	0	0
AR5	0100	0101

Память данных

0100h	5678	5678

Пример 4 MACAR T, B, B

	Перед командой	После команды
A	00 1234 0000	00 1234 0000
B	00 0002 0000	00 009D 0000
T	0444	0444
FRCT	1	1

17.3.54 MACD Smem, pmad, src

Операнды

Smem: операнд памяти данных одиночного доступа
src: A (аккумулятор A)
B (аккумулятор B)
0 ≤ pmad ≤ 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 23A.
Класс 23B.

Пример MACD *AR3–, COEFFS, A

	Перед командой			После команды		
A	**00**	**0077**	**0000**	**A**	**00**	**007D**	**0B44**
T			**0008**	**T**			**0055**
FRCT			**1**	**FRCT**			**0**
AR5			**0100**	**AR5**			**00FF**
Память программ							
COEFFS			**1234**	**COEFFS**			**1234**
Память данных							
0100h			**0055**	**0100h**			**0055**
0200h			**0066**	**0200h**			**0055**

17.3.55 MACP Smem, pmad, src

Операнды

Smem: операнд памяти данных одиночного доступа
src: A (аккумулятор A).
B (аккумулятор B).
0 ≤ pmad ≤ 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	0	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) × (Pmem адресованный через PAR) + (src) → src
(Smem) → T
(Smem) → Smem + 1

Статусные биты

Результат 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 адрес, следующий 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.

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 23A.
Класс 23B.

Пример MACD *AR3–, COEFFS, A

	Перед командой	После команды
A	00 0077 0000	00 007D 0B44
T	0008	0055
FRCT	1	0
AR3	0100	00FF
Память программ		
COEFFS	1234	1234
Память данных		
0100h	0055	0055
0101h	0066	0055

17.3.56 MACSU Xmem, Ymem, src

Операнды Xmem, Ymem: операнды памяти данных двойного 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
1	0	1	0	0	1	1	D	X	X	X	X	Y	Y	Y	Y

Выполнение unsigned(Xmem) × signed(Ymem) + (src) → src
(Xmem) → T

Статусные биты Результат изменяется под влиянием FRCT и OVM.
Команда влияет на OVsrc.

Описание Данная команда умножает значение без знака памяти данных Xmem на значение памяти данных Ymem со знаком, складывает произведение с src, и сохраняет результат в src. 16-разрядное значение Xmem без знака сохраняется в T. T обновляется значением Xmem без знака в фазе чтения.
Данные, адресованные Xmem, передаются по шине D. Данные, адресованные Ymem, передаются по шине C.

Слова 1 слово.

Циклы 1 цикл.

Классы Класс 7.

Пример MACSU *AR4+, *AR5+, A

	Перед командой	После команды
A	00 0000 1000	00 09A0 AA84
T	0008	8765
FRCT	0	0
AR4	0100	0101
AR5	0200	0201

Память данных

0100h	8765	5678
0200h	1234	00AA

17.3.57 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

17.3.58 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.
Команда влияет на OVdst (или OVsrc, если dst = src).

Описание
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 3A.
Синтаксис 1: Класс 3B.
Синтаксис 2: Класс 7.

Пример 1 MAS *AR5+, A

	Перед командой	После команды
A	00 0000 1000	FF FFB7 4000
T	0400	0400
FRCT	0	0
AR5	0100	0101
Память данных		
0100h	1234	1234

Пример 2 MAS *AR5+, *AR6+, A, B

	Перед командой	После команды
A	00 0000 1000	00 0000 1000
B	00 0000 0004	FF F9DA 0FA0
T	0008	5678
FRCT	1	1
AR5	0100	0101
AR6	0200	0201
Память данных		
0100h	5678	5678
0200h	1234	1234

Пример 3 MASR *AR5+, A

	Перед командой	После команды
A	00 0000 1000	FF FFB7 0000
T	0400	0400
FRCT	0	0
AR5	0100	0101
Память данных		
0100h	1234	1234

Пример 4 MASR *AR5+, *AR6+, A, B

	Перед командой	После команды
A	00 0000 1000	00 0000 1000
B	00 0000 0004	FF F9DA 0000
T	0008	5678
FRCT	1	1
AR5	0100	0101
AR6	0200	0201

Память данных

0100h	5678	5678
0200h	1234	1234

17.3.59 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.
Команда влияет на OVdst (или OVsrc, если dst не указан) и OVB в синтаксисе 1.

Описание
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 2^{15} 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	00 1234 0000
B	00 0002 0000	FF F9DB FFA0
T	0400	5678
FRCT	0	1
AR5	0100	0101

Память данных

0100h	5678	5678

Пример 2 MASA T, B

	Перед командой	После команды
A	00 1234 0000	00 1234 0000
B	00 0002 0000	FF FF66 B460
T	0444	0444
FRCT	1	1

Пример 3 MASA T, B

	Перед командой	После команды
A	00 1234 0000	00 1234 0000
B	00 0002 0000	FF FF67 0000
T	0444	0444
FRCT	1	1

17.3.60 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	FFF6	-10
B	FFCB	-53	0035	-53
C	1		0	

Пример 2 MAX A

	Перед командой	После команды
A	00 0000 0055	00 0000 1234
B	00 0000 1234	00 0000 1234
C	0	1

17.3.61 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

17.3.62 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 ≤ lk ≤ 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	0	0	R	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	0	1	0	0	1	0	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
0	1	1	0	0	0	1	D	I	A	A	A	A	A	A	A
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	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 3A.
Синтаксис 1: Класс 3B.
Синтаксис 2: Класс 7.
Синтаксис 3: Класс 6A.
Синтаксис 3: Класс 6B.
Синтаксис 4: Класс 2.

Пример 1 MPY 13, A

	Перед командой	После команды
A	00 0000 0036	00 0000 0054
T	0036	0006
FRCT	1	1
DP	008	008
Память данных		
040Dh	0007	0007

Пример 2 MPY *AR2–, *AR4+0%, B;

	Перед командой	После команды
B	FF FFFF FFE0	00 0000 0020
FRCT	0	0
AR0	0001	0001
AR2	01FF	01FE
AR4	0300	0301
Память данных		
01FFh	0010	0010
0300h	0002	0002

Пример 3 MPY #0FFFEh, A

	Перед командой	После команды
A	00 0000 1234	FF FFFF C000
T	2000	2000
FRCT	0	0

Пример 4 MPYR 0, B

	Перед командой			После команды		
B	FF	FE00	0001	00	0626	0000
T			1234			1234
FRCT			0			0
DP			004			004
Память программ						
0200h			5678			5678

17.3.63 MPYA

1: **MPYA** Smem
2: **MPYA** 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 содерджимое 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	После команды
A	FF 8765 1111	FF 8765 1111
B	00 0000 0320	FF D743 6558
T	1234	5678
FRCT	0	0
AR2	0200	0200

Память данных

0200h	5678	5678

Пример 2 MPYA B

	Перед командой		
A	**FF**	**8765**	**1111**
B	**00**	**0000**	**0320**
T			**4567**
FRCT			**0**

	После команды		
A	**FF**	**8765**	**1111**
B	**FF**	**DF4D**	**B2A3**
T			**4567**
FRCT			**0**

17.3.64 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 17x17-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 3A.
Класс 3B.

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

17.3.65 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

17.3.66 MVDK Smem, dmad

Операнды Smem: операнд памяти данных одиночного доступа.
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	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 19A.
Класс 19B.

Пример 1 MVDK 10, 8000h

	Перед командой	После команды
DP	004	004
Память данных		
020Ah	1234	1234
8000h	ABCD	1234

Пример 2 MVDK *AR3–, 1000h

	Перед командой	После команды
AR3	01FF	01FE
Память данных		
1000h	ABCD	1234
01FFh	1234	1234

17.3.67 MVDM dmad, MMR

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	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 19A.

Пример MVDM 300h, BK

	Перед командой	После команды
BK	ABCD	1234
Память данных		
0300h	1234	1234

17.3.68 MVDP Smem, pmad

Операнды Smem: операнд памяти данных одиночного доступа.
0 ≤ pmad ≤ 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	1	0	1	I	A	A	A	A	A	A	A
16 битная константа															

Выполнение
pmad → PAR
If (RC) ≠ 0
Then
(Smem) → Pmem addressed by PAR
(PAR) + 1 → PAR
Else
(Smem) → 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-разнрядным 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	004
Память программ		
0200h	0123	0123
Память данных		
FE00h	FFFF	0123

17.3.69 MVKD dmad, Smem

Операнды Smem: операнд памяти данных одиночного доступа.
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	0	0	I	A	A	A	A	A	A	A
16 битная константа															

Выполнение
dmad → DAR
If (RC) ≠ 0
Then
(Dmem addressed by DAR) → Smem
(DAR) + 1 → DAR
Else
(Dmem addressed by DAR) → 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 19A.
Класс 19B.

Пример 1 MVKD 300h, 0

	Перед командой	После команды
DP	004	004
Память данных		
0200h	ABCD	1234
0300h	1234	1234

Пример 2 MVKD 1000h, *+AR5

	Перед командой	После команды
AR5	01FF	0200
Память данных		
1000h	1234	1234
0200h	ABCD	1234

17.3.70 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 19A.

Пример MVMD AR7, 8000h

	Перед командой	После команды
AR7	1234	1234
Память данных		
8000h	ABCD	1234

17.3.71 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	Регистр	MMRX/MMRY
AR0	0000	AR5	0101
AR1	0001	AR6	0110
AR2	0010	AR7	0111
AR3	0011	SP	1000
AR4	0100		

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

17.3.72 MVPD pmad, Smem

Операнды Smem: операнд памяти данных одиночного доступа.
0 ≤ pmad ≤ 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	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

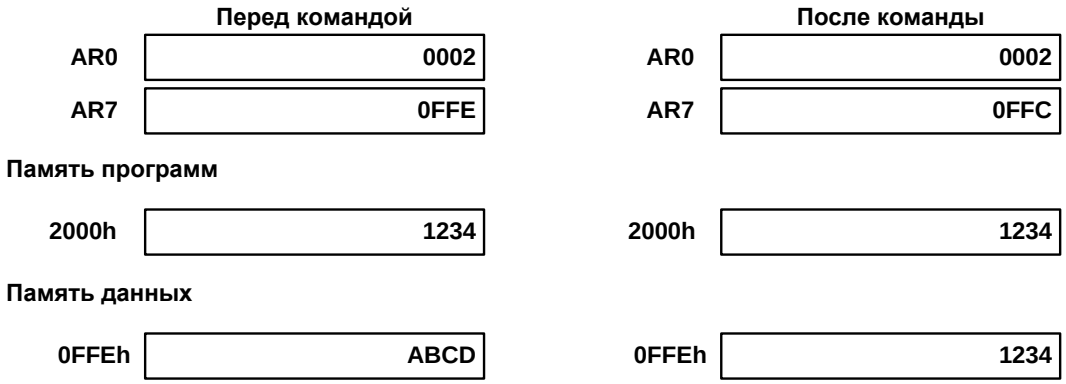
Циклы 3 цикла.
Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Классы Класс 21A.
Класс 21B.

Пример 1 MVPD 0FE00h, 5

	Перед командой	После команды
DP	006	006
Память программ		
FE00h	8A55	8A55
Память данных		
0305h	FFFF	8A55

Пример 2 MVPD 2000h, *AR7–0



17.3.73 NEG 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	0	1	0	0

Выполнение (src) * –1 → dst

Статусные биты Команда под влиянием OVM.
Результат влияет на C и OVdst (или OVsrc, если dst = src).

Описание Данная команда расчитывает дополнительный код (дополнение до двух) содержимого src (также A или B) и сохраняет результат в dst или src, если dst не указан. Данная команда сбрасывает значение бита переноса, C, 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, B

A	FF FFFF F228	A	FF FFFF F228
B	00 0000 1234	B	00 0000 0DD8
OVA		OVA	

Пример 2 NEG B, A

	Перед командой		После команды	
A	00 0000 1234	A	FF 8000 0000
B	00 8000 0000	B	00 8000 0000
OVA	0	OVA	0

Пример 3 NEG A

	Перед командой		После команды	
A	80 0000 0000	A	80 8000 0000
OVA	1	OVA	1
OVM	0	OVM	0

Пример 4 NEG A

	Перед командой	После команды
A	80 0000 0000	00 7FFF FFFF
OVA	0	1
OVM	1	1

17.3.74 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 PC. Это может быть полезным при создании в конвейере задержек выполнения.

Слова 1 слово.

Циклы 1 цикл.

Классы Класс 1.

Пример NOP
Действия отсутствуют.

17.3.75 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

17.3.76 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 созраняется в dst или src, если dst не указан. По предписанию команды значения могут быть сдвинуты. При положительных сдвигах (влево) значения младших разрядов

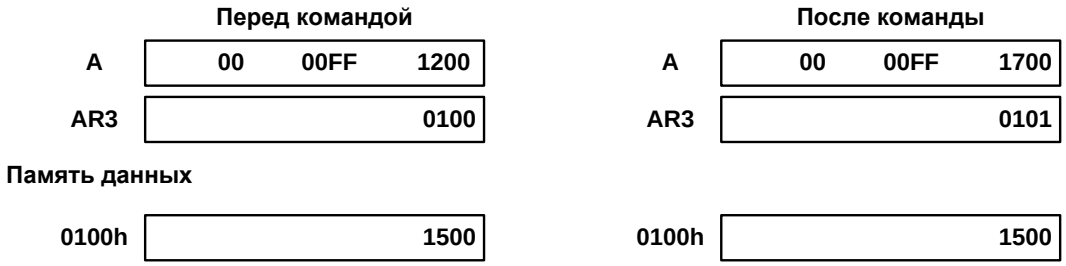
обнуляются, а значения старших разрядов не дополняются знаком. Для отрицательных сдвигах (вправо) значения старших разрядов не дополняются знаком.

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 3A.
Синтаксис 1: Класс 3B.
Синтаксисы 2 и 3: Класс 2.
Синтаксис 4: Класс 1.

Пример 1 OR *AR3+, A



Пример 2 OR A, +3, B

17.3.77 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 18A.
Класс 18B.

Пример ORM 0404h, *AR4+

	Перед командой	После команды
AR4	0100	0101
Память данных		
0100h	4444	4444

17.3.78 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 B. Одновременно, эта команда умножает старшую часть аккумулятора A (биты 32–16) на содержание T, складывает произведение с аккумулятором B, округляет результат операции, и сохраняет конечный результат в аккумуляторе A. 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 3A.
Класс 3B.

Пример POLY *AR3+%

	Перед командой	После команды
A	00 1234 0000	00 0627 0000
B	00 0001 0000	00 2000 0000
T	5678	5678
AR3	0200	0201
Память данных		
0200h	2000	2000

17.3.79 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	008
SP	0300	0301

Память данных

0300h	0092	0092
040Ah	0055	0092

17.3.80 POPM 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
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	0060
SP	03F0	03F1

Память данных

03F0h	0060	0060

17.3.81 PORTR PA, Smem

Операнды Smem: операнд памяти данных одиночного доступа.
0 ≤ PA ≤ 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

17.3.82 PORTW Smem, PA

Операнды Smem: операнд памяти данных одиночного доступа.
0 ≤ PA ≤ 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	001
Память ввода-вывода		
0005h	0000	7FFA
Память данных		
0087h	7FFA	7FFA

17.3.83 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

17.3.84 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	1234
SP	2000	1FFF

Память данных

1FFFh	07FF	1234

17.3.85 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	$\overline{BIO}_{Low}$	0000 0011	NBIO	$\overline{BIO}_{high}$	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	C	C	C	C	C	C	C	C

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:

1. **Группа 1.** Можно выбрать не более двух условий. Каждое условие должно принадлежать разной категории (категория A или B); 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.

2. **Группа 2.** Можно выбрать не более трех условий. Каждое из этих условий должно быть из разных категорий (категория A, B или C); 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 A	Категория B	Категория A	Категория B	Категория C
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 A 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

17.3.86 READA 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	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.

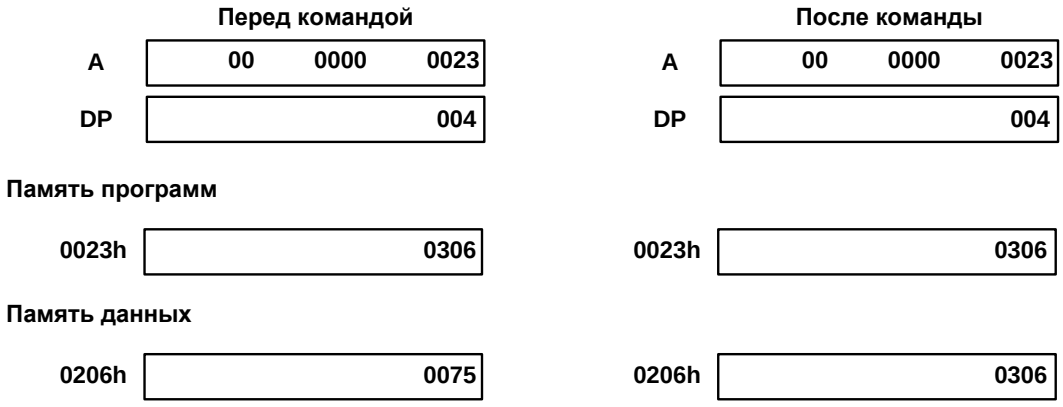
Описание Команда перемещает слово из ячейки памяти программ, адресуемой аккумулятором A в ячейку памяти данных, определенную Smem. Как только конвейер повторений начинает работу, команда становится одноцикловой. Для адресации ячейки памяти программ используется аккумулятор A(15-0).
Данная команда может 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



17.3.87 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 устройства 1901ВЦ1Т 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
PC	0025
INTM	0
IPTR	1

	После команды
PC	0080
INTM	1
IPTR	1

17.3.88 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

17.3.89 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	0110
SP	2001	2002
ST1	xCxx	x4xx
Память данных		
2001h	0110	0110

17.3.90 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 1901ВЦ1Т 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

17.3.91 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 её нет и в процессоре 1901ВЦ1Т. Наличие её кода в потоке команд воспринимается как холостая инструкция NOP.

17.3.92 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

17.3.93 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	00 8000 AAAB
C	x	1
TC	1	1

17.3.94 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	00 5800 091A
C	0	1

17.3.95 RPT

1: **RPT** Smem
2: **RPT** #K
3: **RPT** #lk

Операнды Smem: операнд памяти данных одиночного доступа.
0 ≤ K ≤ 255
0 ≤ lk ≤ 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	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	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 интераций выполнения команды. Число итераций (n) задается в 16-разрядном операнде памяти данных Smem или 8 или 16-ти 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 5A.
Синтаксис 1: Класс 5B.
Синтаксис 2: Класс 1.
Синтаксис 3: Класс 2.

Пример 1 RPT DAT127 ; DAT127 .EQU 0FFF

	Перед командой	После команды
RC	0	000C
DP	031	031
Память данных		
0FFFh	000C	000C

Пример 2 RPT #2 ; Повторение следующей инструкции 3 раза

	Перед командой	После команды
RC	0	0002

Пример 3 RPT #1111h ; Повторение следующей инструкции 4370 раз

Перед командой		После команды	
	0	RC	1111

17.3.96 RPTB[D] pmad

Операнды 0 ≤ pmad ≤ 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 → BRAF
If (delayed) then
    (PC) + 4 → RSA
Else
    (PC) + 2 → RSA
    pmad → 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 PC + 2 (или 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:

1. соответсвующие BRC, RSA и REA были сохранены(при необходимости) и затем восстановлены.
2. 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 29A.

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

17.3.97 RPTZ dst, #lk

Операнды dst: A (аккумулятор A).
B (аккумулятор B).
0 ≤ lk ≤ 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 → dst
lk → 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	00 0000 0000
RC	0000	03FF

17.3.98 RSBX N, SBIT

Операнды 0 ≤ SBIT ≤ 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	0	1	0	1	1	S	B	I	T

Выполнение 0 → STN(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 SBIT=8

	Перед командой		После команды
ST1	35CD	ST1	34CD

Пример 2 RSBX 1,8

	Перед командой		После команды
ST1	35CD	ST1	34CD

17.3.99 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	FF FE00 4321
ASM	01	01
AR0	0002	0002
AR3	0202	0204
Память данных		
0202h	0101	FC00

17.3.100 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

17.3.101 SFTA src, SHIFT [, dst]

Операнды src, dst A (аккумулятор A)
B (аккумулятор B)
–16 ≤ SHIFT ≤ 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
        (src((–SHIFT) – 1)) → C
        (src(39–0)) << SHIFT → dst
        If SXM = 1
        Then
(src(39)) → dst(39–(39 + (SHIFT + 1))) [or src(39–(39 + (SHIFT + 1))),
                if dst is not specified]
        Else
        0 → dst(39–(39 + (SHIFT + 1))) [or src(39–(39 + (SHIFT + 1))),
                if dst is not specified]
Else
        (src(39 – SHIFT)) → C
        (src) << SHIFT → dst
        0 → dst((SHIFT – 1)–0) [or src((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:

1. если значение SHIFT меньше 0, то:

   1) src((–SHIFT) – 1) копируется в бит переноса C.
   2) если SXM равен 1, команда выполняет арифметический сдвиг вправо и старшие биты src сдвигаются в dst(39–(39 + (SHIFT + 1))).
   3) если SXM равно 0, 0 записывается в dst(39–(39 + (SHIFT + 1))).

2. если значение SHIFT больше, чем 0, то:

   1) src(39 – SHIFT) копируется в бит переноса C.
   2) команда осуществляет арифметический сдвиг влево.
   3) 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	FF 8765 0055
B	00 4321 1234	FF FC3B 2802
C	x	1
SXM	1	1

Пример 2 SFTA B, +5

	Перед командой	После команды
B	80 AA00 1234	15 4002 4680
C	0	1
OVM	0	0
SXM	0	0

17.3.102 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 TC.

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 (TC) 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	FF FFFF E002
TC	x	0

17.3.103 SFTL src, SHIFT [, dst]

Операнды src, dst: A (аккумулятор A).
B (аккумулятор B).
–16 ≤ SHIFT ≤ 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
src((–SHIFT) – 1) → C
src(31–0) << SHIFT → dst
0 → dst(39–(31 + (SHIFT + 1)))
If SHIFT = 0
Then
0 → C
Else
src(31 – (SHIFT – 1)) → C
src((31 – SHIFT)–0) << SHIFT → dst
0 → dst((SHIFT – 1)–0) [or src((SHIFT – 1)–0), if dst is not specified]
0 → dst(39–32) [or 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:

3. 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))).

4. если значение SHIFT больше 0, то:

    1) src(31 – (SHIFT – 1)) копируется в бит переноса, C.
    2) Команда выполняет логический сдвиг влево.
    3) 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	FF	8765	0055
B	FF	8000	0000	00	043B	2802
C			0			1

Пример 2 SFTL B, +5

	Перед командой				После команды			
B	**80**	**AA00**	**1234**		**B**	**00**	**4002**	**4680**
C			**0**		**C**			**1**

17.3.104 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)) x (A(32–16)) + (B) → B

((Xmem) – (Ymem)) << 16 → 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	FF FFAB 0000
B	00 0000 0000	00 1BB1 8229
FRCT	0	0
AR3	0100	0101
AR4	0200	0201

Память данных

0100h	0055	0055
0200h	00AA	00AA

1: **SQUR** Smem, dst

2: **SQUR** 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
SQUR 30, B

	Перед командой	После команды
B	00 0000 01F4	00 0000 00E1
T	0003	000F
FRCT	0	0
DP	006	006

Память данных

031Eh	000F	000F

Пример 2 SQUR 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**

17.3.105 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	00 0320 00E1
T	0003	000F
FRCT	0	0
DP	006	006

Память данных

031Eh	000F	000F

Пример 2 SQURA *AR3+, A

	Перед командой	После команды
B	00 0000 01F4	00 0000 02D5
T	0003	000F
FRCT	0	0
DP	031E	031F

Память данных

031Eh	000F	000F

17.3.106 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	00 0000 0320
T	8765	1234
FRCT	0	0
DP	006	006

Память данных

0309h	1234	1234

Пример 2 SQURS *AR3, B

	Перед командой	После команды
B	00 014B 5DB0	00 0000 02D5
T	8765	1234
FRCT	0	0
AR3	0309	0309

Память данных

0309h	1234	1234

17.3.107 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

A	**00**	**70FF**	**FFFF**
AR5			**0202**
BRC			**4321**

Память данных

0202h			**1234**

После команды

A	**00**	**70FF**	**FFFF**
AR5			**0201**
BRC			**4321**

0202h			**4321**

17.3.108 SSBX N, SBIT

Операнды 0 ≤ SBIT ≤ 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 → STN(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	ST1	35CD

Пример 2 SSBX 1,8

	Перед командой		После команды
ST1	34CD	ST1	35CD

17.3.109 ST

1: **ST** T, Smem
2: **ST** TRN, Smem
3: **ST** #lk, Smem

Операнды Smem: операнд памяти данных одиночного доступа.
–32 768 ≤ lk ≤ 32 767

Код операции 1:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	1	1	0	0	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	0	0	1	1	0	1	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	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	004
Память данных		
0200h	0101	FFFF

Пример 2 ST TRN, 5

	Перед командой	После команды
DP	004	004
TRN	1234	1234
Память данных		
0205h	0030	1234

Пример 3 ST T, *AR7–

	Перед командой	После команды
T	4210	4210
AR7	0321	0320
Память данных		
0321h	1200	4210

17.3.110 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.
$0 \le SHFT \le 15$
$-16 \le SHIFT \le 15$

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	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	0	0	0	0	1	1	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
1	0	0	1	1	0	1	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	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 соответсвии 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 неизмененным.

Примечания – Следующие синтаксисы ассемблируются как различный синтаксис в определенных условиях.

5. Синтаксис 3: если SHFT = 0, код операции команды ассемблируется как Синтаксис 1.
6. Синтаксис 4: если SHIFT = 0, код операции команды ассемблируется как Синтаксис 1.

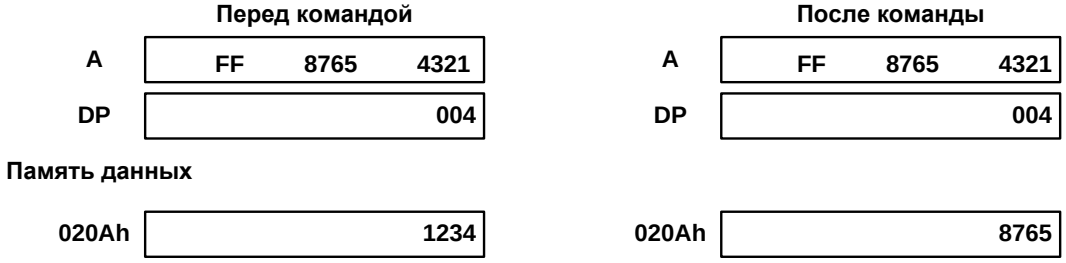
Синтаксис 4: если 0 < SHIFT ≤15 косвенный адрес приравнивается к режиму Xmem, код операции команды ассемблируется как Синтаксис 3.

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.

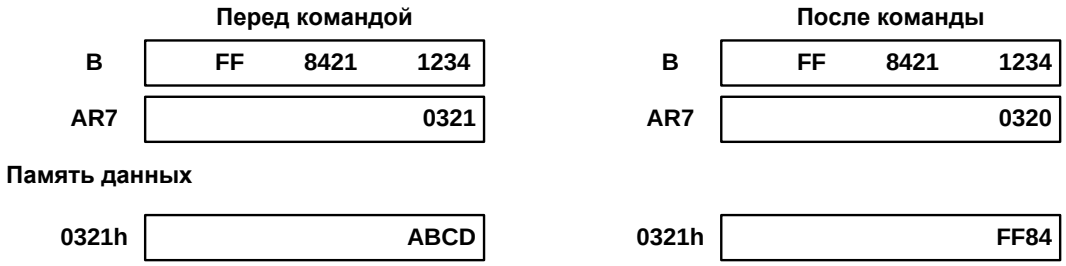
Циклы Синтаксисы 1, 2 и 3: 1 цикл.
Синтаксис 4: 2 цикла.
Необходимо добавить 1 цикл при использовании косвенной адресации с длинным смещением или абсолютной адресации с Smem.

Классы Синтаксисы 1, 2, и 3: Класс 10A.
Синтаксисы 1 и 2: Класс 10B.
Синтаксис 4: Класс 11A.
Синтаксис 4: Класс 11B.

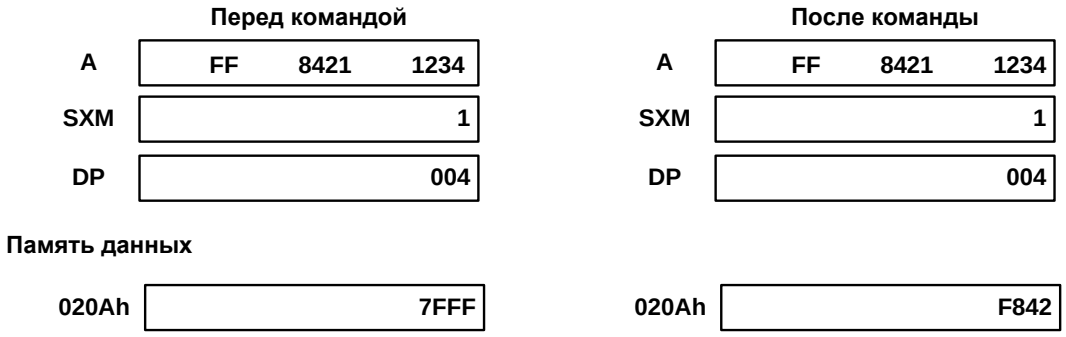
Пример 1 STH A, 10



Пример 2 STH B, –8, *AR7–



Пример 3 STH A, –4, 10



17.3.111 STL

1: **STL** src, Smem
2: **STL** src, ASM, Smem
3: **STL** src, SHFT, Xmem
4: **STL** src [, SHIFT], Smem

Операнды

src: A (аккумулятор A).
B (аккумулятор B).
Smem: операнд памяти данных одиночного доступа.
Xmem: операнд памяти данных двойного доступа.
0 ≤ SHFT ≤ 15
–16 ≤ SHIFT ≤ 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	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	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
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 соответсвии 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 ≤ SHIFT ≤15 косвенный модификатор 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 10A.
Синтаксисы 1, 2 и 3: Класс 10B.
Синтаксис 4: Класс 11A.
Синтаксис 4: Класс 11B.

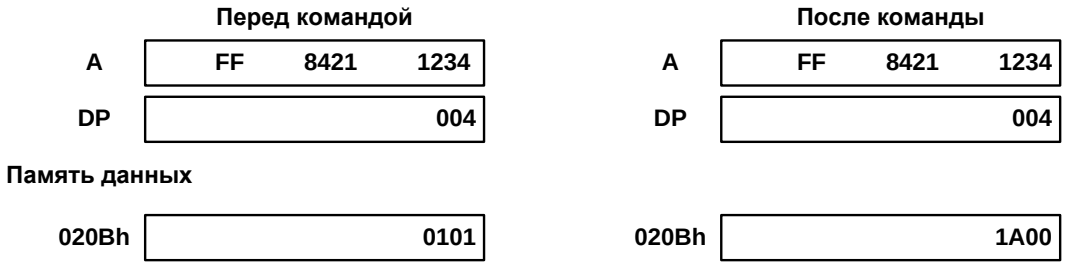
Пример 1 STL A, 11

	Перед командой	После команды
A	FF 8765 4321	FF 8765 4321
DP	004	004
Память данных		
020Bh	1234	4321

Пример 2 STL B, –8, *AR7–

	Перед командой	После команды
B	FF 8421 1234	FF 8421 1234
SXM	1	1
AR7	004	004
Память данных		
0321h	0099	2112

Пример 3 STL A, 7, 11



17.3.112 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 10A.

Пример 1 STLM A, BRC

	Перед командой	После команды
A	FF 8765 4321	FF 8765 4321
BRC(1Ah)	1234	
BRC		4321

Пример 2 STLM B, *AR1–

	Перед командой	После команды
B	FF 8421 1234	FF 8421 1234
AR1	3F17	0016
AR7(17h)	0099	
AR7		1234

17.3.113 STM #lk, MMR

Операнды MMR: Отображаемый в памяти регистр.
–32 768 ≤ lk ≤ 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 Отсутсвуют.

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 12A.

Пример 1 STM 0FFFFh, IMR

	Перед командой	После команды
IMR	FF01	FFFF

Пример 2 STM 8765h, *AR7+

	Перед командой	После команды
AR0	0000	8765
AR7	8010	0011

17.3.114 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	FF 8021 1000
B	00 0000 1111	FF 0422 1000
OVM	0	0
SXM	1	1
ASM	1	1
AR0	0002	0002
AR3	0200	0200
AR5	0300	0302

Память данных

0200h	0101	0309h 0842
0300h	8001	0309h 8001

17.3.115 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	FF 8001 0000
B	FF 8421 1234	FF 8421 1234
SXM	1	0
ASM	1C	1
AR2	01FF	01FE
AR4	0200	0201
Память данных		
01FFh	xxxx	F842
0200h	8001	8001

Пример 2

```
ST A, *AR3
||LD *AR4, T
```

	Перед командой	После команды
A	FF 8421 1234	FF 8421 1234
T	3456	80FF
ASM	1	1
AR3	0200	0200
AR4	0100	0100
Память данных		
0200h	0001	0842
0100h	80FF	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.

17.3.116 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
Else
(Xmem) → (T) + (dst) → dst

Статусные биты 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	00 0011 1111
B	00 0000 1111	00 010D 0000
T	0400	0400
ASM	1C	1C
FRCT	0	0
AR4	0100	0101
AR5	0200	0201

Память данных

100h	1234	0001
200h	4321	4321

17.3.117 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 << (ASM - 16)) → Ymem
If (Rounding)
Then
Round ((dst) – (Xmem) × (T)) → dst
Else
(dst) – (Xmem) × (T) → 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 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+
||MAS *AR5, B
```

	Перед командой	После команды
A	00 0011 1111	00 0011 1111
B	00 0000 1111	FF FEF3 8D11
T	0400	0400
ASM	5	5
FRCT	0	0
AR4	0100	0101
AR5	0200	0200
Память данных		
0100h	1234	0222
0200h	4321	4321

Пример 2

```
ST A, *AR4+
||MASR *AR5+, B
```

Перед командой

A	**00**	**0011**	**1111**
B	**00**	**0000**	**1111**
T			**0400**
ASM			**0001**
FRCT			**0**
AR4			**0100**
AR5			**0200**

Память данных

0100h	**1234**
0200h	**4321**

После команды

A	**00**	**0011**	**1111**
B	**FF**	**FEF4**	**0000**
T			**0400**
ASM			**0001**
FRCT			**0**
AR4			**0101**
AR5			**0201**

Память данных

0100h	**0022**
0200h	**4321**

17.3.118 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	FF 8421 1234
B	xx xxxx xxxx	00 2000 0000
T	4000	4000
ASM	00	00
FRCT	1	1
AR3	0200	0201
AR5	0300	0301
Память данных		
0200h	1111	8421
0300h	4000	4000

17.3.119 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	FF 8421 0000
B	00 1000 0001	FF FBE0 0000
ASM	01	01
SXM	1	1
AR0	0002	0002
AR3	01FF	01FE
AR5	0300	0302

Память данных

01FFh	1111	0842
0300h	8001	8001

17.3.120 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 T в ячейке памяти данных Xmem. Если условие ложное, то команда считывает Xmem и записывает значение 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 1 STRCD *AR5–, AGT

	Перед командой	После команды
A	00 70FF FFFF	00 70FF FFFF
T	4321	4321
AR5	0202	0201
Память данных		
0202h	1234	4321

17.3.121 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 ≤ lk ≤ 32 767
0 ≤ SHFT ≤ 15
–16 ≤ SHIFT ≤ 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 заём, 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:

1. дополняются знаком, если SXM = 1
2. сбрасываются, если SXM = 0

Примечания:
Следующие синтаксисы ассемблируются в разные синтаксисы в соответствующих условиях.

1. Синтаксис 4: если dst = src и SHIFT = 0, код операции команды ассемблируется как Синтаксис 1.

Синтаксис 4: если dst = src, SHIFT ≤ 15, и режим косвенной адресации Smem включен в Xmem, то код операции команды ассемблируется как Синтаксис 1.

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 3A.
Синтаксисы 1, 2 и 3: Класс 3B.
Синтаксис 4: Класс 4A.
Синтаксис 4: Класс 4B.
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	FF FAC0 1200
C	x	0
SXM	1	1
AR1	0100	0101
Память данных		
0100h	1500	1500

Пример 2 SUB A, –8, B

	Перед командой	После команды
A	00 0000 1200	00 0000 1200
B	00 0000 1800	00 0000 17EE
C	x	1
SXM	1	1

Пример 3 SUB #12345, 8, A, B

	Перед командой	После команды
A	00 0000 1200	00 0000 1200
B	00 0000 1800	FF FFCF D900
C	x	0
SXM	1	1

Пример 4 ST B, *AR2–
||LD *AR4+, A

A	00 0000 001C	00 4214 1414
B	FF 8421 1234	FF 8421 1234
SXM	1	1
ASM	1C	1C
AR2	01FF	01FE
AR4	0200	0201

Память данных

01FFh	xxxx	F842
0200h	8001	8001

Пример 5 ST A, *AR3
||LD *AR4, T

	Перед командой	После команды
A	FF 8421 1234	FF 8421 1234
T	3456	80FF
ASM	1	1
AR3	0200	0200
AR4	0100	0100

Память данных

0200h	0001	0842
0100h	80FF	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.

17.3.122 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 заём), 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	FF FFFF FFFF
C	0	0
DP	008	008
Память данных		
0405h	0006	0006

Пример 2 SUBB *AR1+, B

	Перед командой	После команды
B	FF 8000 0006	FF 8000 0000
C	1	1
OVM	1	1
AR1	0405	0406
Память данных		
0405h	0006	0006

17.3.123 SUBC 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	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:

1. если SXM = 1, делитель должен иметь нулевое значение значение старшего бита.
2. 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 должено 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			
B	00	0000	0041		B	00	0002	0009
C			x		C			1
AR1			1000		AR1			1000
Память данных								
1000h			0007		1000h			0007

17.3.124 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

B	00 0000 0002
C	x
AR2	0100

Память данных

0100h	F006

После команды

B	FF FFFF 0FFC
C	0
AR2	00FF

Память данных

0100h	F006

17.3.125 TRAP K

Операнды $0 \le K \le 31$

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 так же 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

17.3.126 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 адрессации) 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

17.3.127 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	$\overline{BIO}_{Low}$	0000 0011	NBIO	$\overline{BIO}_{high}$	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	N	1	C	C	C	C	C	C	C	C

Синтаксис	Код n
1	0
2	1

Выполнение

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:

1. если n = 1 и условие(я) удовлетворяется, то выполняется однословная команда, следующая за данной командой.
2. если n = 2 и условие(я) удовлетворяется, то выполняется одна двухсловная или две однословные команды, следующие за данной командой.
3. 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:

1. **Группа 1:** можно выбрать не более двух условий. Кажде из этих условий должно принадлежать различным категориям (категория A или B); 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.
2. **Группа 2:** можно выбрать не более трех условий. Каждое из этих условий должно принадлежать разным категориям (категория A, B или C); нельзя выбрать два условия из одной категории. Например, можно осуществить проверку TC, C и BIO одновременно, но нельзя одновременно проверять NTC, C и NC.

Условия для данной команды

Группа 1		Группа 2		
Категория A	Категория B	Категория A	Категория B	Категория C
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**

17.3.128 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 ≤ 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	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 3A.
Синтаксис 1: Класс 3B.
Синтаксисы 2 и 3: Класс 2.
Синтаксис 4: Класс 1.

Пример 1 XOR *AR3+, A

	Перед командой			После команды		
A	00	00FF	1200	00	00FF	0700
AR3			0100			0101
Память данных						
0100h			1500			1500

Пример 2 XOR A, +3, B

	Перед командой			После команды		
A	00	0000	1200	00	0000	1200
B	00	0000	1800	00	0000	8800

17.3.129 XORM #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	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	00FF
Память данных		
0100h	4444	4040

18 Контроллер McBSP (DSP)

McBSP обеспечивает:

- полнодуплексный режим работы;
- буферное FIFO на 8 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.

Так же 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;
  - SPI устройства.
- многоканальные прием и передача до 128 каналов;
- произвольная длина слова от 4х до 32х 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.

18.1 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 3-проводные 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 DMA-контроллер 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, раскодирования данных и пр.

Кроме того в McBSP присутствуют статусные выводы для информирования ЦП или DMA-контроллера о различных событиях и посредством программируемых выводов прерываний, независимых для приемника и передатчика.

18.2 Регистры McBSP

18.2.1 McBSP1

Таблица 18-1 – 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
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 и передатчика
0x3000_004C	0x0026	0x09	MCRH	Регистр управления многоканальными режимами приемника и передатчика
0x3000_004C	0x0026	0x0A	XCERL	Регистр маски передатчика
0x3000_004C	0x0026	0x0B	XCERH	Регистр маски передатчика
0x3000_004C	0x0026	0x0C	RCERL	Регистр маски приемника
0x3000_004C	0x0026	0x0D	RCERH	Регистр маски приемника
0x3000_004C	0x0026	0x0E	PCR	Регистр управления выводами
0x3000_004C	0x0026	0x0F	SPSR	Регистр состояния

18.2.2 McBSP2

Таблица 18-2 – 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
0x3000_005C	0x002E	0x08	MCRL	Регистр управления многоканальными режимами приемника и передатчика
0x3000_005C	0x002E	0x09	MCRH	Регистр управления многоканальными режимами приемника и передатчика
0x3000_005C	0x002E	0x0A	XCERL	Регистр маски передатчика
0x3000_005C	0x002E	0x0B	XCERH	Регистр маски передатчика
0x3000_005C	0x002E	0x0C	RCERL	Регистр маски приемника
0x3000_005C	0x002E	0x0D	RCERH	Регистр маски приемника
0x3000_005C	0x002E	0x0E	PCR	Регистр управления выводами
0x3000_005C	0x002E	0x0F	SPSR	Регистр состояния

18.2.3 McBSP3

Таблица 18-3 – 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 и передатчика
0x3000_006C	0x0036	0x09	MCRH	Регистр управления многоканальными режимами приемника и передатчика
0x3000_006C	0x0036	0x0A	XCERL	Регистр маски передатчика
0x3000_006C	0x0036	0x0B	XCERH	Регистр маски передатчика
0x3000_006C	0x0036	0x0C	RCERL	Регистр маски приемника
0x3000_006C	0x0036	0x0D	RCERH	Регистр маски приемника
0x3000_006C	0x0036	0x0E	PCR	Регистр управления выводами
0x3000_006C	0x0036	0x0F	SPSR	Регистр состояния

18.3 Конфигурирование последовательного порта

Настройка последовательного порта осуществляется через регистры управления SPCR и PCR. В них содержатся биты управления а так же биты состояния McBSP.

Так же в регистр PCR предназначен для управления назначением выводов McBSP.

18.3.1 SPCRH

Таблица 18-4 – Регистр SPCRH

15	14	13	12	11	10	9	8
FREE	XINTM[2:0]			XEVNT1[1:0]		XEVNT0[1:0]	
R/W,+0	R/W,+0			R/W,+0		R/W,+0	
7	6	5	4	3	2	1	0
SOFT	RINTM[2:0]			REVNT1[1:0]		REVNT0[1:0]	
R/W,+0	R/W,+0			R/W,+0		R/W,+0	

Таблица 18-5 – Описание бит регистра SPCRH

№ 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
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

18.3.2 SPCRL

Таблица 18-6 – Регистр SPCRL

15	14	13	12	11	10	9	8
DXENA	LW_ACC	JBOUND[1:0]		XJUST[1:0]		RJUST[1:0]	
R/W,+0	R/W,+0	R/W,+1		R/W,+0		R/W,+0	
7	6	5	4	3	2	1	0
DLB	ABIS	CLKSTP[1:0]		nXRST	nRRST	nFGRST	nCGRST
R/W,+0	R/W,+0	R/W,+0		R/W,+0	R/W,+0	R/W,+0	R/W,+0

Таблица 18-7 – Описание бит регистра SPCRL

№ 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
13…12	JBOUND[1:0]	Граница выравнивания байтов/слов 00 – 1 байт 01 – 2 байта 1x – 4 байта
11…10	XJUST[1:0]	Выравнивание передаваемых данных 0x – Выравнивание по правой границе 1x – 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 0x – 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

18.3.3 SPSRH

Таблица 18-8 – Регистр SPSRH

15	14	13	12	11	10	9	8
-	XFIFO_F	XFIFO_H	XFIFO_E	XLOST	XSERR	XEMPTY	XRDY
	R,+0	R,+0	R,+1	R,+0	R,+0	R,+1	R,+1

7	6	5	4	3	2	1	0
-	RFIFO_F	RFIFO_H	RFIFO_E	RLOST	RSERR	REMPTY	RRDY
	R,+0	R,+0	R,+1	R,+0	R,+0	R,+0	R,+0

Таблица 18-9 – Описание бит регистра SPSRH

№ 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 (заполненн 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 – Имелась 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 (заполненн более чем на половину)
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 – Имелась 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

18.3.4 PCRL

Таблица 18-10 – Регистр PCRL

15	14	13	12	11	10	9	8
-	-	XIOEN	RIOEN	FSXM	FSRM	CLKXM	CLKRM
		R/W,+0	R/W,+0	R/W,+1	R/W,+1	R/W,+1	R/W,+1

7	6	5	4	3	2	1	0
	CLKS_ST	DX_ST	DR_ST	FSXP	FSRP	CLKXP	CLKRP
	R,+0	R,+0	R,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0

Таблица 18-11 – Описание бит регистра PCRL

№ 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
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

18.4 Управление приемом и передачей

Управление приемом и передачей данных управляется посредством регистров управления приемником RCR и управления передатчиком XCR

18.4.1 RCRH, XCRH

Таблица 18-12 – Регистры RCRH, XCRH

15	14	13	12	11	10	9	8
MPHASE	FLEN_P1[6:0]						
R/W,+0	R/W,+31						
7	6	5	4	3	2	1	0
CODEC[2:0]			WLEN_P1[4:0]				
R/W,+0			R/W,+15				

Таблица 18-13 – Описание бит регистров RCRH, XCRH

№ 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 4-х бит не гарантируется

18.4.2 RCRL, XCRL

Таблица 18-14 – Регистры RCRL, XCRL

15	14	13	12	11	10	9	8
FIG	FLEN_P0[6:0]						
R/W,+0	R/W,+31						
7	6	5	4	3	2	1	0
MSB1st	DELAY[1:0]		WLEN_P0[4:0]				
R/W,+0	R/W,+1		R/W,+15				

Таблица 18-15– Описание бит регистров RCRL, XCRL

№ 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)
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 4х бит не гарантируется

18.5 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 кол-ва 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 заполнено. После этого данные попадаюсь в сдвиговый регистр передатчика XSR откуда выдвигаются через вывод TX.

18.6 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).

18.7 Программный внутренний сброс (биты управления регистра SPCR)

В McBSP предусмотрен независимый сброс отдельных его блоков посредство 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 так же синхронно.

Примечание – перед сменой параметров приемника/передатчика рекомендуется перевести их в состояние программного внутреннего сброса, сменить параметры, вывести из состояния программного внутреннего сброса

18.8 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 DMA-запросов или запросов прерывания ЦП.

18.9 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 так же 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).

18.10 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. Так же имеется программируемый вывод состояния передатчика, на который может быть выведен данный признак (SPCRH.XINTM=000b).

18.11 События и прерывания

В McBSP имеются 6 программируемых вывода для информирования управляющее ЦП (XINT и RINT) или DMA-контроллер (XEVNT[x2] и REVENT[x2]) о возникновении события.

Прерывания приемника (RINT) и передатчика (XINT) информируют ЦП об изменении состояния последовательного порта. Данные выводы программируемы и для каждого доступны 8 настроек, определяемых полем конфигурации (RINTM и XINTM соответственно) источников событий:

1. RINTM/XINTM = 000b. Прерывание по каждому слову перемещаемому между интерфейсными 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 выводы XEVNT0/1 и REVENT0/1 применяются поля XEVNT0/1 и REVNT0/1 соответственно. В отличие от выводов прерываний данные выводы не имеют промежуточного регистра на выходе блока. Назначение сигналов аналогично выводам RINT/XINT.

18.12 Настройка битовой и кадровой синхронизации

На рисунке показан 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;
- Кол-во слов в кадре для каждой фазы;
- Кол-во битов с 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).

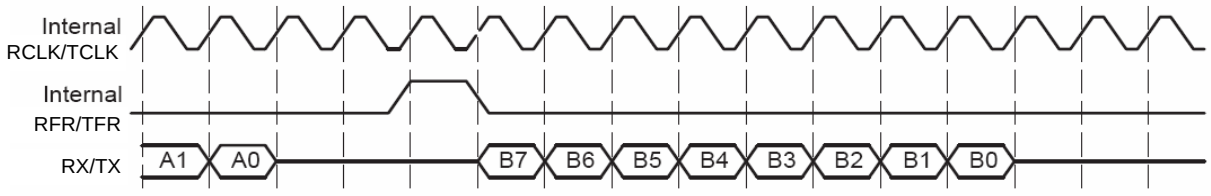


Рисунок 18–1 – Работа кадровой и битовой синхронизации

Битовая и кадровая синхронизации приемника и передатчика могут задаваться от внутреннего или от внешнего источника. Управление режимами осуществляется через биты режима в регистре PCR. Так же 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 вентилей типа XOR.

18.12.1 Фазы кадровой синхронизации

Кадровая синхронизация обозначает начало обмена в McBSP. Поток данных, следующий за кадровой синхронизацией, может иметь одну или 2 фазы (фаза 1 и фаза 2). Кол-во 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 число слов в фазе кадра (от 1 до 128 слов), а размер слова определяется длину 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.

18.12.2 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 4 байта (8 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 4-х слов;
- WLEN_P0 = 0000111b (0x07), слово из 8-ми бит.

В данном случае между ЦП или DMA-контроллером и McBSP будет совершено 8 обменов 8 битными словами – 4 чтения из DRR и 4 записи в DXR.

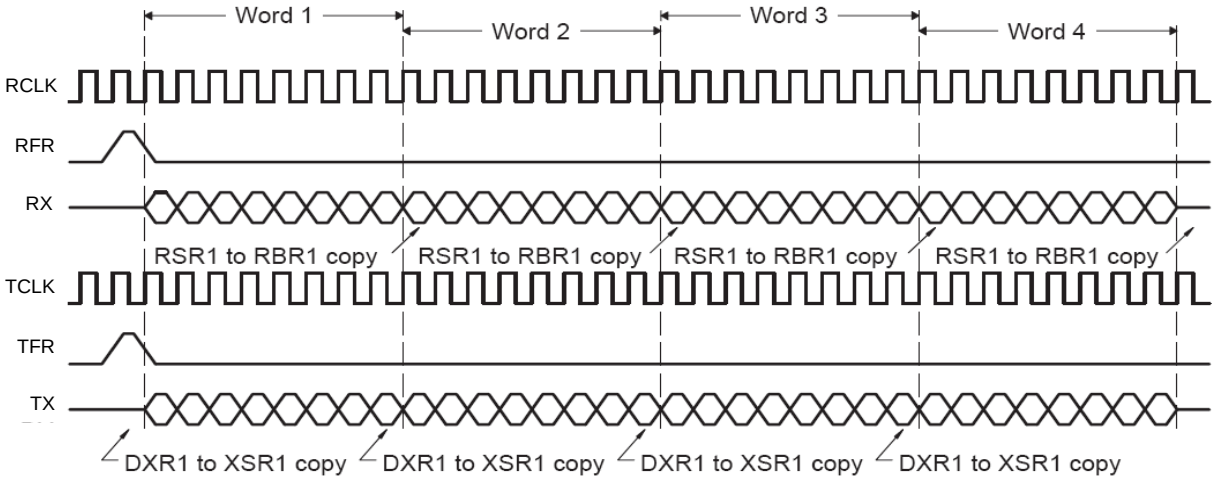


Рисунок 18–2 – Пример упаковки данных методом задания размера (несколько слов)

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 1го слова
- WLEN_P0 = 11111b (0x1F), слово из 32х бит

В данном случае будет произведено всего 2 обмена с ЦП или DMA-контроллером – по одному 32-разрядному слову на прием и на передачу.

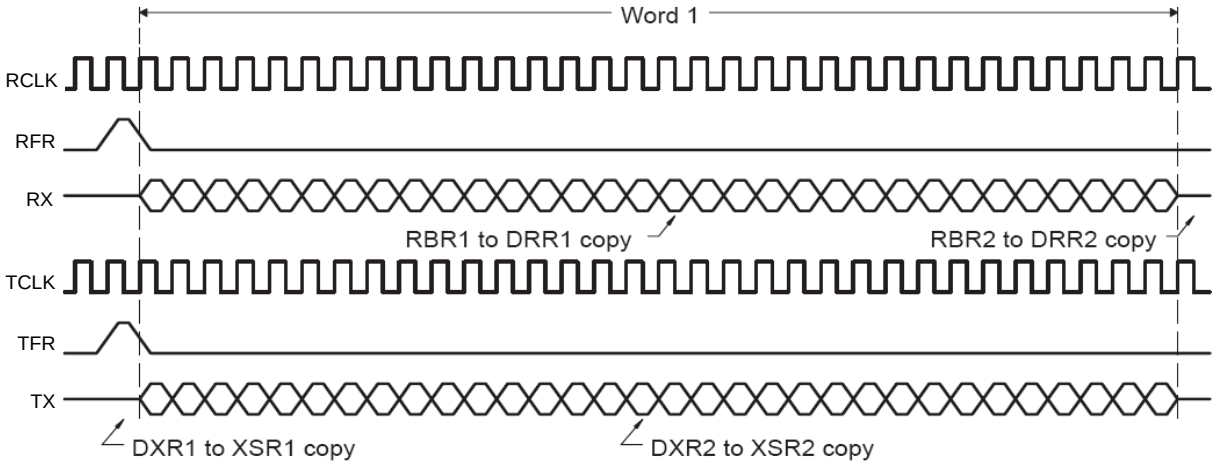


Рисунок 18–3 – Пример упаковки данных методом задания размера (одно слово)

Как видно из примера управление разрядностью передаваемых слов и их кол-вом позволяет проводить упаковку и распаковку данных, тем самым снижая нагрузку на ЦП или DMA-контроллер.

18.13 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 0 до 3-х битовых интервалов включительно. По сбросу установлена задержка в 1 битовый интервал.

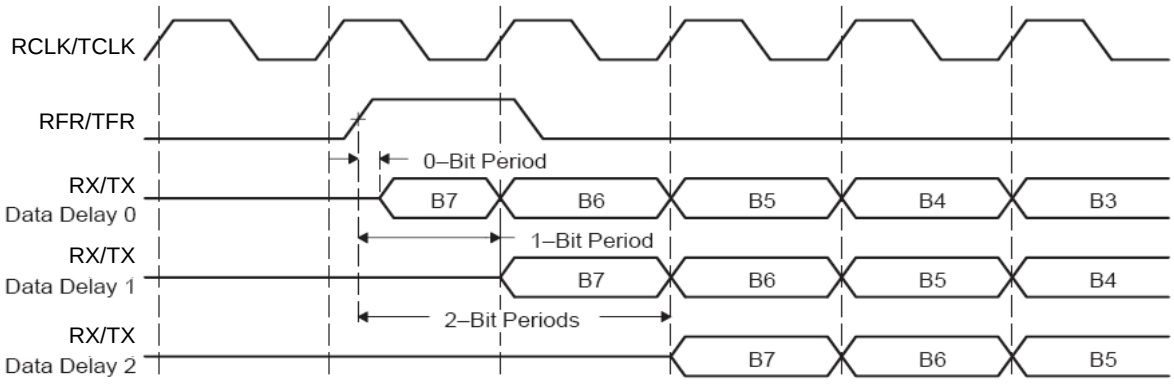


Рисунок 18–4 – Задержка данных

Обычно кадровый импульс обнаруживается и захватывается по фронту сигнала битовой синхронизации. Т.о. 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. Т.о первый бит передаваемых данных должен уже находиться в регистре XSR и т.о. транслироваться на вывод DX.

Другой часто используемый режим это задержка данных на 2 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 2 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.

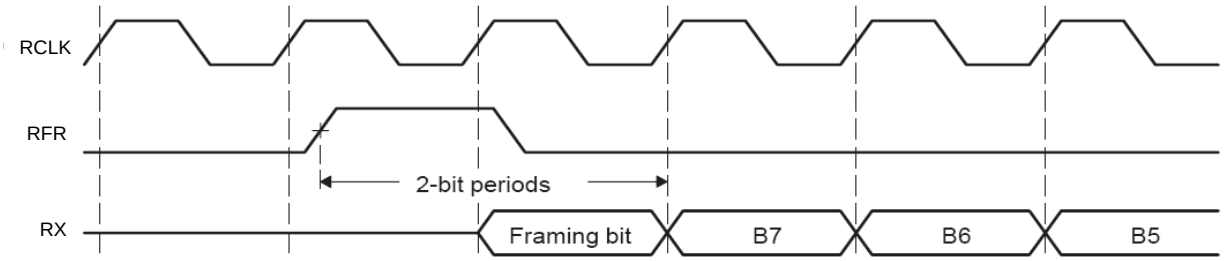


Рисунок 18–5 – Задержка данных на два битовых интервала

18.14 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.

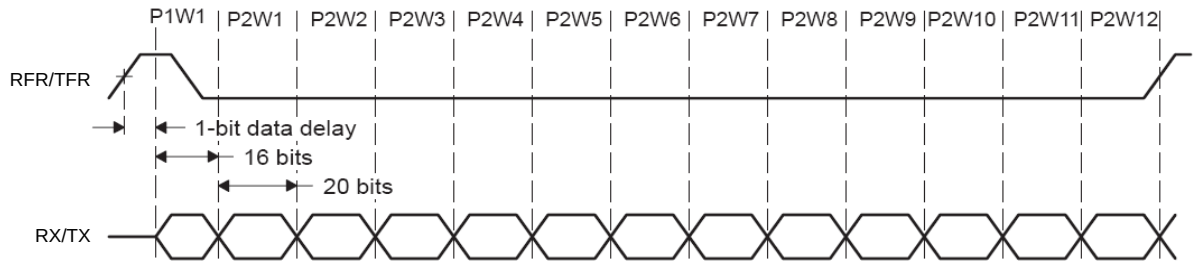


Рисунок 18–6 – Многофазный кадр

Примечательно что в кадрах данного формата импульс кадровой синхронизации перекрывает первое слово кадра. В McBSP кадровая синхронизация осуществляется по переходу кадрового синхроимпульса из неактивного состояния в активное и т.о. сигнал кадровой синхронизации длительностью менее длины кадра эквивалентен импульсу в один битовый интервал.

18.15 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 4х бит работа устройства не гарантируется.

Ниже на рисунке приведен пример однофазного кадра содержащего одно 8-битное 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.

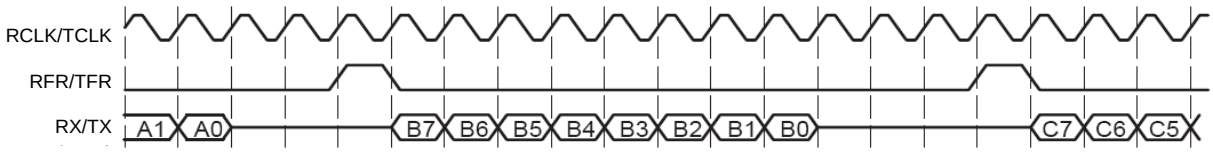


Рисунок 18–7 – Работа приемника

Ниже показан пример приема данных по последовательному порту (Рисунок 18–8). 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 DMA-контроллером. При чтении регистра DRR ЦП или DMA-контроллером признак готовности данных снимается.

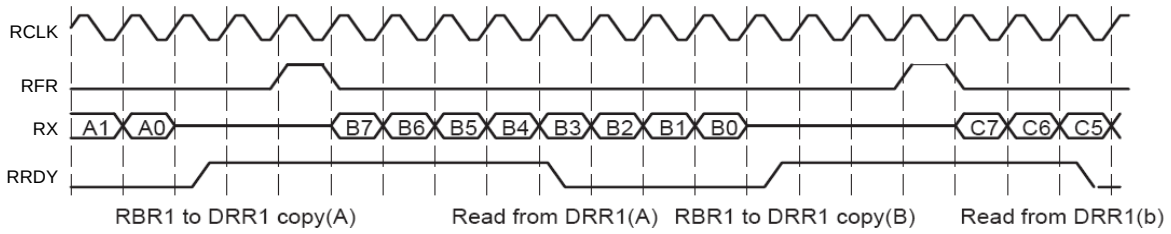


Рисунок 18–8 – Выставление признака готовности приемника

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.

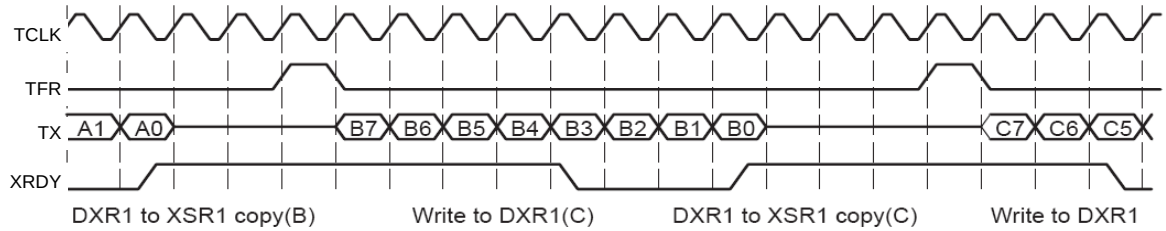


Рисунок 18–9 – Работа передатчика

Частота следования кадров определяется периодом между сигналами кадровой синхронизации:

$Ff = Fb/N,$

$N = D+Nb+Nz,$

где

Ff – частота следования кадров;

Fb – 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 кол-ва битов и в пределе ограничивается только числом значащих битов кадра. Т.о. Максимальная частота следования кадров определяется как

Ff = Fb/Nb,

где
Ff – частота следования кадров;
Fb – частота следования битов
Nb — число битов в кадре

Ниже показан 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.

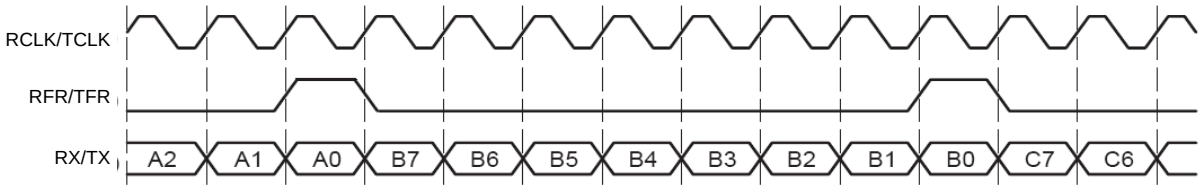


Рисунок 18–10 – Работа передатчика на предельной частоте следования кадров

В действительности допускаются непрерывные потоки данных и т.о. нет необходимости стробировать импульсом кадровой синхронизации все кадры. Теоретически достаточно только первого кадрового импульса для запуска много кадровой передачи. в McBSP предусмотрен подобный режим работы игнорированием импульса кадровой синхронизации кроме первого. Данный режим включается программно установкой бита FIG = 1.

18.16 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.

18.17 Упаковка данных при помощи режима пропуска кадровой синхронизации

Ранее был описан один из методов упаковки данных при помощи изменения размера принимаемого и кол-ва 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. т.о. 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 32х бит необходимо установить бит [X/R]CR.FIG = 1 и т.о. 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 18–11, Рисунок 18–12).

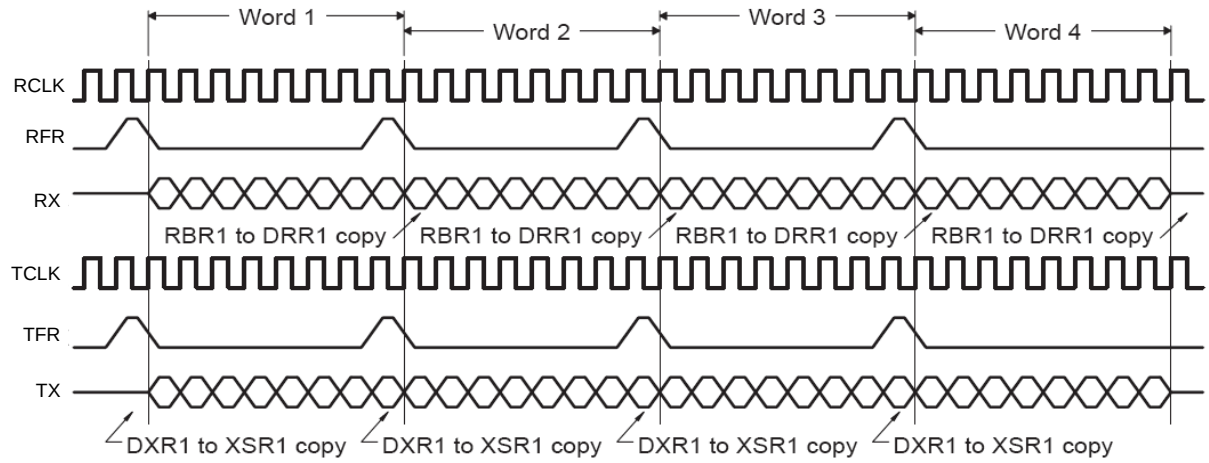


Рисунок 18–11 – Пример упаковки данных при помощи режима кадровой синхронизации (несколько слов)

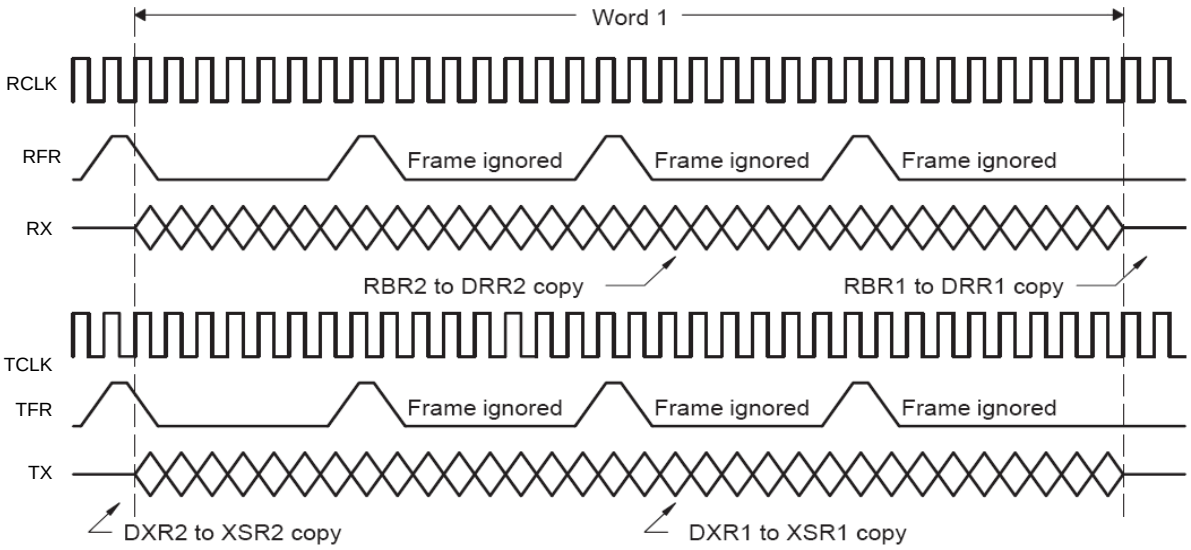


Рисунок 18–12 – Пример упаковки данных при помощи режима кадровой синхронизации (одно слово)

18.18 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. Так же 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 продолжаются без разрывов между словами данных. Если же кадровый синхроимпульсов кроме первого, запускающего прием/передачу, не оказывают никакого влияния па 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.

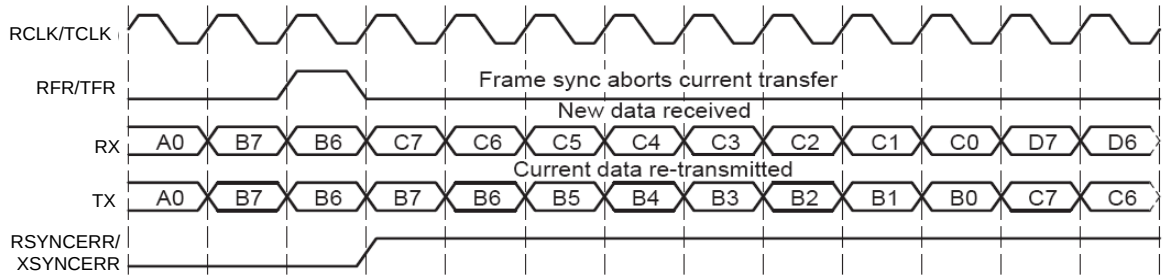


Рисунок 18–13 – Работа модуля в обычном режиме

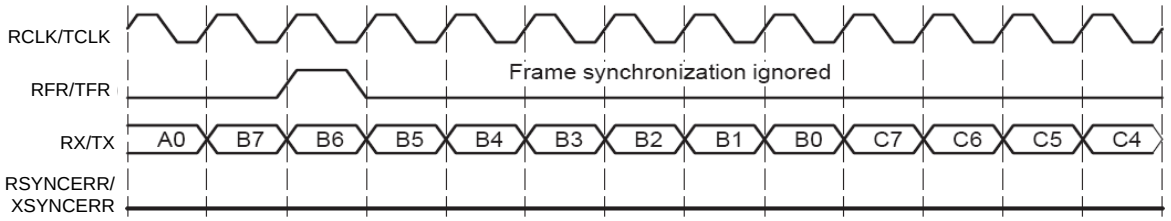


Рисунок 18–14 – Работа модуля в режиме пропуска кадровых синхроимпульсов

18.19 Особые ситуации работы последовательного порта

В контроллере McBSP существует 5 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 DMA-контроллер 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.

18.20 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. Так же 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 показано дерево решения по обработке всех кадровых синхроимпульсов приемника.

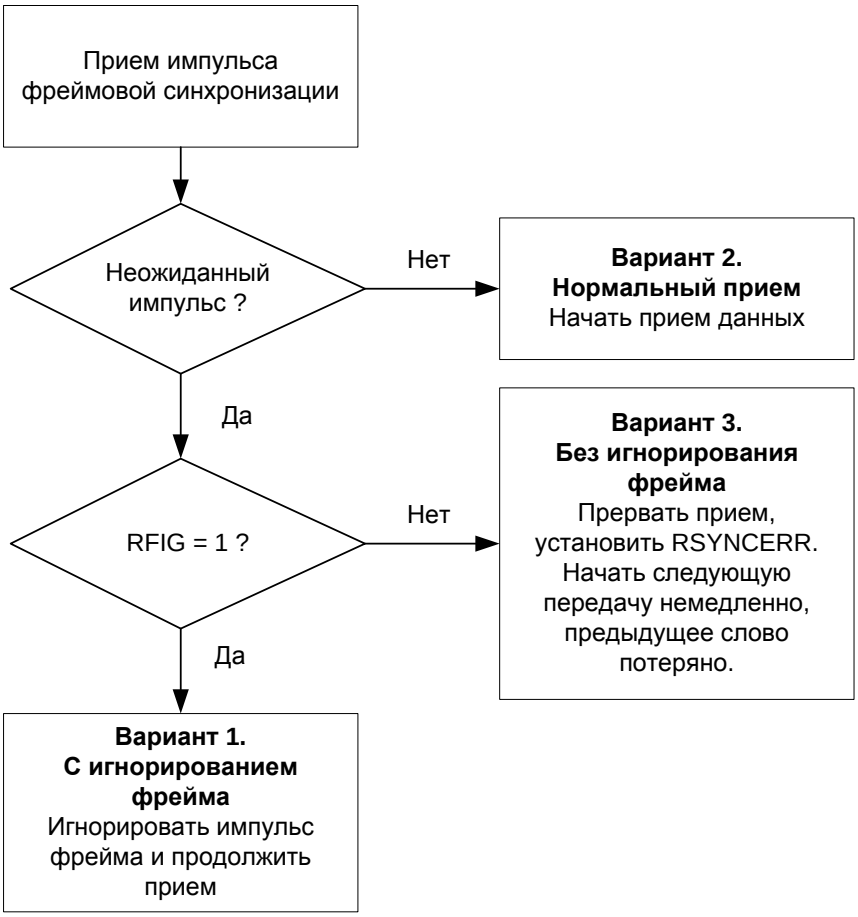


Рисунок 18–15 – Алгоритм обработки всех кадровых синхроимпульсов приемника

Диаграмма (см. Рисунок 18–16) 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 4 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 3 случая:
   1). SFSR первый синхроимпульс после сброса приемника (RSR = 1);
   2). SFSR первый синхроимпульс после снятия флага ошибки переполнения приемника (RFULL = 1);
   3). 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.

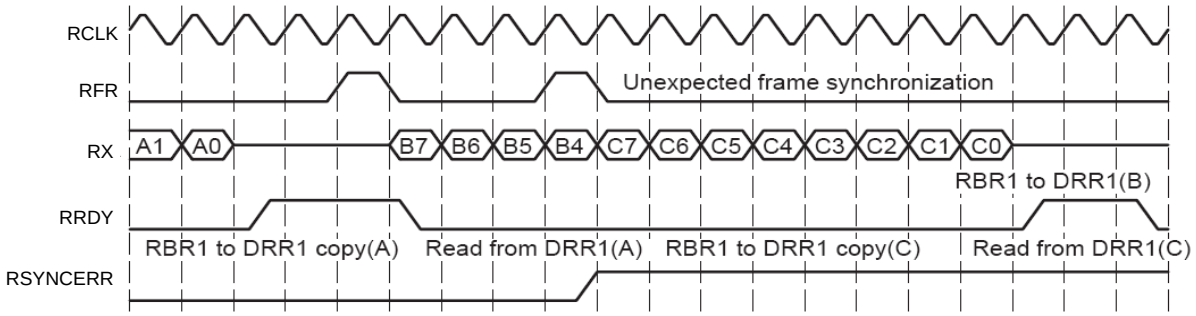


Рисунок 18–16 – Диаграмма обработки непредвиденных кадровых импульсов

18.21 Перезапись передаваемых данных.

Рисунок 18–17 показывает, что происходит, если данные в DXR были перезаписаны до того, как были отправлены. В примере изначально программист записывает данные C для передачи. После чего записывает данные D, перезаписывая данные C до того, как они были скопированы в FIFO передатчика XBR. Т.о. Данные C не будут переданы. ЦП или может избежать этого опрашивая бит готовности передатчика к приему новых данных для отправки (XRDY). DMA-контроллер может избежать этого, работая синхронизируя передачи по XRDY в одиночном режиме или по состоянию XBR в блочном режиме (состояния полуполного или пустого FIFO).

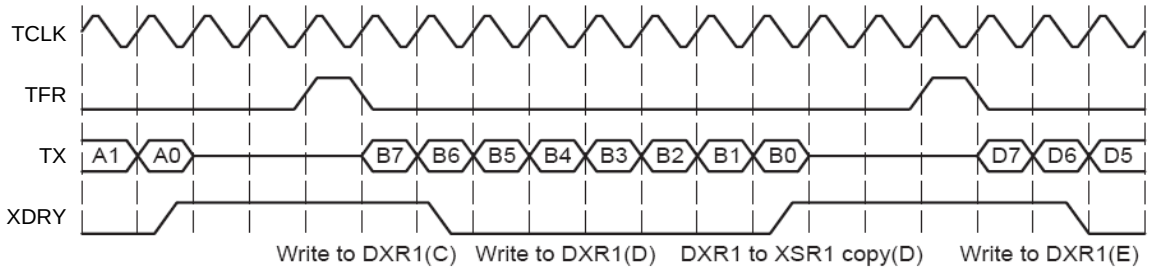


Рисунок 18–17 – Диаграмма работы модуля, в случае перезаписи неотправленных данных

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.

18.22 Выравнивание данных и расширение знака

В McBSP предусмотрено автоматическое выравнивание данных по границе байта, слова, двойного слова, а так же 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 8й 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 8 – 5 = 3 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.

18.23 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 4 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.

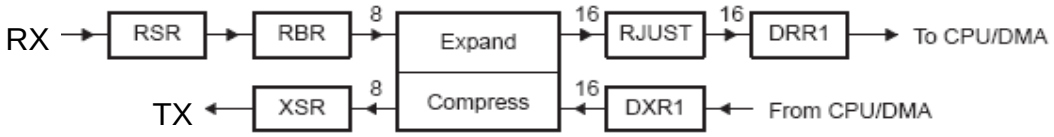


Рисунок 18–18 – Структурная схема модуля кодирования/декодирования данных

18.24 Компадирование/декомпадирование по u- и A-законам

При компадировании по u- и A-законам необходимо выравнивание данных согласно приведенному ниже (Рисунок 18–19 и Рисунок 18–20).

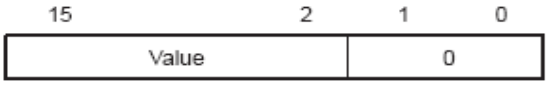


Рисунок 18–19 – Выравнивание данных для компадирования по u-закону в DXR

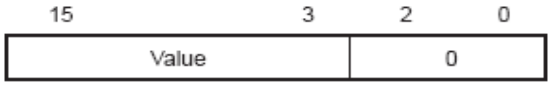


Рисунок 18–20 – Выравнивание данных для компадирования по A-закону в DXR

При декомпадировании данные декодируются как 16 битное слово. При этом декомпадированные данные в приемнике расположены так же 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.

18.25 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 1-битовый интервал, то необходимо установить следующие настройки:

- MPHASE = 0b , однофазный кадр;
- CODEC = 001b (0x1), кодирование/декодирование кодом Манчествер-2;
- FLEN_P0 = 0000000b (0x00), кадр из 1го слова;
- WLEN_P0 = 00111b (0x0F), слово из 8 бит;
- DELAY = 01b (0x1), задержка 1 битовый интервал.

18.26 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.

18.27 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.

Схема подключений синхронизации приведена на рисунке ниже.

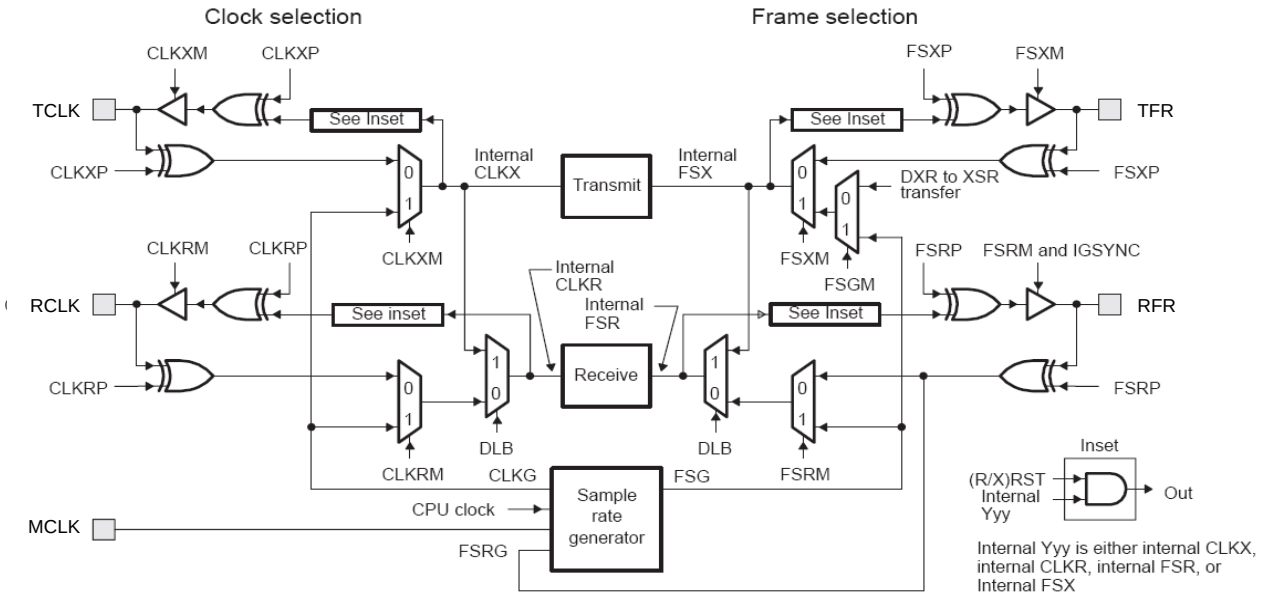


Рисунок 18–21 – Структурная схема модуля управления синхронизацией

Примечание – После смены источника частоты синхронизации для того или иного блока необходимо произвести программный сброс данного блока, в противном случае работа блока не гарантируется.

18.28 Внутренний генератора кадровой и битовой синхронизации

Внутренний генератор кадровой и битовой синхронизации структурно скомпонован из 3-х стадий деления опорной частоты для формирования заданных битового и кадрового синхроимпульсов. Схема генератора изображена на Рисунок 18–22.

- Делитель опорного сигнала (задается CLKDIV). Формирует сигнал битовой синхронизации CLKG.
- Формирователь кадрового синхросигнала (задается FPER). Задает 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.

Так же генератор позволяет осуществлять синхронизацию от внешнего кадрового импульса.

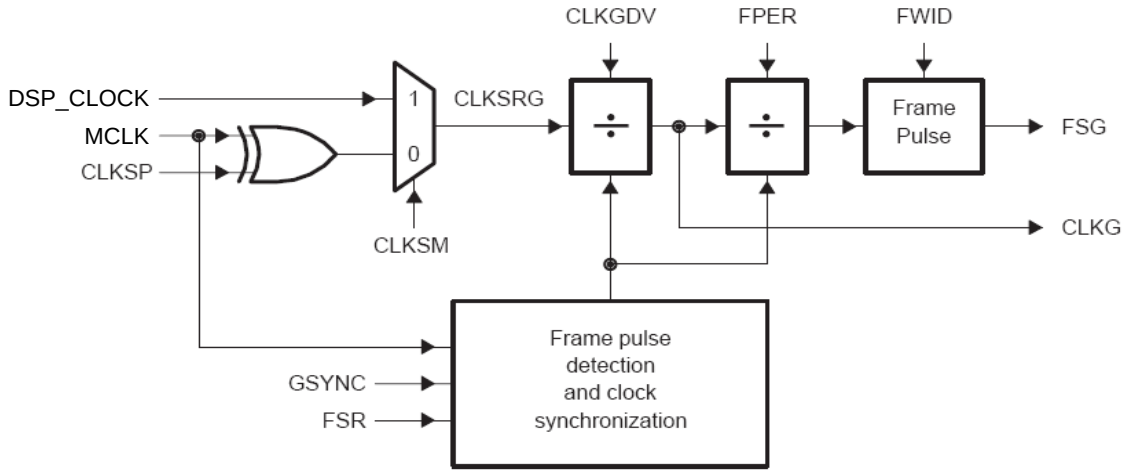


Рисунок 18–22 – Структурная схема генератора кадровой и битовой синхронизации

18.28.1 SRGRH

Таблица 18-16 – Регистр SRGRH

15	14	13	12	11	10	9	8
GSYNC	FSGM	CLKSM	CLKSP	FPER[11:8]			
R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0			
7	6	5	4	3	2	1	0
FPER[7:0]							
R/W,+0							

Таблица 18-17 – Описание бит регистра SRGRH

№ 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.
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

18.28.2 SRGRL

Таблица 18-18 – Регистр SRGRL

15	14	13	12	11	10	9	8
FWID[7:0]							
R/W,+0							

7	6	5	4	3	2	1	0
CLKGDV[7:0]							
R/W,+0							

Таблица 18-19 – Описание бит регистра SRGRL

№ 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

18.29 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 RRST = 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. Обождать 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.

18.30 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 * 100%.

18.31 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.

18.32 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 18–23 и Рисунок 18–24) приведены примеры диаграмм для разных настроек битовой и кадровой синхронизации.

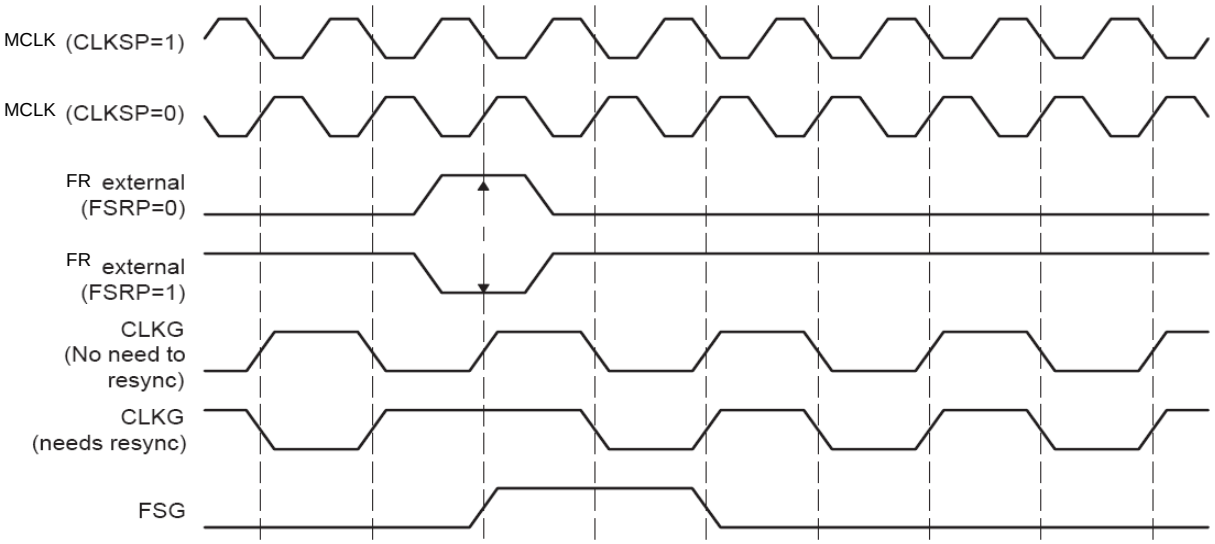


Рисунок 18–23

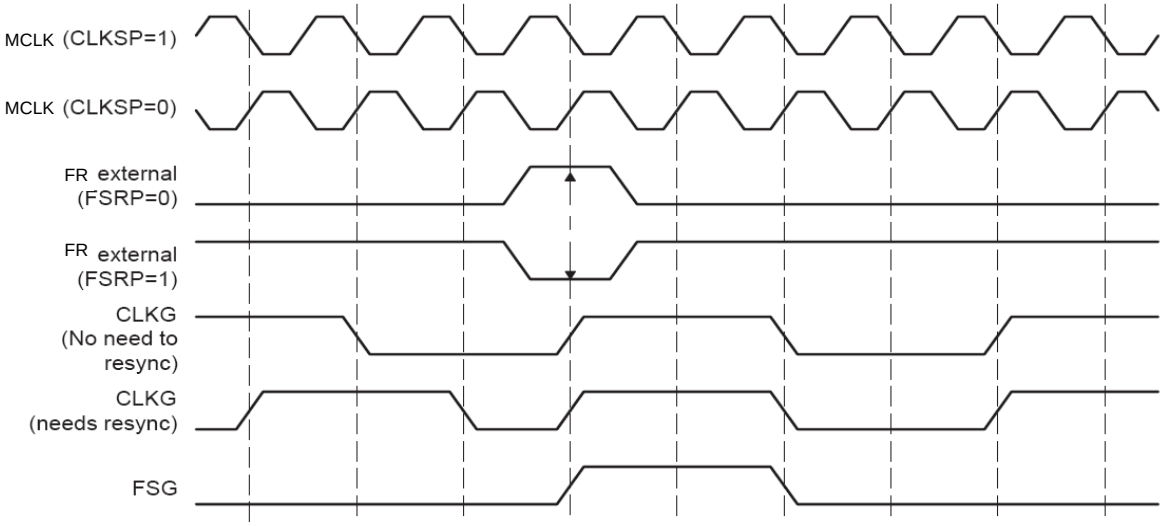


Рисунок 18–24

Так же 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.

18.33 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. Так же данный режим может быть использован для кодирования и декодирования массивов данных, а так же для выравнивания по заданной границе и расширения знака без участия ЦП.

18.34 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 т.о. для его работы необходима работа генератора битовой синхронизации (!).

18.35 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 быт соответственно и т.о. 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, т.е. К примеру длины синхроимпульса 1 битовый интервал соответствует значение 0 в поле SRGR.FWID.

Рекомендуется задавать длительность кадрового синхроимпульса менее длины передаваемого/принимаемого слова. Так же 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 18–25).

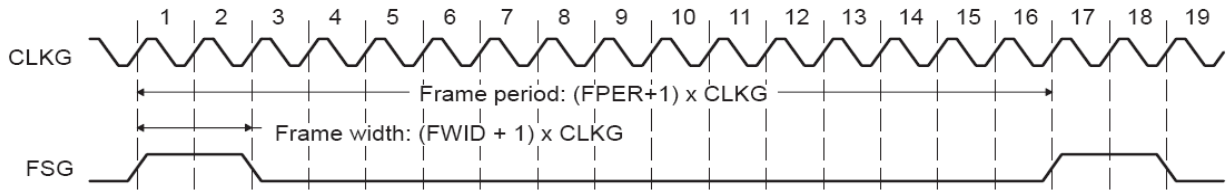


Рисунок 18–25 – Пример формирования кадровой синхронизации

18.36 Примеры битовой и кадровой синхронизации в различных форматах

В данном разделе приведены примеры настроек для некоторых форматов синхронных последовательных интерфейсов.

18.36.1 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 синхронизации 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.

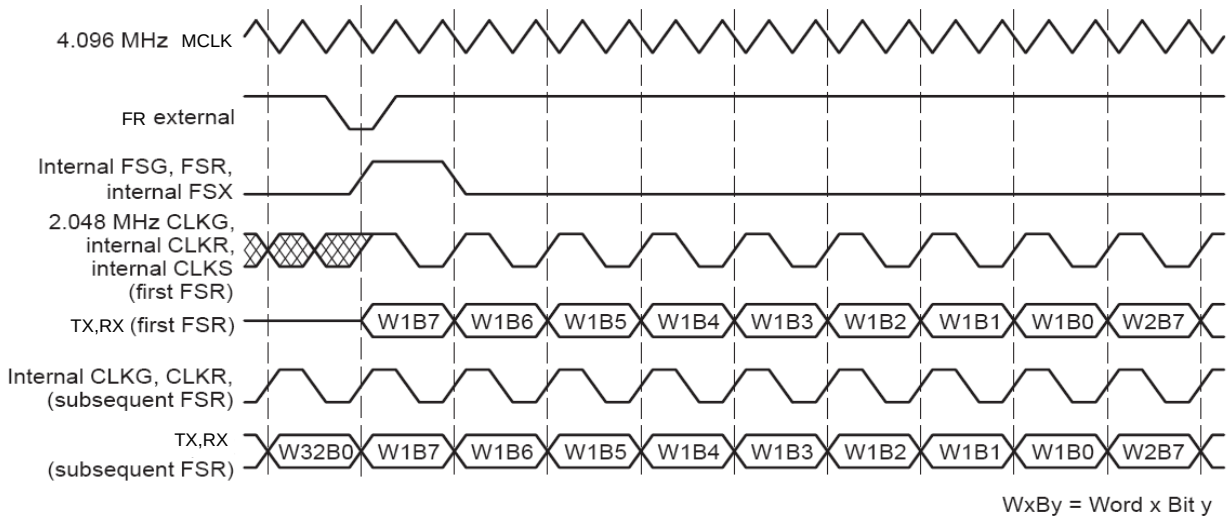


Рисунок 18–26 – Диаграмма работы в режиме ST-BUS

18.36.2 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 синхронизации 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.

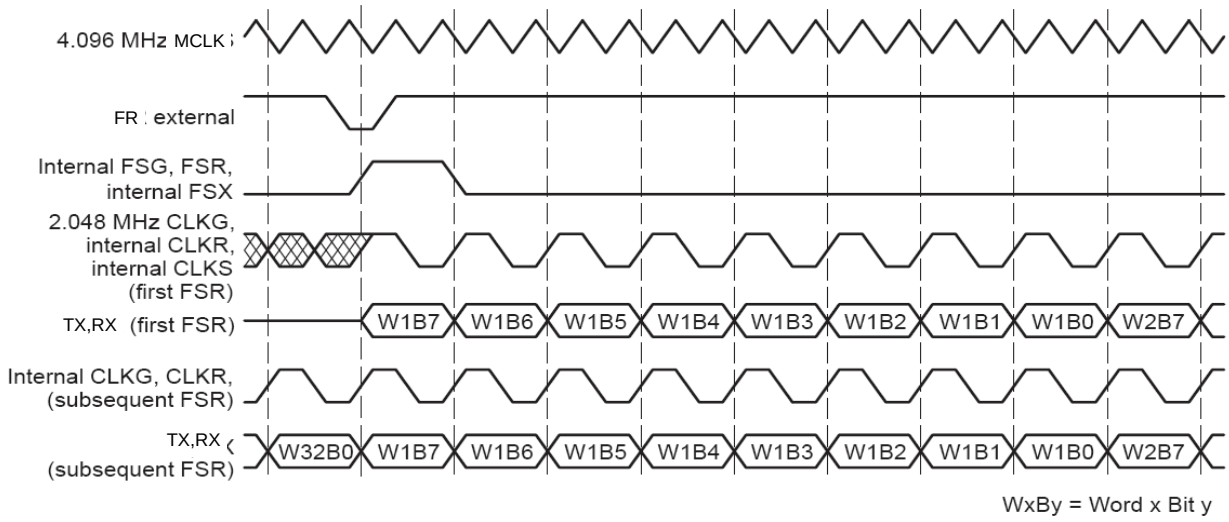


Рисунок 18–27 – Диаграмма работы в режиме ST-BUS с удвоенной частотой

18.37 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 кол-во 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 XCERH и RCERH. 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 так же не происходит.

При отключенных каналах отработка передач между сдвиговыми регистрами и FIFO осуществляется и все статусные признаки изменяются, кроме связанных с собственно приемом/передачей данных.

18.37.1 MCRH, MCRL, XMCR, RMCR

Таблица 18-20 – Регистры MCRH, MCRL, XMCR, RMCR

15	14	13	12	11	10	9	8
BLK7_EN	BLK6_EN	BLK5_EN	BLK4_EN	BLK3_EN	BLK2_EN	BLK1_EN	BLK0_EN
R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0
7	6	5	4	3	2	1	0
CUR_BLK						MCM	
R,+0			R/W,+0	R/W,+0	R/W,+0	R/W,+0	

Таблица 18-21 – Описание бит регистров MCRH, MCRL, XMCR, RMCR

№ 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 активен 1 – Блок каналов неактивен
7…5	CUR_BLK[2:0]	Номер текущего блока в кадре
4…2	-	
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

18.38 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 8 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.

На рисунке приведено распределение каналов и блоков в кадре.

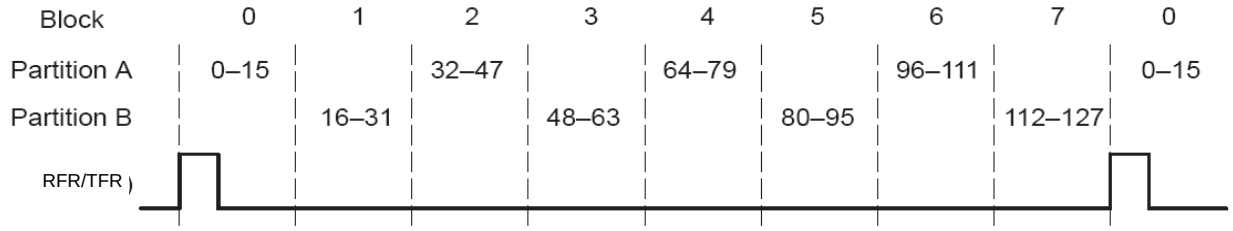


Рисунок 18–28

18.39 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 кол-ва каналов в кадре, то активность того или иного канала кроме так же определяется активностью или неактивностью блока (см выше). Регистр управления активность 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.

18.39.1 XCERH, RCERH

Таблица 18-22 – Регистры XCERH, RCERH

31[15]	**30**[14]	**29**[13]	**28**[12]	**27**[11]	**26**[10]	**25**[9]	**24**[8]
CH31_EN	CH30_EN	CH29_EN	CH28_EN	CH27_EN	CH26_EN	CH25_EN	CH24_EN
R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0
23[7]	**22**[6]	**21**[5]	**20**[4]	**19**[3]	**18**[2]	**17**[1]	**16**[0]
CH23_EN	CH22_EN	CH21_EN	CH20_EN	CH19_EN	CH18_EN	CH17_EN	CH16_EN
R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0

Таблица 18-23 – Описание бит регистров XCERH, RCERH

№ 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 активен 1 – Канал неактивен

18.39.2 XCERL, RCERL

Таблица 18-24 – Регистры XCERL, RCERL

15[15]	**14**[14]	**13**[13]	**12**[12]	**11**[11]	**10**[10]	**9**[9]	**8**[8]

CH15_EN	CH14_EN	CH13_EN	CH12_EN	CH11_EN	CH10_EN	CH9_EN	CH8_EN
R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0
7[7]	**6**[6]	**5**[5]	**4**[4]	**3**[3]	**2**[2]	**1**[1]	**0**[0]
CH7_EN	CH6_EN	CH5_EN	CH4_EN	CH3_EN	CH2_EN	CH1_EN	CH0_EN
R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0	R/W,+0

Таблица 18-25 – Описание бит регистров XCERL, RCERL

№ 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 активен 1 – Канал неактивен

18.39.3 Интерфейс A-bis

В режиме A-bis McBSP может принимать и передавать до 1024 бит ИКМ-тракта. Приемник *может* извлечь все 1024 бита из ИКМ-кадра в соответствии с заданной маской и выдать на ЦП прерывание при упаковке в слова до 16 бит.

19 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 RISC ядер, запросы для DMA DSP и DMA RISC). Временное разрешение таймера определяется частотой синхросигналов DSP. Высокий динамический диапазон таймера достигнут 16-битовым счетчиком с 4-битовым предварительным делителем частоты.

19.1 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 19-1 – Регистры таймера DSP подсистемы

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

19.1.1 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 19-2 – Регистр TIM

Номер	15...0
Доступ	RW
Сброс	0
	Текущее значение таймера

Таблица 19-3 – Биты регистра управления таймером TIM

№ Бита	Имя бита	Краткое описание назначения бита
15-0	TIM	Текущее значение таймера

19.1.2 Регистр периода таймера (PRD)

16-битный регистр периода таймера отображаемый в памяти (PRD), используется для того, чтобы перезагружать регистр таймера (TIM).

Таблица 19-4 – Регистр PRD

Номер	15...0
Доступ	RW
Сброс	0
	Начальное значение таймера

Таблица 19-5 – Биты регистра управления таймером PRD

№ Бита	Имя бита	Краткое описание назначения бита
15-0	PRD	Начальное значение таймера

19.1.3 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 19-6) и их описание (Таблица 19-7).

Таблица 19-6 – 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 19-7 – Биты регистра управления таймером TCR

№ Бита	Имя бита	Краткое описание назначения бита
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. TSS = 0 таймер работает. TSS = 1 таймер остановлен.
3-0	TDDR	Базовое значение предварительного счетчика.

20 Работа таймера (DSP)

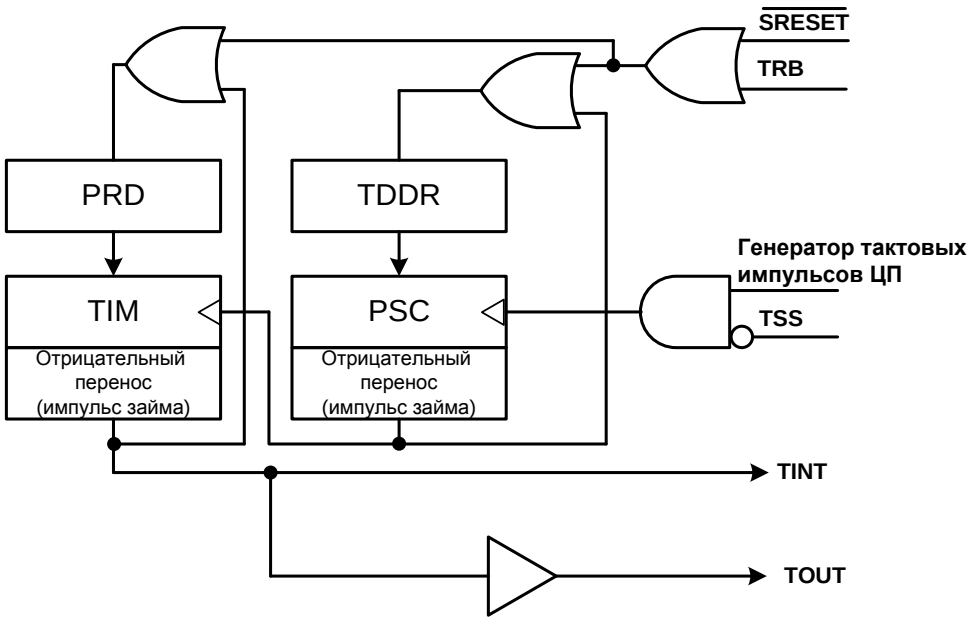


Рисунок 20–1 – Блок-схема таймера

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 выполненым, 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
tc(C) – период синхрогенератора DSP;
u – содержание TDDR плюс 1;
v – содержание PRD плюс 1.

21 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 реистров ядра DSP) (Таблица 21-1). При этом, в качестве адресов источника, назначения и управляющих структур используются система адресации RISC (т.е. общее адресное пространство и байтовые адреса).

21.1 Особенности контроллера DMA(DPS)

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 21-3). 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, програмные 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 21-2). 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 «Контроллер прямого доступа в память DMA RISC».

21.2 Возможности доступа DMA(DPS)

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 так же как в RISC подсистеме (32-х битный байтовый адрес). DMA контроллер DSP может быть использован для модификации памяти программ внутри DSP подсистемы.

Таблица 21-1 – 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**
0x3000_0040	*Регистры периферийных модулей RISC*	*NA*
0x5000_0000	*EXTERNAL BUS*	*NA*

Обозначения:
NA – нет доступа;
RO – только чтение;
WR – полный доступ (чтение/запись).

21.3 Регистры контроллера DMA(DPS)

Набор регистров и их функции контроллера DMA DSP такие же, как и контроллера DMA RISC. Все регистры могут быть доступны как со стороны RISC так и со стороны DSP (Таблица 21-2). Т.к. количество каналов контроллера сокращено, во всех регистрах доступны биты управления каналами с 1 по 16.

Таблица 21-2 – 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	
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	
0x3000_010C	0086h	ERR_CLRL	Регистр сброса флага ошибки
0x3000_010E	0087h	ERR_CLRH	Регистр сброса флага ошибки

21.4 Распределение аппаратных запросов по каналам DMA(DPS)

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 21-3 – Распределение аппаратных запросов DMA DSP по каналам

Номер канала	Источник reg	Источник sreg	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 Криптомодлуля
10	-	-	Только программные запросы
11	-	-	Только программные запросы
12	-	-	Только программные запросы
13	-	-	Только программные запросы
14	-	-	Только программные запросы
15	-	-	Только программные запросы

22 Контроллер AudioCodec (DSP)

Характеристики контроллера:

- Одноканальный ∑Δ АЦП с разрядность выходных отчетов 16 бит;
- Одноканальный ∑Δ 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 DMA каналы для тракта АЦП и ЦАП;
- Блок формирует сигнал прерывания по внутренним событиям;
- Имеет встроенный формирователь синхросигналов;
- ЦАП имеет защиту от короткого замыкание;
- Примечание: все измерения проводились для частоты отсчетов – 8 кГц.

Таблица 22-1 – ***Параметры каналов ЦАП/АЦП***

Параметр	Значение
Уровень “0” для ЦАП	1,5 В
Размах сигнала для ЦАП, p-p	2 В (max)
Уровень “0” для АЦП	1,5 В
Размах входного сигнала для АЦП	2 В (max)

22.1 Задание рабочей частоты аудиокодека

Рабочая частота аудиокодека задается при помощи регистра ADC_MCO_CLOCK (см. раздел “Сигналы тактовой частоты”). Запись данных в регистры осуществляется на частоте PCLK. Для пересинхронизации данных перед их обработкой цифровой частью аудиокодека реализован два раздельных FIFO интерфейса (для ЦАП и АЦП) по 16 16-ти битных отсчетов каждый.

22.2 Архитектура модуля

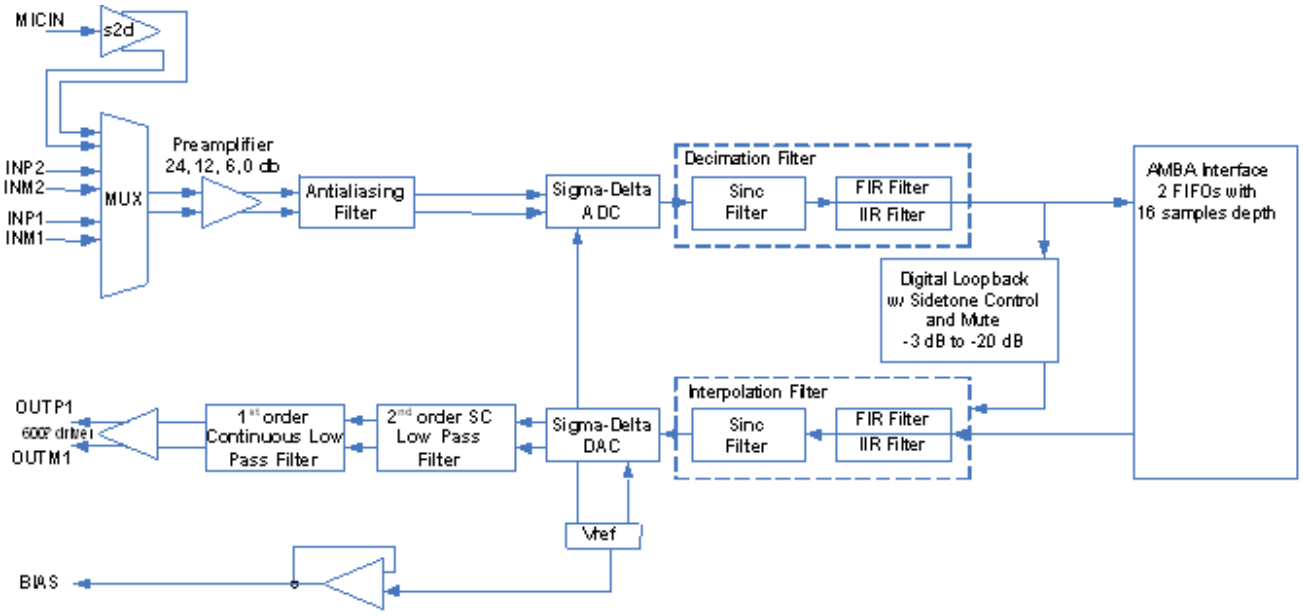


Рисунок 22–1 – Структура IP звукового кодека

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 IIREN 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

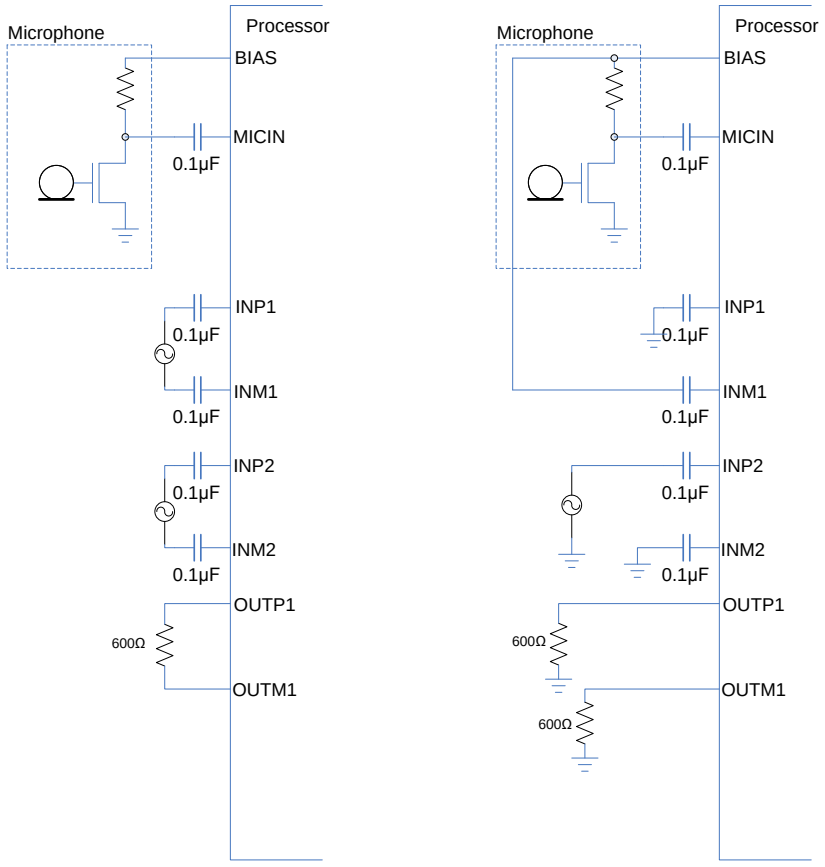
В модуле имеется встроенный программируемый усилитель для управления уровнями сигнала на выходе АЦП. Программируемый усилитель может быть установлен с использованием поля ADGAIN регистра ADCCTL. Диапазон усилителя канала АЦП составляет от 20 до -42 дБ с шагом 1 дБ. Чтобы избежать внезапных скачков на 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.

22.3 Аналоговые площадки ввода-вывода

Для работы модуля реализованы три программируемых аналоговых входа и один программируемый аналоговый выход (Таблица 22-2). 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.



(а) (б)

Рисунок 22–2 – Дифференциальное (а) и псевдо дифференциальное (б) подключение модуля

Таблица 22-2 – 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

22.3.1 Вход микрофона

Модуль поддерживает один не диференциальный 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 формирование смещения в 1,5 В. Для уменьшения шума в этом режиме необходимо подключить микрофонный вход, как показано на рисунке (Рисунок 22–3 а).

Запись значение 10 в поле AINSEL переводит аудиокодек в псевдо-диференциальный режим. В этом режиме недифференциальный вход объединяется с

микрофонным входом и выходом напряжения смещения BIAS (Рисунок 22–3 б). Для уменьшения помех выводы MICIN и INM1 должны иметь разводку одинаковой длины.

Недифференциальный вход преобразуется внутри в диференциальный перед преобразованием для лучшей защиты от шумов. Для улучшения динамических характеристик модуль поддерживает настраиваемый предусилитель. Управление предусилителем осуществляется при помощи поля INBG регистра ADCCTL.

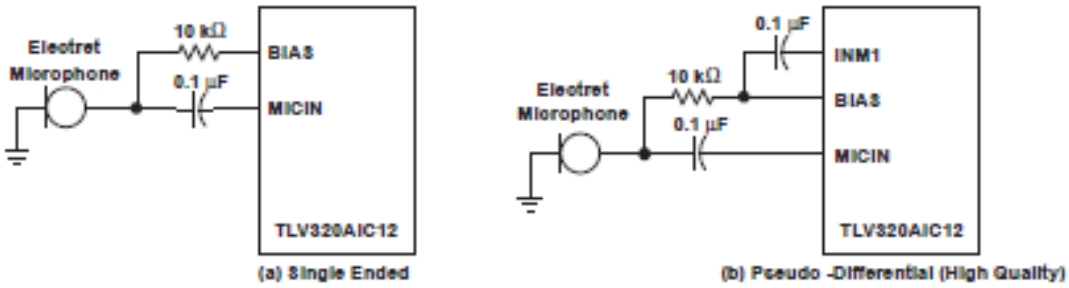


Рисунок 22–3 – Подключение MICIN

22.3.2 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 наилучшегоподавления шумов. Для получения максимального динамического диапазона, сигнал должен быть подключен ко входному терминалу как показано на Рисунок 22–4.

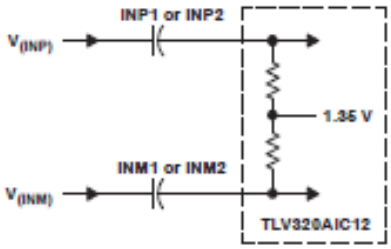


Рисунок 22–4 – Внутрення схема смещения INP и INM

Недифференциальный вход

Каждый из дифференциальных входов (INP1/2 и INM1/2) может быть подключен как недиференциальный. Для этого вывод INP соответвующеговхода необходимо подключитьк источнику сигнала, а вывод INM к земле (Рисунок 22–5).

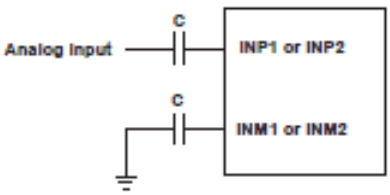


Рисунок 22–5 – Недифференциальный вход

22.3.3 Аналоговые выходы

Выходы OUTP и OUTM диференциальные выходы канала ЦАП. Выходы OUTP и OUTM могут обеспечить прямую нагрузку 600 ОМ и могут быть использованы как не дифференциальны выходы (Рисунок 22–6).

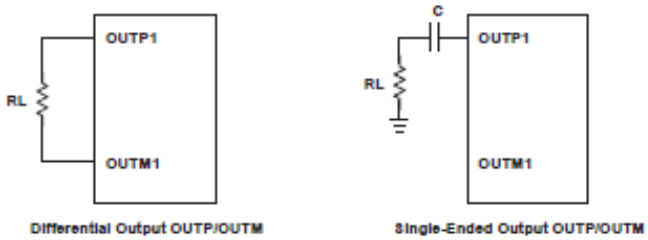


Рисунок 22–6 – Подключение выходовOUTP1/OUTM1

22.4 Работа с модулем

22.4.1 Воспроизведение звука (ЦАП)

Для воспроизведения потока данных необходимо:

1. Подать синхросигнал на DSP подсистему (см. “Сигналы тактовой частоты”).
2. Снять сброс с периферии DSP (см. “Организация управления DSP подсистемой”).
3. Подать синхросигнал на модуль Аудиокодек (см. “Сигналы тактовой частоты”).
4. Записать в FIFO DAC 16 отсчетов (регистр DACREG). В случае если на момент начала работы модуля оставить FIFO пустым, возможны сбои при дальнейшей работе.
5. Настроить DAC припомощи 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 пофлагу 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.

22.4.2 Оцифровка входного потока (АЦП)

Для оцифровки входного потока данных необходимо:

1. Подать синхросигнал на DSP подсистему (см. “Сигналы тактовой частоты”).
2. Снять сброс с периферии DSP (см. “Организация управления DSP подсистемой”).
3. Подать синхросигнал на модуль Аудиокодек (см. “Сигналы тактовой частоты”).

4. Настроить ADC припомощи регистра ADCCTL. В регистре задаются: бит включения АЦП, выбор источника для АЦП (см. “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. В том случае, если новые отсчеты не считаны, возможна потеря отсчетов и бит ADCVF регистра IRQFLAG будет взведен.

22.4.3 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. “Контроллер DMA (DSP)” и “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.

22.4.4 Прерывания модуля

Прерывания от модуля аудиокодек могут быть обработаны как при помощи RISC процессора, так и при помощи DSP процессора (см. разделы “Прерывания и исключения RISC” и “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 оциврованной значение FIFO ADC (В том случае если прерывание незамаскировано в регистре MASKCTL).

22.5 Регистры управления

Таблица 22-3 – 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

22.5.1 Регистр общего управления кодеком

Таблица 22-4 – 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 БИХ фильтра 0 – фильтр выключен 1 – фильтр включен

22.5.2 Регистр управления АЦП

Таблица 22-5 – Описание бит регистра управления АЦП

Бит	Поле	Нач. значение	Описание
5:0	ADGAIN	101010	Уровень записи (шаг – 1dB) 111111 : Mute 111110 : +20dB 111101 : +19dB … 101010 : 0dB 101001 : -1dB … 000000 : -42dB
7:6	INBG	00	Входной предусилитель 00 : 0dB 01 : 6dB 10 : 12dB 11 : 24 dB
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

22.5.3 Регистр управления ЦАП

Таблица 22-6 – Описание бит регистра управления ЦАП

Бит	Поле	Нач. значение	Описание
5:0	DAGAIN	101010	Уровень воспроизведения (шаг – 1dB) 111111 : Mute 111110 : +20dB 111101 : +19dB … 101010 : 0dB 101001 : -1dB … 000000 : -42dB
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.
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 – -21dB 110 – -18dB 110 – -15dB 110 – -12dB 110 – -9dB 110 – -6dB 000 – -3dB
15	DACRES	1	Сброс ЦАП (аналоговая часть). 0 – ЦАП находится в сбросе. 1 – ЦАП находится в рабочем режиме.

22.5.3.1 Маска вектора прерываний

Таблица 22-7 – 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

22.5.4 Флаги прерываний

Таблица 22-8 – 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

22.5.4.1 АЦП FIFO регистр

Таблица 22-9 – Описание бит регистра АЦП FIFO

Бит	Поле	Нач. значение	Описание
15:0	ADCSMP	00	Новый отчет тракта АЦП

22.5.4.2 ЦАП FIFO регистр

Таблица 22-10 – Описание бит регистра ЦАП FIFO

Бит	Поле	Нач. значение	Описание
15:0	DACSMP	00	Новый отчет тракта ЦАП

23 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 микроконтроллера блок шифрования представляет собой набор программно доступных 16-разрядных регистров (Таблица 23-1).

23.1 Описание регистров

Таблица 23-1 – Описание регистров блока шифрования

Адрес RISC	Адрес DSP	Название регистра	Описание
0x3000_0080	0x0040	CRPT_CWR	регистра управления
0x3000_0084	0x0042	CRPT_SR	регистра состояния
0x3000_0088	0x0044	CRPT_DATA	регистра данных
0x3000_008C	0x0046	CRPT_KR	регистра ключа
0x3000_0090	0x0048	CRPT_CR	регистра констант замены
0x3000_0094	0x004A	CRPT_SYNR	регистра синхропосылки
0x3000_0098	0x004C	CRPT_IMIT	регистра имитовставки
0x3000_009C	0x004E	CRPT_ITER	регистра 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 (8 байт) и имитовставки (8 байт).

Блок шифрования содержит два практически одинаковых модуля, одни 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.

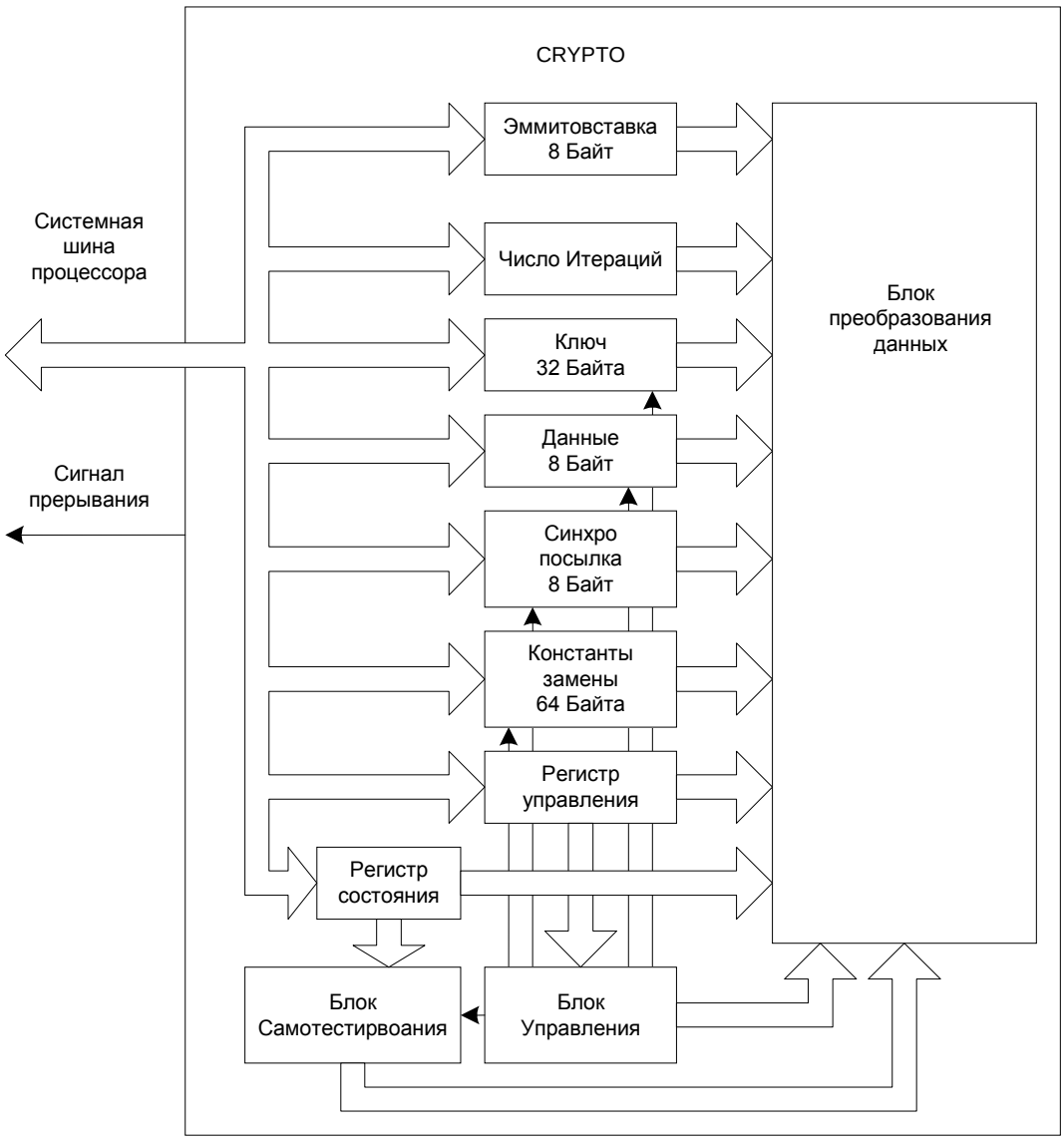


Рисунок 23–1 – Блок-схема блока аппаратной поддержки алгоритма шифрования по ГОСТ 28147-89

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 дополнятся 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 одни старт преобразования для завершения расчета имитовставки. При этом требуется установить бит IM в регистре CRPT_CWR, а новые данные посылать не требуется.

23.1.1 Регистр управления CRPT_CWR

Таблица 23-2 – Описние бит регистра управления CRPT_CWR

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
бит 7	6	5	4	3	2	1	бит 0
бит 7		BIST: Тестовый режим работы.					
бит 6		RST: Сброс – предназначен для приведения устройства в исходное состояние. Примечание: бит «RST» сбрасывается автоматически в следующем цикле системной тактовой частоты.					
бит 5		IE_CRPT: Разрешение прерывания. Используется для разрешения («1») или запрета («0») выдачи сигнала прерывания.					
бит 4		DIR: Направление шифрации: «0» – шифрация, «1» – дешифрация.					
бит 3		START: Старт. Предназначен для запуска процесса шифрации/дешифрации данных. Примечание: бит «START» сбрасывается автоматически после начала преобразования.					
бит 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.					
бит 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.					

Обозначения здесь и далее по тексту:
R – бит для чтения;
W – бит с возможностью записи;
U – бит не реализован, читается как 0;
-n – значение бита после сброса по включению питания:
1 – установлен;
0 – сброшен;
x – значение не известно.

23.1.2 Регистр состояния CRPT_SR

Таблица 23-3 – Описние бит регистра состояния CRPT_SR

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
DNC_BIS T	KW(*)	SW(*)	DR(*)	DW(*)	KF(*)	ERROR_ CRPT	READY_ CRPT
бит 7	6	5	4	3	2	1	бит 0
бит 7		DNC_BIST: Результат динамического контроля/самотестирования.					
бит 6		KW(*): Состояние указателя чтения памяти имитоданных.					
бит 5		SW(*): Состояние указателя записи памяти синхропосылки.					
бит 4		DR(*): Состояние указателя чтения памяти данных.					

бит 3	DW(*): Состояние указателя записи памяти данных.
бит 2	KF(*): Состояние указателя записи памяти ключа.
бит 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.
бит 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.

Примечание:

(*) 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).

23.1.3 Регистр данных шифрации CRPT_DATA

Таблица 23-4 – Описние бит регистра данных шифрации CRPT_DATA

R/W-0	…	R/W-0
DATA15	…	DATA0
бит 15		бит 0
бит 15…0	DATA15…DATA0: 16-ти 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-ти разрядных слов по адресу 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, т.е. при установке бита «RST».	

23.1.4 Регистр ключа шифрации CRPT_KR

Таблица 23-5 – Описние бит регистра ключа шифрации CRPT_KR

R/W-0	…	R/W-0
KEY15	…	KEY0
бит 15	…	бит 0
бит 15…0	KEY15… KEY0: 8-ми 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-ти разрядных слов по адресу регистра ключа, начиная с младшего байта.	

23.1.5 Регистр синхропосылки CRPT_SYNR

Таблица 23-6 – Описние бит регистра синхропосылки CRPT_SYNR

R/W-0	…	R/W-0
SYNCH15	…	SYNCH0
бит 15	…	бит 0
бит 15…0	SYNCH15…SYNCH0: 16-ти 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-ти разрядных слов по адресу регистра синхропосылки, начиная с младшего байта. Чтение осуществляется аналогично.	

23.1.6 Регистр констант замены CRPT_CR

Таблица 23-7 – Описние бит регистра констант замены CRPT_CR

R/W-0		R/W-0
CONST15		CONST0
бит 15		бит 0
бит 15…0	CONST15…CONST0: 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 4 константы. Заполнение узлов замены осуществляется, начиная с 1 узла и заканчивая узлом 8.	

23.1.7 Регистр имитовставки CRPT_IMIT

Таблица 23-8 – Описние бит регистра имитовставки CRPT_IMIT

R/W-0		R/W-0
IMIT15		IMIT0
бит 15		бит 0
бит 15…0	IMIT15… IMIT0: 16-ти 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.	

23.1.8 Регистр числа итераций CRPT_ITER

Таблица 23-9 – Описние бит регистра числа итераций CRPT_ITER

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_CRP	ITER5	ITER4	ITER3	ITER2	ITER1	ITER0

	Т						
бит 7	6	5	4	3	2	1	бит 0
бит 7		зарезервировано.					
бит 6		EN_CRPT: Бит разрешения работы блока. «0» – блок аппаратной поддержки алгоритма кодирования выключен; «1» – блок включен.					
бит 5…0		ITER5…ITER0: Регистр счетчика итераций циклов шифрации/дешифрации.					

Примечание – Бит EN_CRPT меняется только в том случае если бит DNC_BIST регистра CRPT_CWR сброшен.

23.2 Работа блока по шифрованию и дешифрованию данных

23.2.1 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 п.7.;
9. прочитать имитовставку из порта имитовставки.

Примечание – Пункты 1 и 2 (программный сброс и установка режимов) можно выполнять одновременно (одной командой).

23.2.2 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.

23.2.3 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.

23.2.4 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.

23.2.5 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.

23.2.6 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.

23.2.7 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.

23.2.8 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 представлена в Таблица 23-11.

В приложении А ГОСТ Р 34.11-94 на странице 7 приведён ключ замены:

K1= 733D2C20 65686573 74746769 79676120

626E7373 20657369 326C6568 33206D54

Порядок занесения этого ключа в регистр CRPT_KR приведен в Таблица 23-12.

Таблица 23-10 – 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 23-11 – 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 23-12 – 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

24 Контроллер интерфейса MDR_USB

Контролер USB реализует функции контроллера функционального устройства (Device) и управляющего устройства (Host) в соответствии со спецификацией USB 2.0.

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 1-й до 4-х 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.

24.1 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 2…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 USB интерфейса.

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).

24.2 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 TXLC. 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_HIS устанавливается в 1 при возникновении одного из событий.

24.3 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 EPISOEN в соответствующем регистре SEP[x].CTRL.

24.4 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 24–1 а). Бит NAKSENT регистра SEP[x].STS устанавливается в 1.

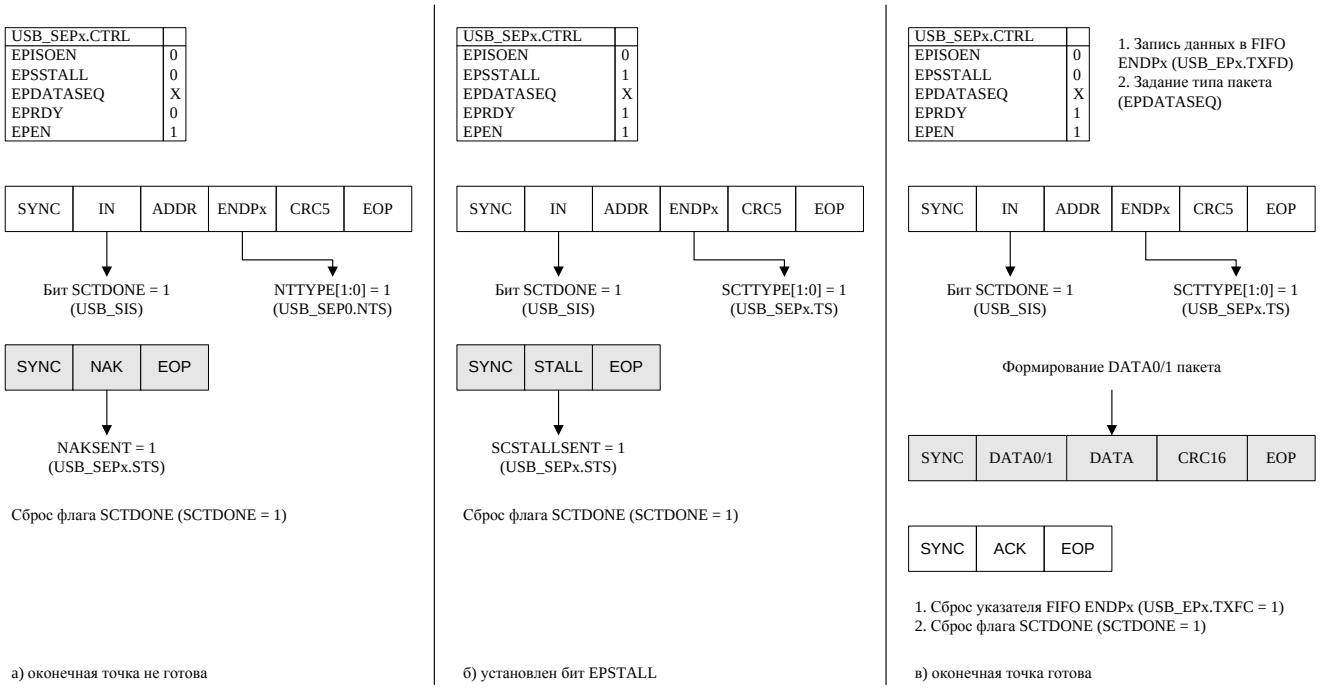


Рисунок 24–1 (а, б, в) – Транзакция IN (USB Device)

Если оконечная точка готова и установлен бит EPSSTALL в регистре SEP[x].CTRL, то контроллер отправляет STALL пакет (Рисунок 24–1 б). Бит SCSTALLSENT регистра SEP[x].STS устанавливается в 1.

Если оконечная точка готова (Рисунок 24–1 в), биты SCTTYPE[1:0] в регистре SEP[x].TS устанавливаются в значение 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].TXFC 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.

24.5 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 24–2 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 24–2 б). Бит SCSTALLSENT регистра SEP[x].STS устанавливается в 1.

Если оконечная точка готова (Рисунок 24–2 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.

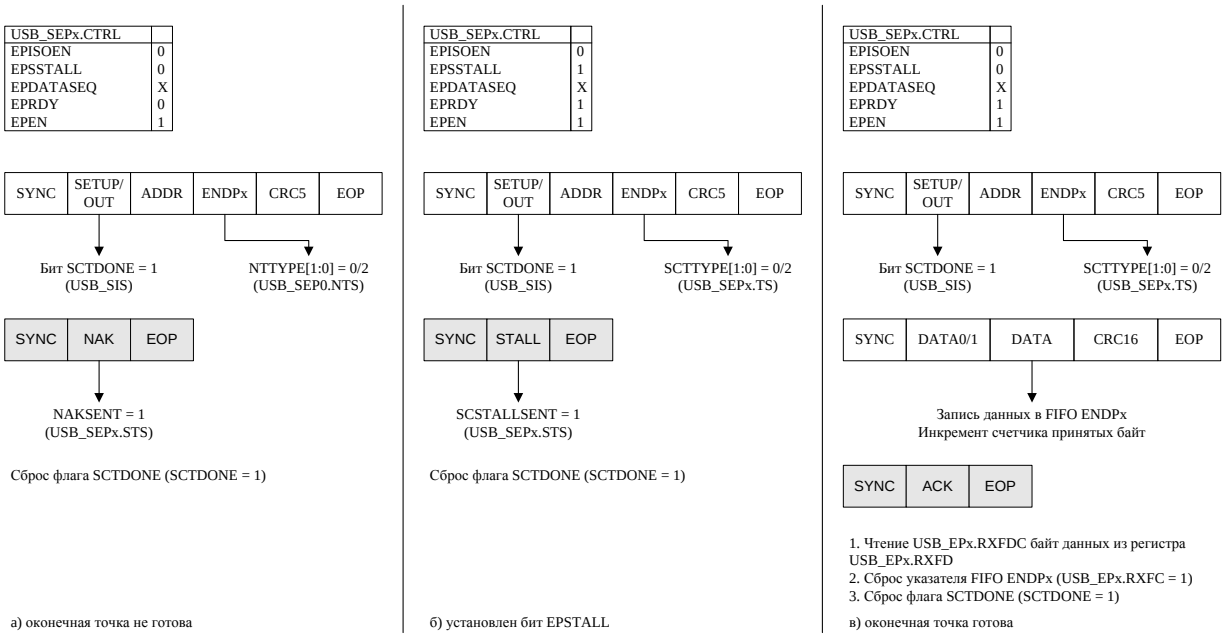


Рисунок 24–2 (а, б, в) – Транзакция SETUP/OUT (USB Device)

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[x].RXFD. Количество принятых байтов содержится в регистре EP[x].RXFDC. После приема очередного пакета необходимо выполнять сброс указателя FIFO приема записью 1 в регистр EP[x].RXFC.

После окончания транзакции бит SCTDONE регистра SIS должен быть очищен записью 1.

24.6 Транзакция SETUP/OUT (USB Host)

Для начала транзакции должны быть заданы адрес устройства (регистр HTXA), оконечная точка (регистр HTXE) и тип token пакета (регистр HTXT). Данные записываются побайтно в регистр HTXFD. Максимальный размер передаваемого пакета составляет 64 байтов. Попытка записи более 64 байтов подряд приведет к переполнению FIFO. Запись 1 в HTXFDC сбрасывает указатель FIFO передачи в 0. Перед началом формирования очередного пакета необходимо выполнять сброс указателя FIFO. Транзакция запускается при установке бита TREQ регистра HTXC. Host отправляет пакет Setup/Out и пакет данных.

После окончания транзакции бит TDONE = 1 (регистр HIS). 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 24–3 а), то бит NAKRXED = 1 (регистр HRXS).

Если в ответ получен пакет STALL (Рисунок 24–3 б), то бит STALLRXED = 1 (регистр HRXS).

Если в ответ получен пакет ACK (Рисунок 24–3 в), то бит ACKRXED = 1 (регистр HRXS).

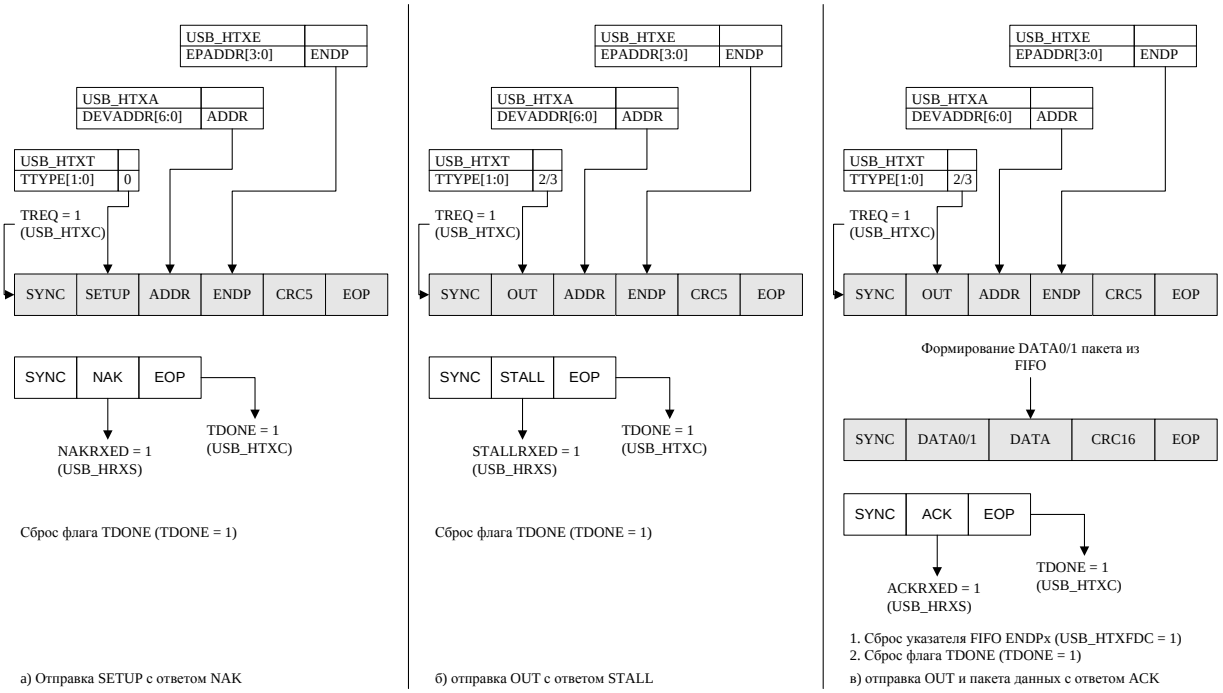


Рисунок 24–3 (а, б, в) – Транзакция SETUP/OUT (USB Host)

24.7 Транзакция IN (USB Host)

Для начала транзакции должны быть заданы адрес устройства (регистр HTXA), оконечная точка (регистр HTXE) и тип token пакета (регистр HTXT). Транзакция запускается при установке бита TREQ регистра HTXC. Host отправляет IN пакет.

После окончания транзакции бит TDONE = 1 (регистр HIS). 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 24–4 а), то бит NAKRXED = 1 (регистр HRXS).

Если в ответ получен пакет STALL (Рисунок 24–4 б), то бит STALLRXED = 1 (регистр HRXS).

Если приходит DATA0/DATA1 пакет (Рисунок 24–4 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).

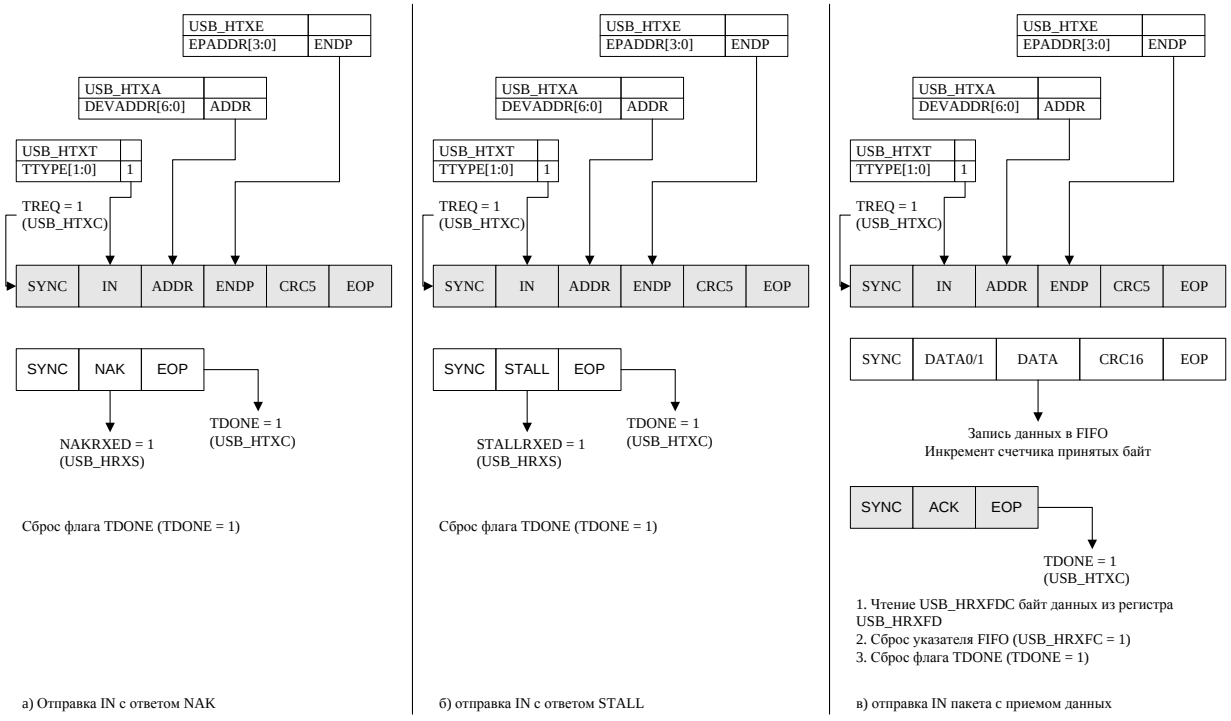


Рисунок 24–4 (а, б, в) – Транзакция IN (USB Host)

24.8 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 отсылаться каждые 1 мс. Если FSPL = 0, то автоматически будет отправляться EOP каждые 1 мс.

После отправки SOF пакета бит SOFS = 1 (регистр HIS). 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.

24.9 Описание регистров управление контроллером USB интерфейса

Таблица 24-1 – Описание регистров управление контроллером USB интерфейса

Базовый адрес	Название	Описание
0x4001_0000	MDR_USB	Контроллер USB интерфейса
Смещение		
0x380	MDR_USB->HSCR	Общее управление для контроллера USB интерфейса
0x384	MDR_USB->HSVR	Версия аппаратного контроллера USB интерфейса
	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 оконечной точки для отправки пакета
0x18 0x1C	MDR_USB->HFN_L MDR_USB->HFN_H	Регистр задания номера фрейма для отправки SOF
0x20	MDR_USB->HIS	Регистр флагов событий контроллера хост.
0x24	MDR_USB->HIM	Регистра флагов разрешения прерываний по событиям контролера 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 оконечной 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 0x8C	MDR_USB->HRXDC_L MDR_USB->HRXDC_H	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
	Контроллер SLAVE	
0x100 0x110 0x120 0x130	MDR_USB->SEP[0].CTRL MDR_USB->SEP[1].CTRL , MDR_USB->SEP[2].CTRL , MDR_USB->SEP[3].CTRL	Управление очередью нулевой оконечной точки
0x104 0x114 0x124 0x134	MDR_USB->SEP[0].STS , MDR_USB->SEP[1].STS , MDR_USB->SEP[2].STS , MDR_USB->SEP[3].STS	Состояние оконечной точки
0x108 0x118 0x128 0x138	MDR_USB->SEP[0].TS , MDR_USB->SEP[1].TS , MDR_USB->SEP[2].TS , MDR_USB->SEP[3].TS	Состояние типа передачи оконечной точки
0x10C 0x11C 0x12C 0x13C	MDR_USB->SEP[0].NTS , MDR_USB->SEP[1].NTS , MDR_USB->SEP[2].NTS , MDR_USB->SEP[3].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		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[0].RXFD, MDR_USB->SEP[1].RXFD, MDR_USB->SEP[2].RXFD, MDR_USB->SEP[3].RXFD	Принятые данные оконечной точки
0x188 0x18C 0x208 0x20C 0x288 0x28C 0x308 0x30C	MDR_USB->SEP[0].RXFDC_L MDR_USB->SEP[0].RXFDC_H , MDR_USB->SEP[1].RXFDC_L , MDR_USB->SEP[1].RXFDC_H , MDR_USB->SEP[2].RXFDC_L , MDR_USB->SEP[2].RXFDC_H , MDR_USB->SEP[3].RXFDC_L , MDR_USB->SEP[3].RXFDC_H	Число данных в оконечной точке
0x190 0x210 0x290 0x310	MDR_USB->SEP[0].RXFC, MDR_USB->SEP[1].RXFC, MDR_USB->SEP[2].RXFC, MDR_USB->SEP[3].RXFC	Управление очередью приема оконечной точки

0x1C0 0x240 0x2C0 0x340	MDR_USB->SEP[0].TXFD , MDR_USB->SEP[1].TXFD , MDR_USB->SEP[2].TXFD , MDR_USB->SEP[3].TXFD	Данные для передачи через оконечную точку
0x1D0 0x250 0x2D0 0x350	MDR_USB- >SEP[0].TXFDC MDR_USB- >SEP[1].TXFDC MDR_USB- >SEP[2].TXFDC MDR_USB- >SEP[3].TXFDC	Управление очередью передачи оконечной точки

24.9.1 MDR_USB->HSCR

Таблица 24-2 – 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 24-3 – Описание бит регистра HSCR

№ 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

24.9.2 MDR_USB->HSVR

Таблица 24-4 – Регистр HSVR

Номер	31...8	7...4	3...0
Доступ	U	RO	RO
Сброс	0	0	0
	-	**REVISION**	**VERSION**

Таблица 24-5 – Описание бит регистра HSVR

№ 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

24.9.3 MDR_USB->HTXC

Таблица 24-6 – 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 24-7 – Описание бит регистра HTXC

№ 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. Изохронный режим не применим ни с какими другими типами передачи; 0 – запрещение изохронного режима
2	PREEN	Флаг разрешения преамбулы: 1 – разрешение преамбулы. Должна быть установлена только когда host подсоединен к low speed устройству через хаб. Преамбула – это заголовок перед всеми пакетами передачи и передается на full speed независимо от состояния FULL_SPEED_LINE_RATE_BIT.
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

24.9.4 MDR_USB->HTXT

Таблица 24-8 – Регистр HTXT

Номер	31...2	1	0
Доступ	U	R/W	R/W
Сброс	0	0	0
	-	**TTYPE**	

Таблица 24-9 – Описание бит регистра HTXT

№ 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

24.9.5 MDR_USB->HTXLC

Таблица 24-10 – Регистр HTXLC

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
	-	**FSLR**	**FSLP**	**DC**	**TXLS[1:0]**	

Таблица 24-11 – Описание бит регистра HTXLC

№ 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, если работа ведется с low speed через хаб, то должна быть установлена full speed полярность
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+

24.9.6 MDR_USB->HTXSE

Таблица 24-12 – Регистр HTXSE

Номер	31…1	0
Доступ	U	R/W
Сброс	0	0
	-	**SOFEN**

Таблица 24-13 – Описание бит регистра HTXSE

№ 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 отсылаться каждые 1 мс. SOF отправляется на full speed независимо от состояние 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

24.9.7 MDR_USB->HTXA

Таблица 24-14 – Регистр HTXA

Номер	31...7	6...0
Доступ	U	R/W
Сброс	0	0
	-	**DEVADDR[6:0]**

Таблица 24-15 – Описание бит регистра HTXA

№ 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

24.9.8 MDR_USB->HTXE

Таблица 24-16 – Регистр HTXE

Номер	31...4	3...0
Доступ	U	R/W
Сброс	0	0
	-	**EPADDR[3:0]**

Таблица 24-17 – Описание бит регистра HTXE

№ 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 оконечной точки устройства для обращения

24.9.9 MDR_USB->HFN

Таблица 24-18 – Регистр HFN

Номер	31...11	10…0
Доступ	U	R/W
Сброс	0	0
	-	**FNUM[10:0]**

Таблица 24-19 – Описание бит регистра HFN

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31...11	-	Зарезервировано
10…0	FNUM[10:0]	Номер фрейма

24.9.10 MDR_USB->HIS

Таблица 24-20 – Регистр HIS

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 24-21 – Описание бит регистра HIS

№ 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 подсоединение 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

24.9.11 MDR_USB->HIM

Таблица 24-22 – 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 24-23 – Описание бит регистра HIM

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...4	-	Зарезервировано
3	SOFIE	1 – разрешение выработки прерывание при окончании передачи. 0 – запрещение выработки прерывания
2	CONEVIE	1 – разрешение выработки прерывание при повторе передачи. 0 – запрещение выработки прерывания
1	RESUMEIE	1 – разрешение выработки прерывание при подсоединении или отсоединении. 0 – запрещение выработки прерывания
0	TDONEIE	1 – разрешение выработки прерывание при передаче SOF. 0 – запрещение выработки прерывания

24.9.12 MDR_USB->HRXS

Таблица 24-24 – 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 24-25 – Описание бит регистра HRXS

№ 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

24.9.13 MDR_USB->HRXP

Таблица 24-26 – Регистр HRXP

Номер	31...4	3…0
Доступ	U	R/W
Сброс	0	0
	-	**RPID[3:0]**

Таблица 24-27 – Описание бит регистра HRXP

№ 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

24.9.14 MDR_USB->HRXA

Таблица 24-28 – Регистр HRXA

Номер	31…7	6…0
Доступ	U	R/W
Сброс	0	0
	-	**RADDR[6:0]**

Таблица 24-29 – Описание бит регистра HRXA

№ 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

24.9.15 MDR_USB->HRXE

Таблица 24-30 – Регистр HRXE

Номер	31...4	3...0
Доступ	U	R/W
Сброс	0	0
	-	**RXENDP[3:0]**

Таблица 24-31 – Описание бит регистра HRXE

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений.**
31...4	-	Зарезервировано
3…0	RXENDP[3:0]	Номер оконечной точки в последнем принятом пакете, который был послан

24.9.16 MDR_USB->HRXCS

Таблица 24-32 – Регистр HRXCS

Номер	31...2	1	0
Доступ	U	R/W	R/W
Сброс	0	0	0
	-	**RXLS[1:0]**	

Таблица 24-33 – Описание бит регистра HRXCS

№ 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

24.9.17 MDR_USB->HSTM

Таблица 24-34 – Регистр HSTM

Номер	31...8	7...0
Доступ	U	R/W
Сброс	0	0
	-	**HSTM[7:0]**

Таблица 24-35 – Описание бит регистра HSTM

№ 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

24.9.18 MDR_USB->HRXFD

Таблица 24-36 – Регистр HRXFD

Номер	31...8	7...0
Доступ	U	R/W
Сброс	0	0
	-	**RX FIFO DATA[7:0]**

Таблица 24-37 – Описание бит регистра HRXFD

№ 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

24.9.19 MDR_USB->HRXDC

Таблица 24-38 – Регистр HRXDC

Номер	31...16	15...0
Доступ	U	R/W
Сброс	0	0
	-	**FIFO DATA COUNT[15:0]**

Таблица 24-39 – Описание бит регистра HRXDC

№ 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

24.9.20 MDR_USB->HRXFC

Таблица 24-40 – Регистр HRXFC

Номер	31...1	0
Доступ	U	R/W
Сброс	0	0
	-	**FIFO FORCE EMPTY**

Таблица 24-41 – Описание бит регистра HRXFC

№ 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

24.9.21 MDR_USB->HTXFD

Таблица 24-42 – Регистр HTXFD

Номер	31...8	7...0
Доступ	U	R/W
Сброс	0	0
	-	**TX FIFO DATA[7:0]**

Таблица 24-43 – Описание бит регистра HTXFD

№ 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

24.9.22 MDR_USB->HTXFC

Таблица 24-44 – Регистр HTXFC

Номер	31...1	0
Доступ	U	R/W
Сброс	0	0
	-	**FIFO FORCE EMPTY**

Таблица 24-45 – Описание бит регистра HTXFC

№ 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)

24.9.23 MDR_USB->SEP[0].CTRL, MDR_USB->SEP[1].CTRL, MDR_USB->SEP[2].CTRL, MDR_USB->SEP[3].CTRL

Таблица 24-46 – 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 24-47 – Описание бит регистра USB_SEPx.CTRL

№ 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

24.9.24 MDR_USB->SEP[0].STS, MDR_USB->SEP[1].STS, MDR_USB->SEP[2].STS, MDR_USB->SEP[3].STS

Таблица 24-48 – 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 24-49 – Описание бит регистра USB_SEPx.STS

№ 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

24.9.25 MDR_USB->SEP[0].TS, MDR_USB->SEP[1].TS, MDR_USB->SEP[2].TS, MDR_USB->SEP[3].TS

Таблица 24-50 – Регистр SEP[x].TS

Номер	31…2	1	0
Доступ	U	R/W	R/W
Сброс	0	0	0
	-	**SCTTYPE[1:0]**	

Таблица 24-51 – Описание бит регистра SEP[x].TS

№ 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

24.9.26 MDR_USB->SEP[0].NTS, MDR_USB->SEP[1].NTS, MDR_USB->SEP[2].NTS, MDR_USB->SEP[3].NTS

Таблица 24-52 – Регистр SEP[x].NTS

Номер	31…2	1	0
Доступ	U	R/W	R/W
Сброс	0	0	0
	-	**NTTYPE[1:0]**	

Таблица 24-53 – Описание бит регистра USB_SEPx.NTS

№ 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

24.9.27 MDR_USB->SC

Таблица 24-54 – Регистр SC

Номер	31…6	5	4	3	2	1	0
Доступ	U	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0	0
	-	**SCFSR**	**SCFSP**	**SCDC**	**SCTXLS[1:0]**		**SCGEN**

Таблица 24-55 – Описание бит регистра USB_SC

№ 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 USB шины: 1 – FULL SPEED; 0 –LOW SPEED
3	SCDC	Флаг прямого управления линиями USB шины: 1 – разрешено прямое управление 0 – запрещено прямое управление
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 оконечных точек 0 – все оконечные точки запрещены

24.9.28 MDR_USB->SLS

Таблица 24-56 – Регистр SLS

Номер	31…2	1	0
Доступ	U	R/W	R/W
Сброс	0	0	0
	-	**SCRXLS[1:0]**	

Таблица 24-57 – Описание бит регистра SLS

№ 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

24.9.29 MDR_USB->SIS

Таблица 24-58 – Регистр SIS

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**	**SC SOF REC**	**SC RESET EV**	**SC RESUME**	**SC TDONE**

Таблица 24-59 – Описание бит регистра USB_SIS

№ 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
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

24.9.30 MDR_USB->SIM

Таблица 24-60 – 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 24-61 – Описание бит регистра B_SIM

№ 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

24.9.31 MDR_USB->SA

Таблица 24-62 – Регистр SA

Номер	31…7	6…0
Доступ	U	R/W
Сброс	0	0
	-	**SDEVADDR[6:0]**

Таблица 24-63 – Описание бит регистра SA

№ 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

24.9.32 MDR_USB->SFN

Таблица 24-64 – Регистр SFN

Номер	31…11	10…0
Доступ	U	R/W
Сброс	0	0
	-	**FRAME NUM [10:0]**

Таблица 24-65 – Описание бит регистра SFN

№ 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

24.9.33 MDR_USB->SEP[0].RXFD, MDR_USB->SEP[1].RXFD, MDR_USB->SEP[2].RXFD, MDR_USB->SEP[3].RXFD

Таблица 24-66 – Регистр SEP[x].RXFD

Номер	31…8	7…0
Доступ	U	R/W
Сброс	0	0
	-	**RX FIFO DATA[7:0]**

Таблица 24-67 – Описание бит регистра SEP[x].RXFD

№ 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

24.9.34 MDR_USB->SEP[0].RXFDC, MDR_USB->SEP[1].RXFDC, MDR_USB->SEP[2].RXFDC, MDR_USB->SEP[3].RXFDC

Таблица 24-68 – Регистр SEP[x].RXFDC

Номер	31…16	15…0
Доступ	U	R/W
Сброс	0	0
	-	FIFO DATA COUNT [15:0]

Таблица 24-69 – Описание бит регистра SEP[x].RXFDC

№ 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

24.9.35 MDR_USB->SEP[0].RXFC, MDR_USB->SEP[1].RXFC, MDR_USB->SEP[2].RXFC, MDR_USB->SEP[3].RXFC

Таблица 24-70 – Регистр SEP[x].RXFC

Номер	31…1	0
Доступ	U	R/W
Сброс	0	0
	-	FIFO FORCE EMPTY

Таблица 24-71 – Описание бит регистра USB_SEPx.RXFC

№ 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

24.9.36 MDR_USB->SEP[0].TXFD, MDR_USB->SEP[1].TXFD, MDR_USB->SEP[2].TXFD, MDR_USB->SEP[3].TXFD

Таблица 24-72 – Регистр SEP[x].TXFD

Номер	31…8	7…0
Доступ	U	R/W

Сброс	0	0
	-	**TX FIFO DATA[7:0]**

Таблица 24-73 – Описание бит регистра SEP[x].TXFD

№ 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

24.9.37 MDR_USB->SEP[0].TXFDC, MDR_USB->SEP[1].TXFDC, MDR_USB->SEP[2].TXFDC, MDR_USB->SEP[3].TXFDC

Таблица 24-74 – Регистр SEP[x].TXFDC

Номер	31…1	0
Доступ	U	R/W
Сброс	0	0
	-	**FIFO FORCE EMPTY**

Таблица 24-75 – Описание бит регистра SEP[x].TXFDC

№ 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

25 Таймеры общего назначения MDR_TIMERx

Все блоки таймеров выполнены на основе 16-битного перезагружаемого счетчика, который синхронизируется с выхода 16-битного предделителя. Перезагружаемое значение хранится в отдельном регистре. Счет может быть прямой, обратный или двунаправленный (сначала прямой до определенного значения, а затем обратный).

Каждый из четырех таймеров микроконтроллера содержит 16-битный счетчик, 16-битный предделитель частоты и 4-канальный блок захвата/сравнения. Их можно синхронизировать системной синхронизацией, внешними сигналами или другими таймерами.

Помимо составляющего основу таймера счетчика, в каждый блок таймера также входит четырехканальный блок захвата/сравнения. Данный блок выполняет как стандартные функции захвата и сравнения, так и ряд специальных функций. Таймеры с 4 каналами схем захвата и ШИМ с функциями формирования «мертвой зоны» и аппаратной блокировки. Каждый из таймеров может генерировать прерывания и запросы DMA.

Особенности:

- 16-битный счетчик; счёт прямой, обратный или двунаправленный.
- 16-разрядный программируемый предварительный делитель частоты.
- до четырех независимых 16-битных каналов захвата на один таймер. Каждый из каналов захвата может захватить (скопировать) текущее значение таймера при изменении некоторого входного сигнала. В случае захвата имеется дополнительная возможность генерировать прерывание и/или запрос DMA.
- четыре 16-битных регистра сравнения (совпадения), которые позволяют осуществлять непрерывное сравнение, с дополнительной возможностью генерировать прерывание и/или запрос DMA при совпадении;
- имеется до четыре внешних выводов, соответствующих регистрам совпадения со следующими возможностями:
  - сброс в НИЗКИЙ уровень при совпадении;
  - установка в ВЫСОКИЙ уровень при совпадении;
  - переключение (инвертирование) при совпадении;
  - при совпадении состояние выхода не изменяется;
  - переключение при некотором условии.

25.1 Функционирование

Таймер предназначен для того, чтобы подсчитывать циклы периферийной тактовой частоты Fdts или какие-либо внешние события и произвольно генерировать прерывания, запросы DMA или выполнять другие действия. Значения таймера, при достижении которых будут выполнены те или иные действия, задаются восемью регистрами совпадения. Кроме того, в микроконтроллере имеются четыре входа захвата, чтобы захватить значение таймера при изменении некоторого входного сигнала, с возможностью генерировать прерывание или запрос DMA.

25.1.1 Структурная схема

Структурная схема блока Таймер представлена на Рисунок 25–1.

Таймер содержит основной 16-ти битный счетчик CNT, блок регистров управления и четыре канала схем захвата/ШИМ.

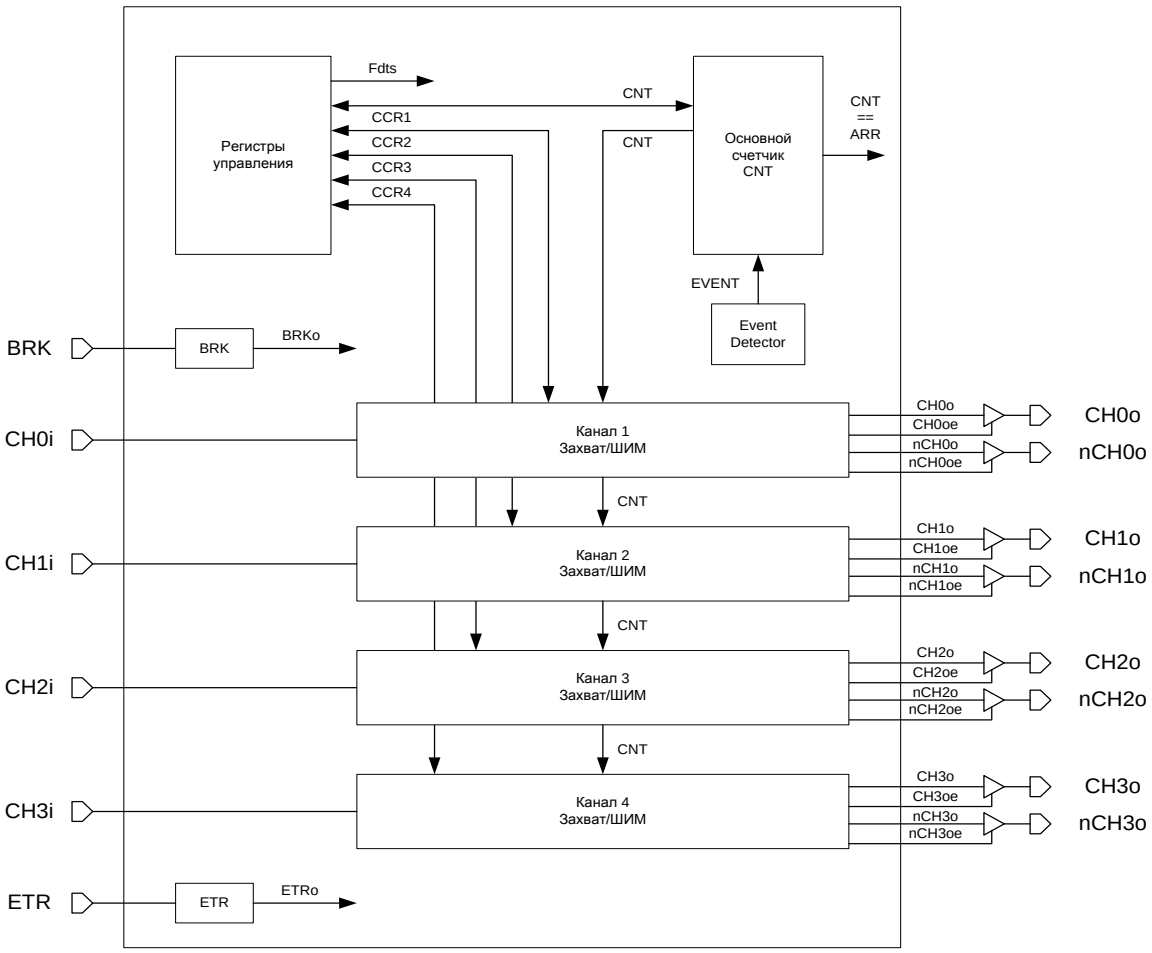


Рисунок 25–1 – Структурная схема таймера

Таймер позволяет работать в режимах:

- таймер;
- расширенный таймер, с объединением нескольких таймеров;
- режим захвата;
- ШИМ.

25.1.2 Инициализация таймера

Перед началом работы с таймерами в первую очередь должны быть включены тактовые сигналы.

Для задания тактовой частоты блока необходимо установить бит разрешения тактирования блока (бит 13 для таймера 1, бит 14 для таймера 2, бит 15 для таймера 3 регистра PER_CLOCK). В регистре TIM_CLOCK установить бит TIMxCLKEN, чтобы разрешить тактовую частоту для определенного таймера, задать коэффициент деления тактовой частоты HCLK для каждого таймера.

После подачи тактового сигнала на блок таймера можно приступать к работе с ним.

25.1.3 Режим таймера

Таймеры построены на базе 16-битного счетчика, объединенного с 16-битным предварительным делителем. Скорость счета таймера зависит от значения, находящегося в регистре делителя.

Счетчик может считать вверх, вниз или вверх и вниз (счёт прямой, обратный, двунаправленный).

Базовый блок таймера включает в себя:

- основной счетчик таймера (TIMx_CNT);
- делитель частоты при счете основного счетчика (TIMx_PSC);
- основание счета основного счетчика (TIMx_ARR).

Сигналом для изменения CNT может служить как внутренняя частота TIM_CLK, так и события в других счетчиках, либо события на линиях TxCHi данного счетчика.

Чтобы запустить работу основного счетчика, необходимо задать:

- Начальное значение основного счетчика таймера – TIMx_CNT;
- Значение предварительного делителя счетчика – TIMx_PSG, при этом основной счетчик будет считать на частоте CLK = TIMx_CLK/(PSG + 1);
- Значение основания счета для основного счетчика TIMx_ARR;
- Режим работы счетчика TIMx_CNTRL:
  - выбрать источник события переключения счетчика EVENT_SEL;
  - режим счета основного счетчика CNT_MODE (значения 00 и 01 при тактировании внутренней частотой, значения 10 и 11 при тактировании внешними сигналами);
  - направление счета основного счетчика DIR;
- разрешить работу счетчика CNT_EN.

По событиям совпадения значения основного счетчика с значением нуля или значением основания счета генерируется прерывание и запрос DMA, которые могут быть замаскированы.

25.2 Режимы счета

Счет вверх: CNT_MODE = 00, DIR = 0 (пример: счет вверх от 0 до 0x13, стартовое значение 0x04)

```
MDR_TIMERx->CNTRL  = 0x00000000;      //Режим инициализации таймера
//Настраиваем работу основного счетчика
MDR_TIMERx->CNT = 0x00000004; //Начальное значение счетчика
MDR_TIMERx->PSG = 0x00000000; //Предделитель частоты
MDR_TIMERx->ARR = 0x00000013; //Основание счета

//Разрешение работы таймера.
MDR_TIMERx->CNTRL = 0x00000001;      //Счет вверх по TIM_CLK.
```

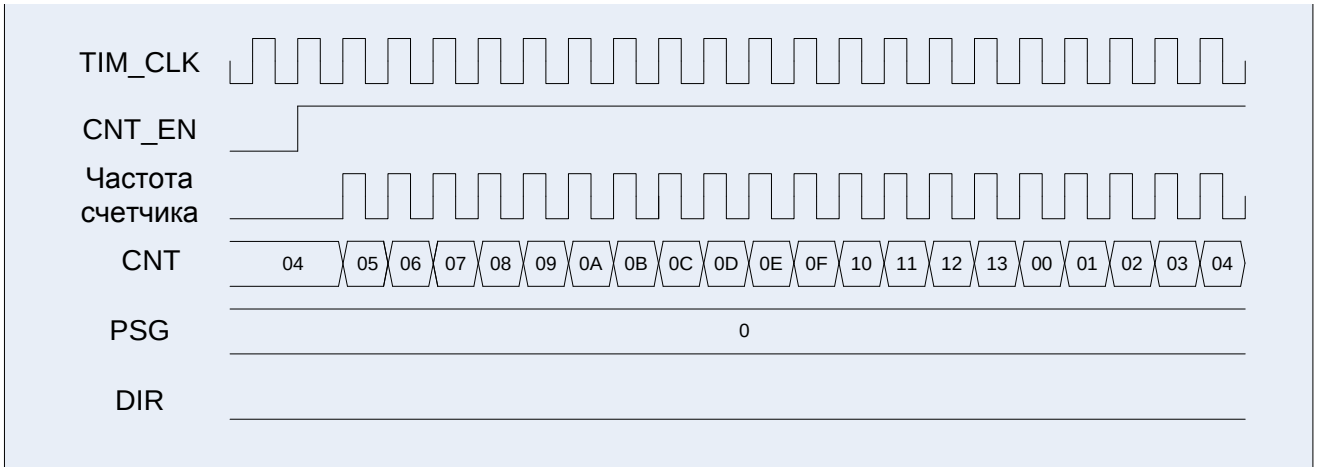


Рисунок 25–2 – Диаграммы работы таймера, счет вверх

Счет вниз: CNT_MODE = 00, DIR = 1 (пример: счет вниз от 0x13 до 0, стартовое значение 0x04)

```
MDR_TIMERx->CNTRL  = 0x00000000;     //Режим инициализации таймера
//Настраиваем работу основного счетчика
MDR_TIMERx->CNT = 0x00000004; //Начальное значение счетчика
MDR_TIMERx->PSG = 0x00000000; //Предделитель частоты
MDR_TIMERx->ARR = 0x00000013; //Основание счета

//Разрешение работы таймера.
MDR_TIMERx->CNTRL = 0x00000009;      //Счет вниз по TIM_CLK.
```

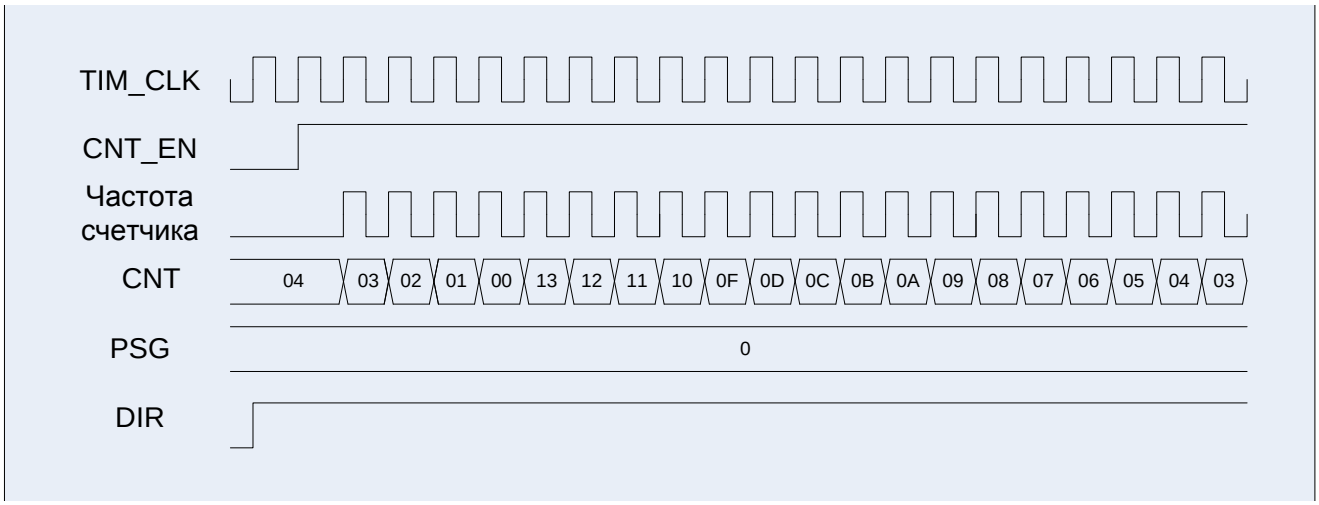


Рисунок 25–3 – Диаграммы работы таймера, счет вниз

Счет вверх/вниз: 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
MDR_TIMERx ->ARR = 0x00000013;       //Основание счета

//Разрешение работы таймера.
MDR_TIMERx ->CNTRL = 0x00000041;     //Счет вверх/вниз по TIM_CLK.
```

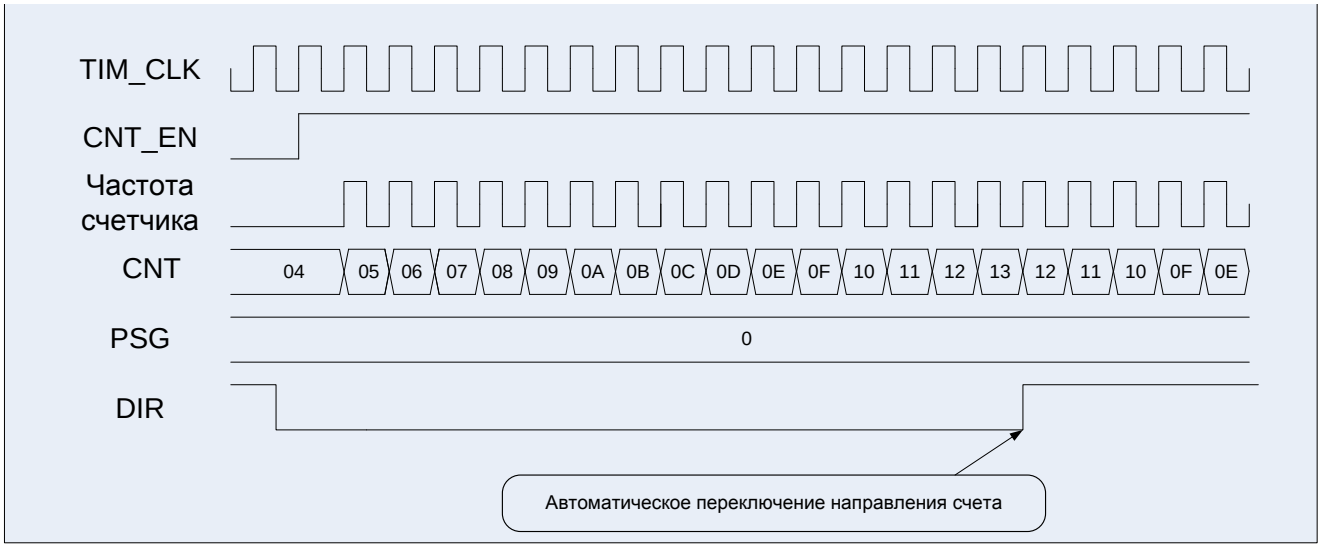


Рисунок 25–4 – Диаграммы работы таймера, счет вверх/вниз, сначала вверх

Счет вверх/вниз: 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
MDR_TIMERx->ARR = 0x00000013; //Основание счета

//Разрешение работы таймера.
MDR_TIMERx->CNTRL = 0x00000049;      //Счет вверх/вниз по TIM_CLK.
```

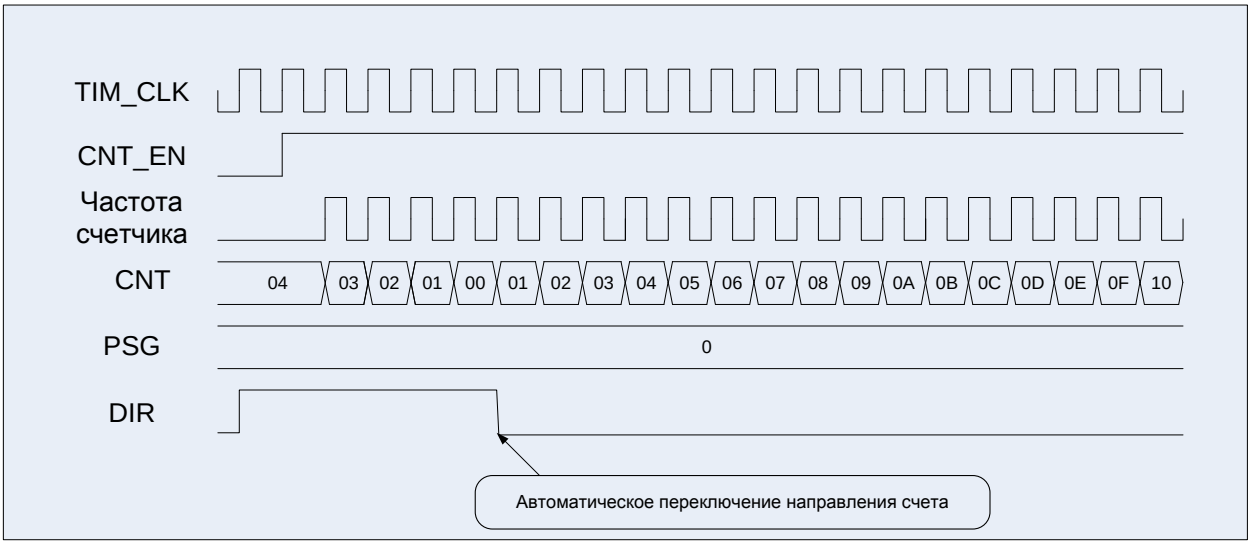


Рисунок 25–5 – Диаграммы работы таймера, счет вверх/вниз, сначала вниз

25.3 Источник событий для счета

Источники событий для счета:

- внутренний тактовый сигнал (TIM_CLK);
- события в других счетчиках (CNT==ARR в таймере X);
- внешний тактовый сигнал режим 1: События на линиях TxCHO данного счетчика;
- внешний тактовый сигнал режим 2: События на входе ETR данного счетчика.

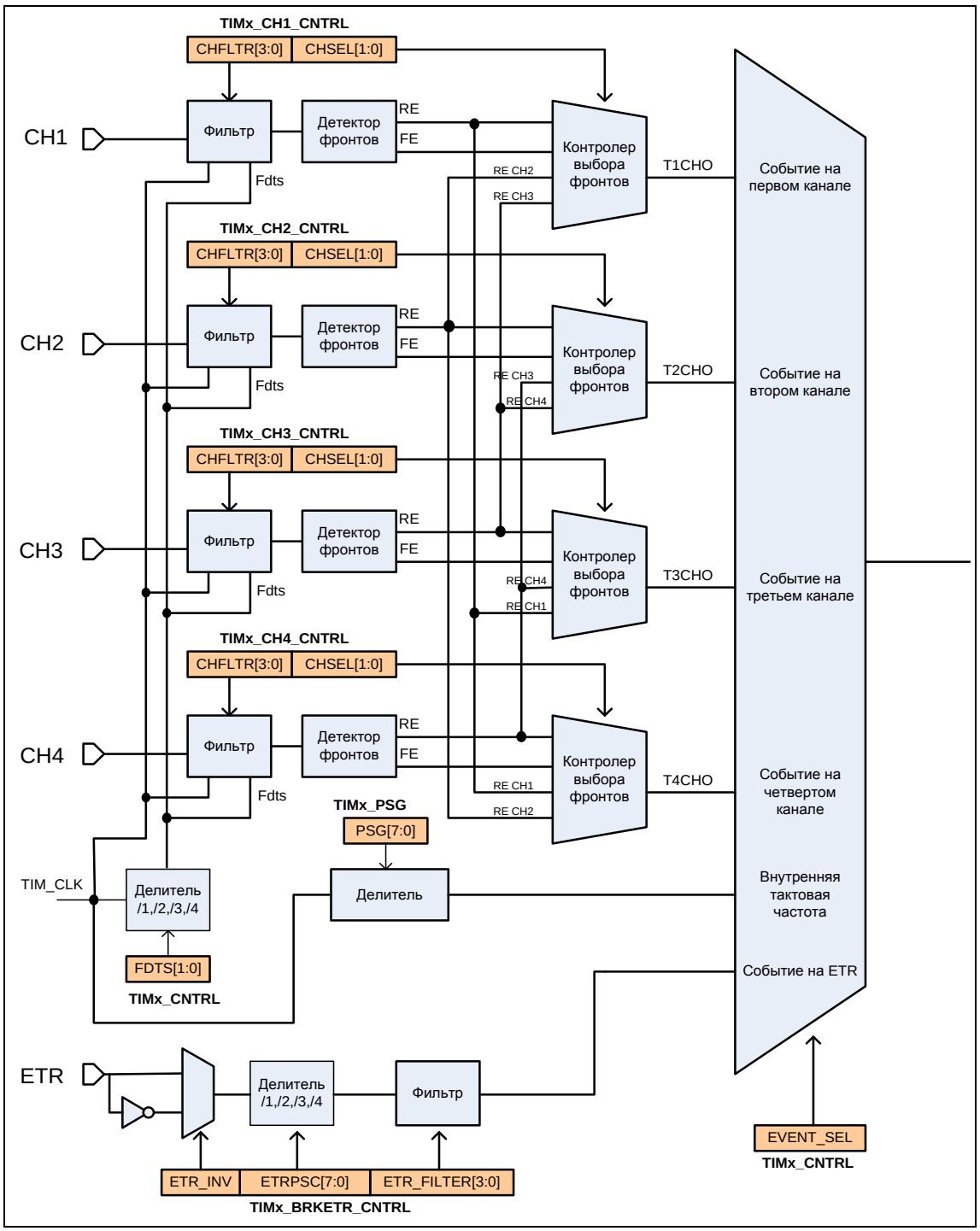


Рисунок 25–6 – Структурная схема формирования события для счета

25.3.1 Внутренний тактовый сигнал (TIM_CLK)

Этот режим выбирается, когда CNT_MODE = 0x, EVENT_SEL = 0000. Для запуска этого режима необходимо задать начальное значение основного счетчика, значение предварительного делителя основного счетчика, основание счета для основного счетчика и задать режим работы в регистре CNTRL. Значения регистров CNT, PSG и ARR можно изменять даже во время работы счетчика, при этом их значения вступят в силу по CNT = ARR или CNT = 0, в зависимости от направления счета. Значение регистра основания счета (ARR) может вступить в силу мгновенно после записи его в регистр при условии установленного поля ARRB_EN = 1 (регистр CNTRL). Если значение предварительного делителя основного счетчика не равно 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 изменится на 1 и снова начинается счет. Поле DIR определяет, в какую сторону будет меняться значение счетчика: DIR = 0 – счетчик считает вверх (см. Рисунок 25–7), DIR = 1 – счетчик считает вниз (см. Рисунок 25–8).

CNT_MODE = 00, EVENT_SEL = 0000, DIR = 0

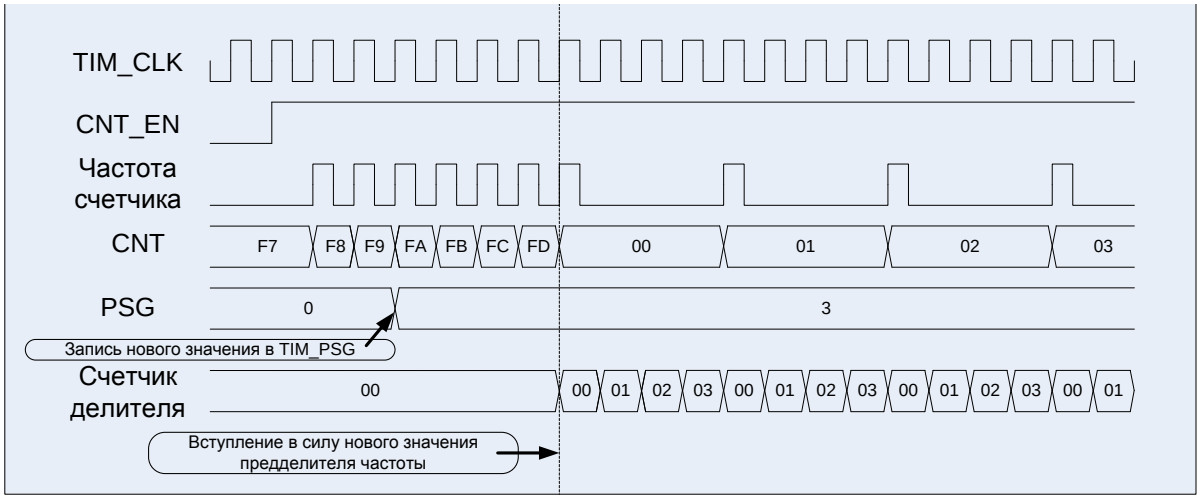


Рисунок 25–7 – Диаграммы работы счетчика: счет вверх

CNT_MODE = 00, EVENT_SEL = 0000, DIR = 1

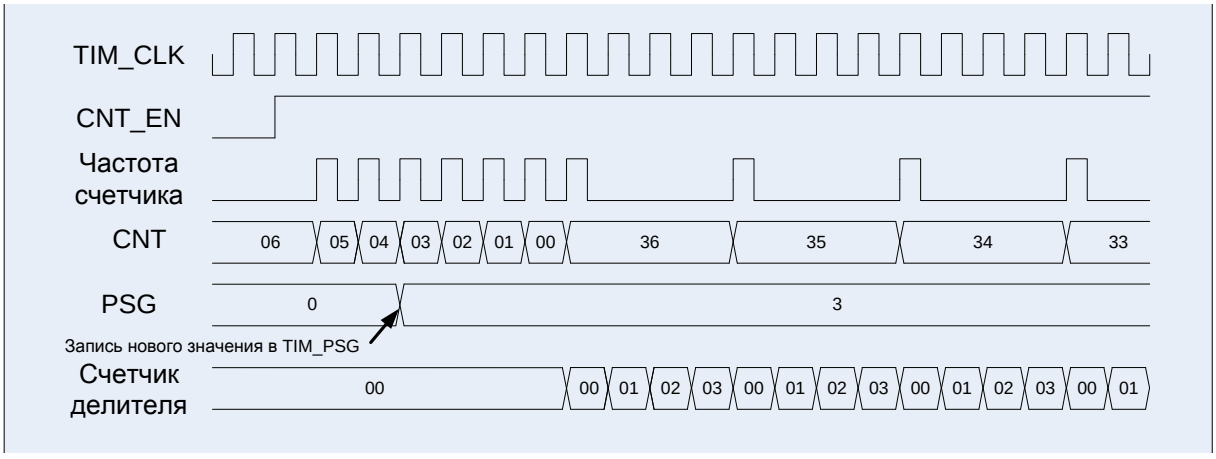


Рисунок 25–8 – Диаграммы работы счетчика: счет вниз

Если CNT_MODE = 00, то направление счета определяется полем DIR, если CNT_MODE = 01, счетчик считает вверх/вниз с автоматическим изменением DIR (см. Рисунок 25–9).

CNT_MODE = 01, EVENT_SEL = 0000, DIR = 1

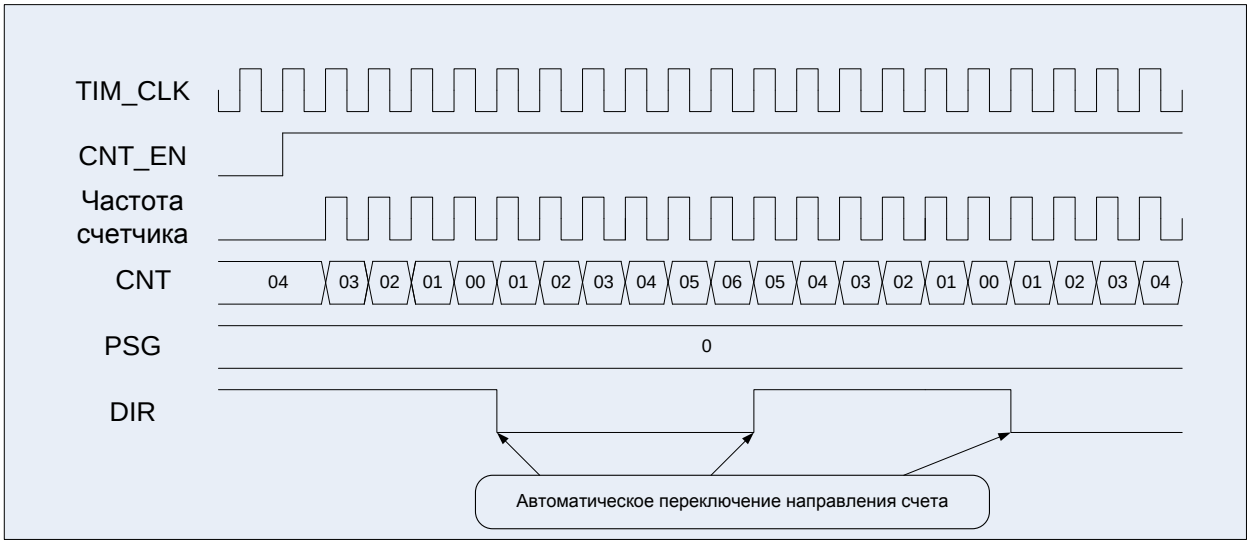


Рисунок 25–9 – Диаграммы работы счетчика: счет вниз/вверх

25.3.2 События в других счетчиках (CNT==ARR в таймере X)

Каждый из блоков таймеров полностью независим друг от друга, но у них предусмотрена возможность синхронизированной друг с другом 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 имеются входы запуска от других трех таймеров, а также внешние входы, связанные с выводами блоков захвата/сравнения.

У каждого из блоков таймеров имеется выход запуска, который соединен с входами других трех таймеров. Синхронизация таймеров возможна в нескольких различных режимах. Ниже показан пример каскадного соединения счетчиков.

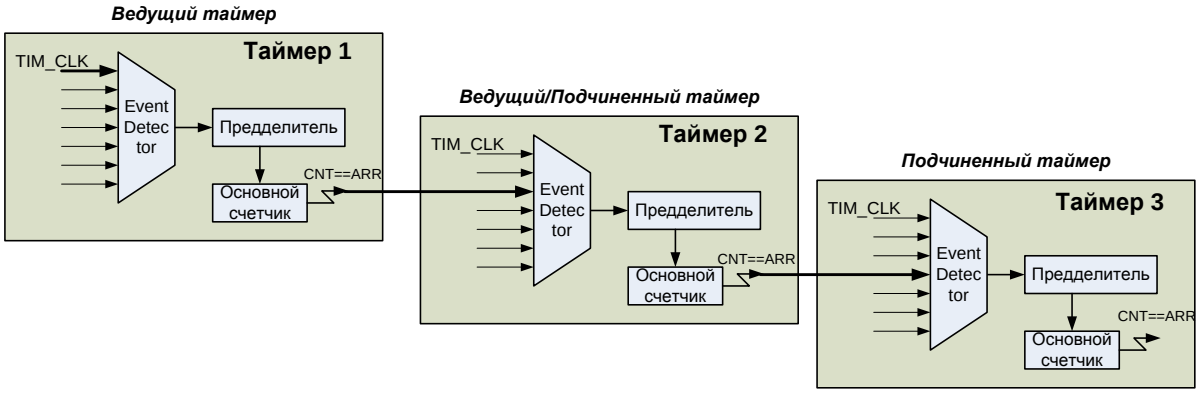


Рисунок 25–10 – Пример каскадного соединения таймеров

DIR_1, DIR_2, DIR_3 = 0;
EVENT_SEL_1 = 0000, EVENT_SEL_2 = 0001, EVENT_SEL_3 = 0010;
CNT_MODE_1, CNT_MODE_2, CNT_MODE_3 = 00;

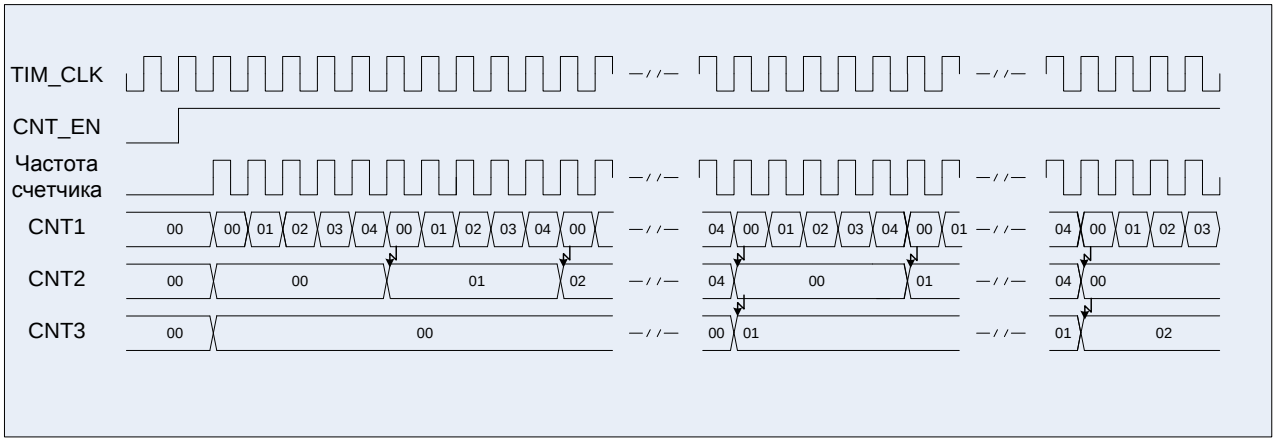


Рисунок 25–11 – Диаграммы работы трех таймеров в каскаде

25.3.3 Внешний тактовый сигнал «Режим 1». События на линиях TxCHO данного счетчика

Этот режим выбирается, когда EVENT_SEL = 01xx в регистре CNTRL. Счетчик может считать по положительному фронту, или по отрицательному фронту на выбранном входе, или по положительному фронту на других каналах (см. Рисунок 25–10). На входе сигнала стоит фильтр, с помощью которого можно контролировать длительность сигнала. Для фильтрования можно использовать как сигнал TIM_CLK, при этом может быть идентифицированная длительность 1, 2, 4, 8 TIM_CLK, так и производную от TIM_CLK частоту FDTS. Частота семплирования данных задается в регистре CNTRL в поле FDTS.

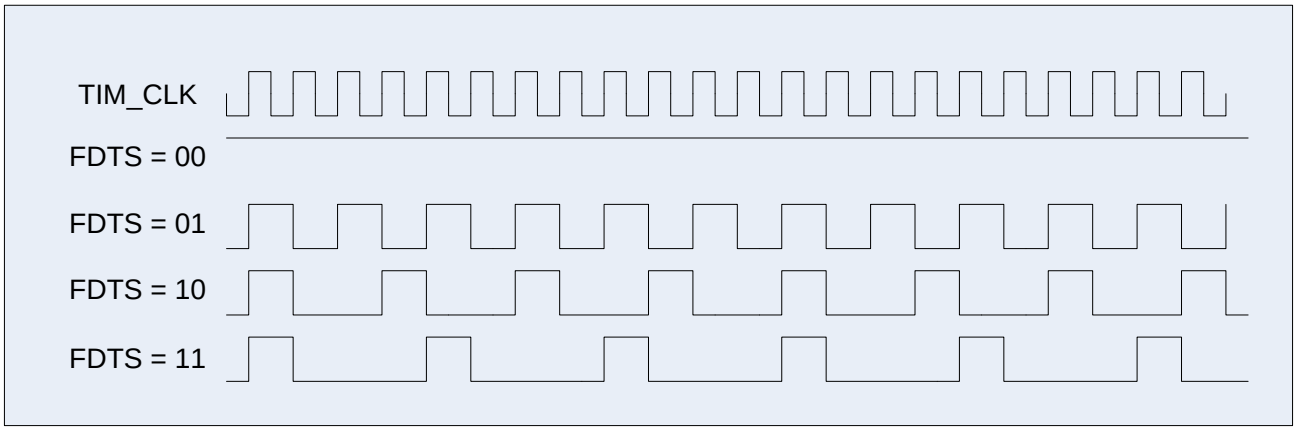


Рисунок 25–12 – Диаграммы возможных частот семплирования данных (FDTS)

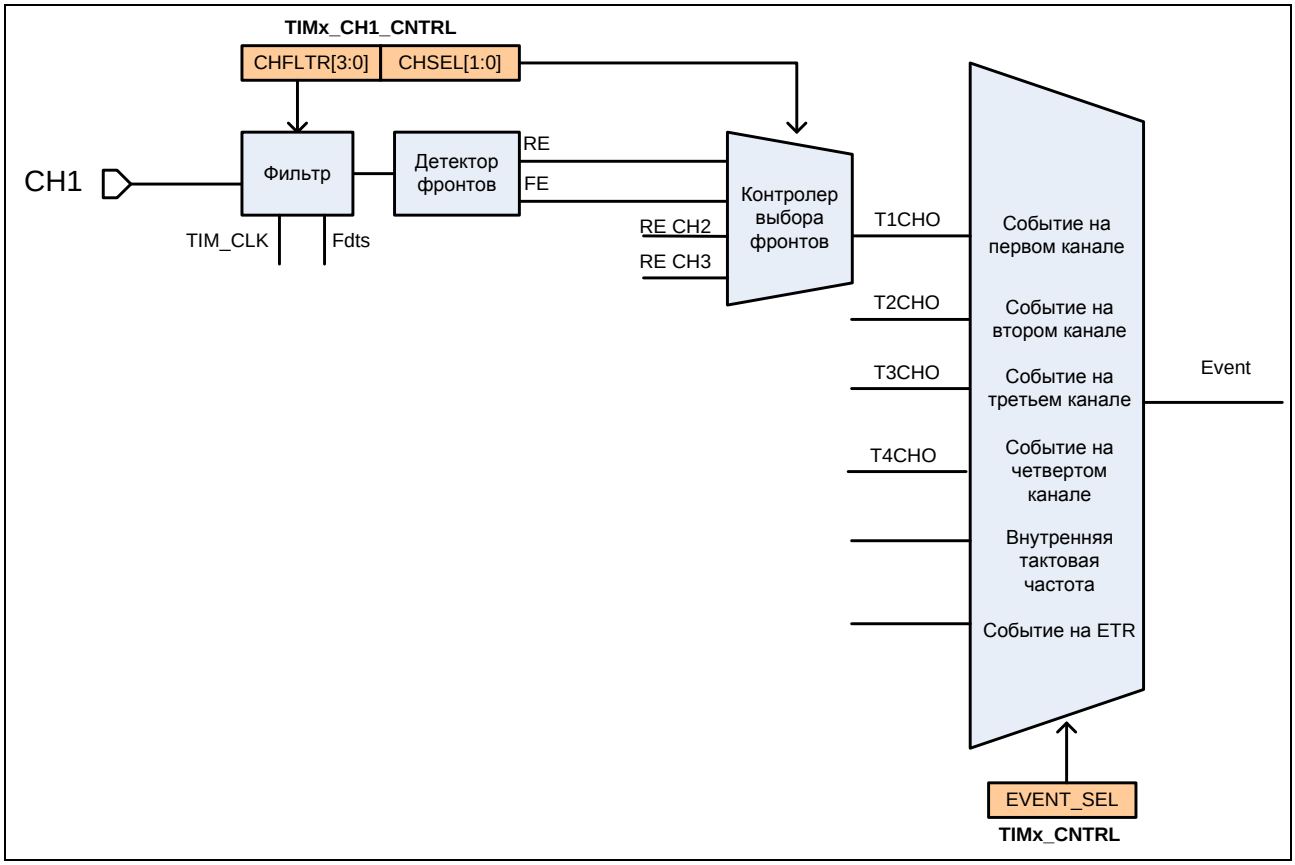


Рисунок 25–13 – Тактирование с входа первого канала

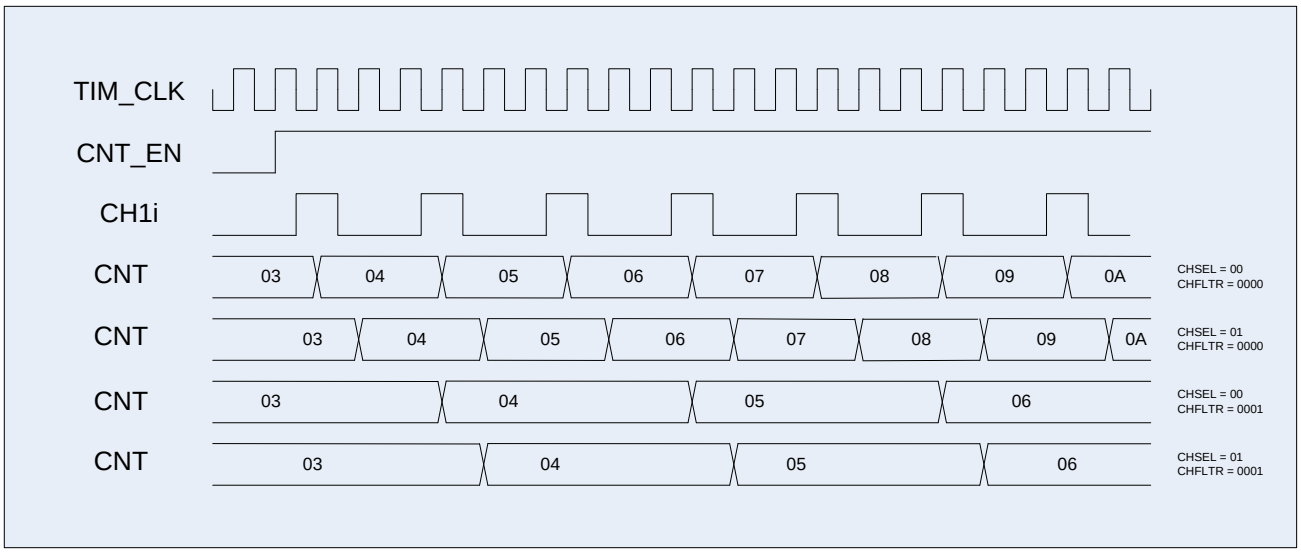


Рисунок 25–14 – Диаграмма внешнего тактирования с разными вариантами фильтра

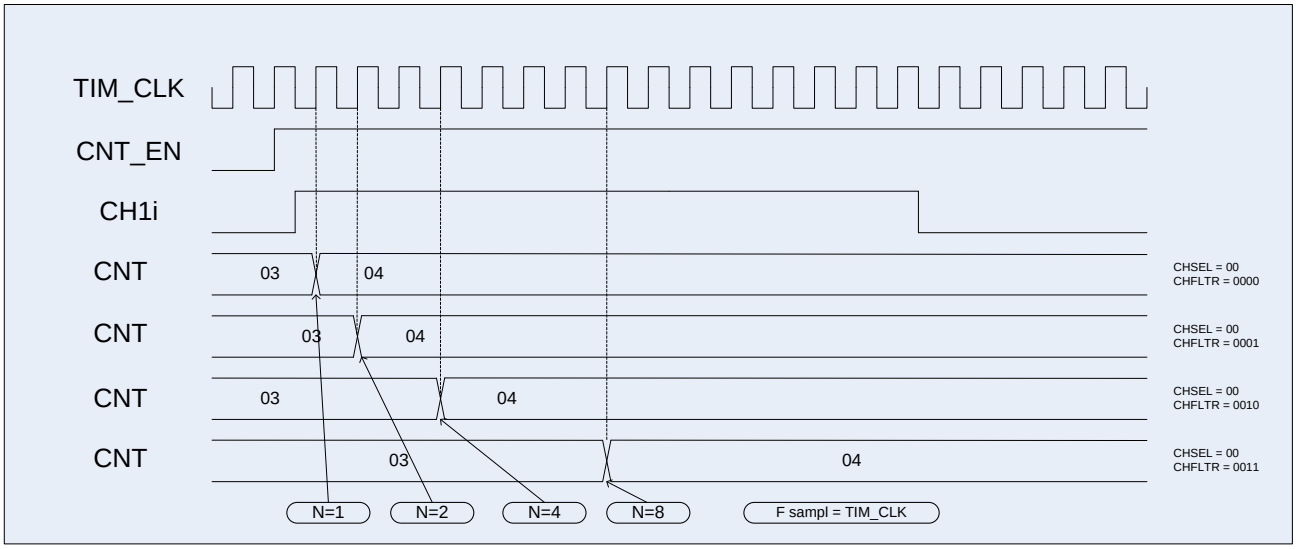


Рисунок 25–15 – Диаграмма внешнего тактирования с разными вариантами фильтра

25.3.4 Внешний тактовый сигнал «Режим 2». События на входе ETR данного счетчика

Этот режим выбирается, когда EVENT_SEL = 1000 в регистре CNTRL. В регистре BRKETR_CNTRL можно настроить коэффициент деления 2, 4 или 8 (ETRPSC) данного входа тактовой частоты, а также использовать инверсию входа.

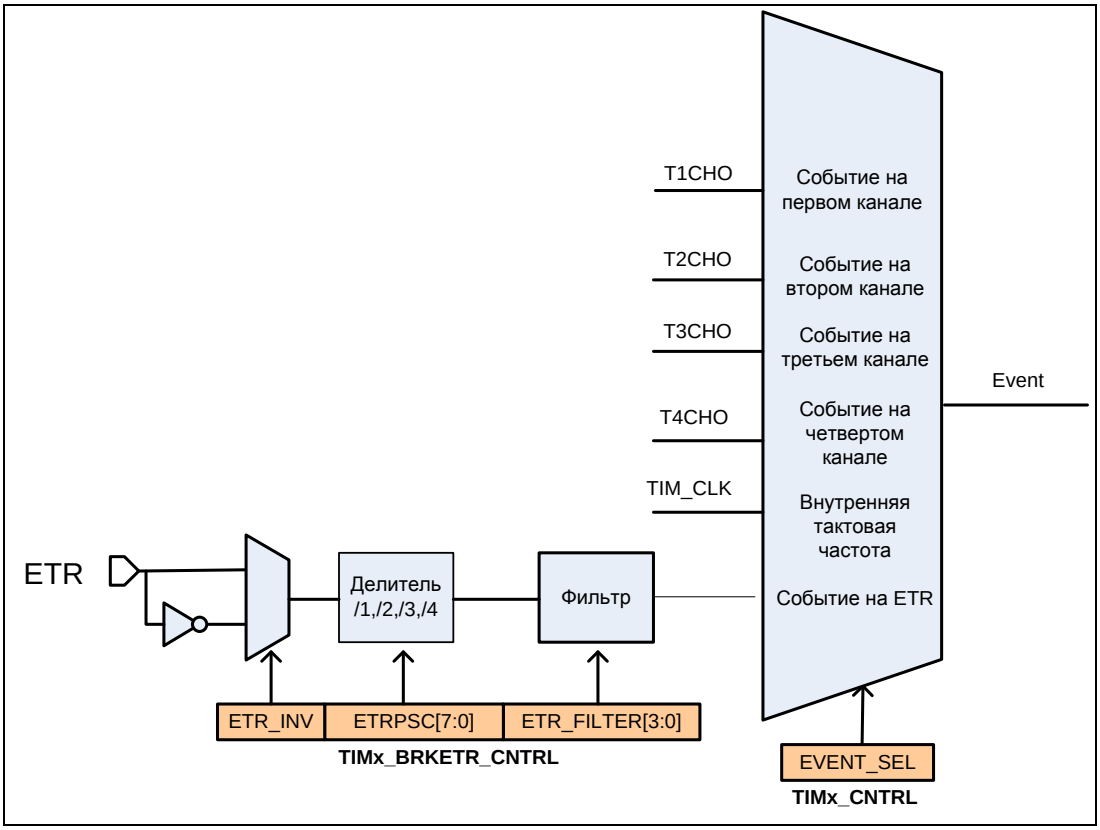


Рисунок 25–16 – Схема тактирования сигналом со входа ETR

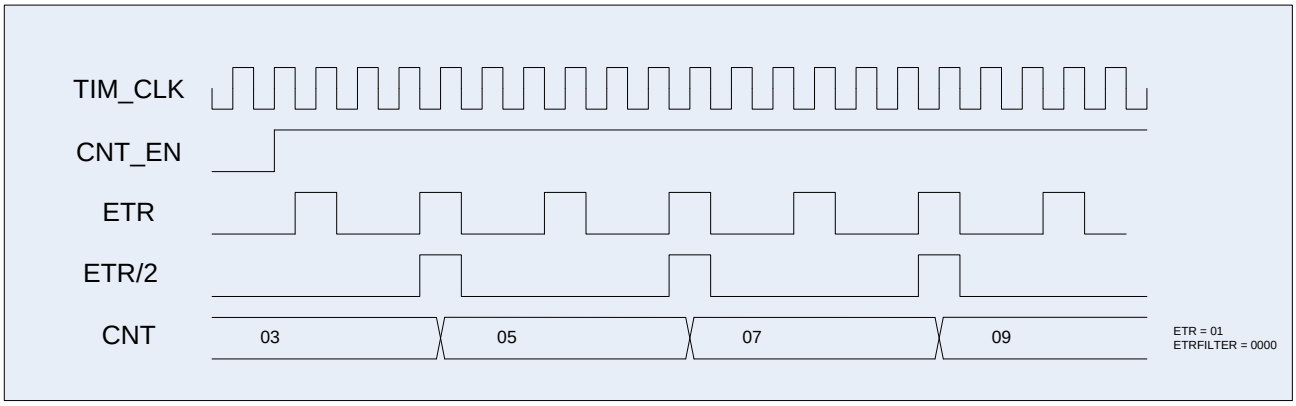


Рисунок 25–17 – Диаграмма тактирования сигналом со входа ETR

25.4 Режим захвата

Структурная схема блока захвата представлена на Рисунок 25–18.

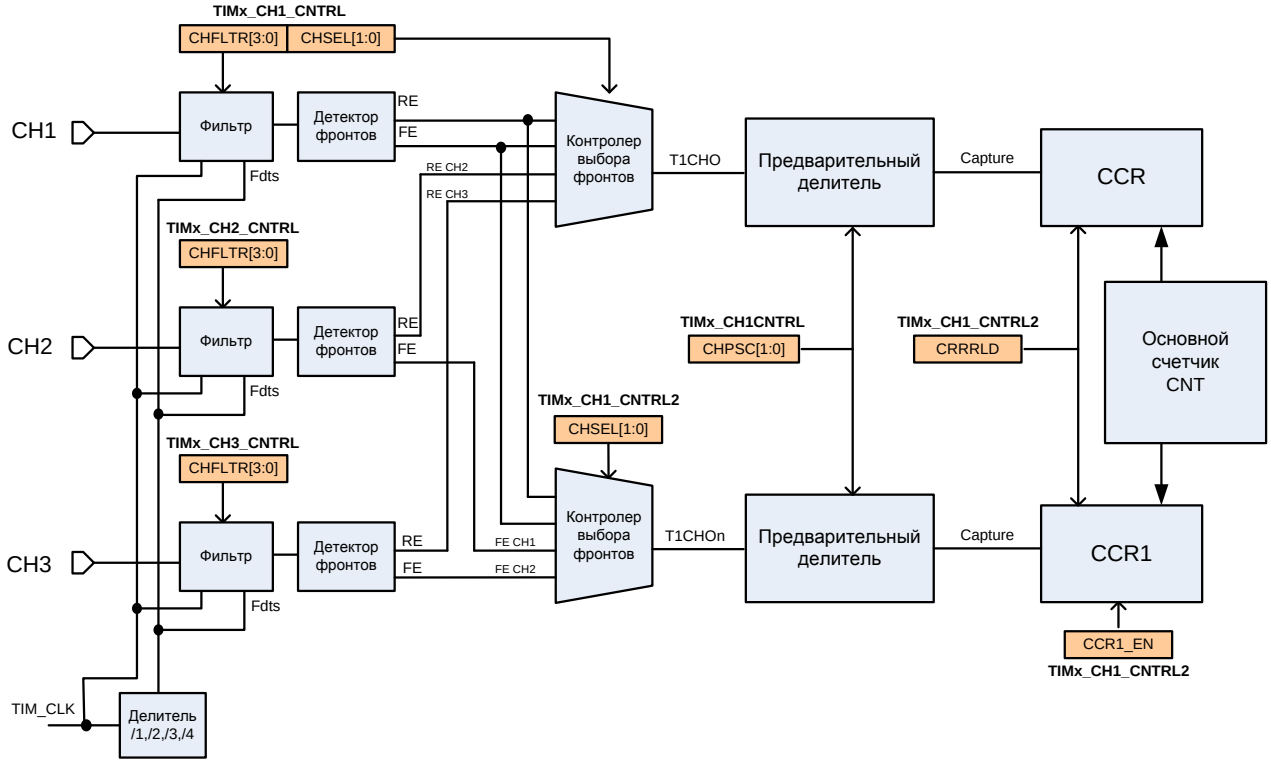


Рисунок 25–18 – Структурная схема блока захвата на примере канала 1

Для включения режима захвата для определенного канала необходимо в регистре управления каналом CHy_CNTRL записать “1” в поле CAPnPWM. Для регистрации событий по линии CHxi используется схема регистрации событий. Входной сигнал фиксируется в Таймере с частотой Fdts, или TIM_CLK. Также вход может быть настроен на прием импульсов заданной длины за счет конфигурирования блока FILTER. На выходе блока фильтр вырабатывает сигнал положительного перепада и отрицательного перепада. На блоке MUX производится выбор используемого для захвата сигнала между положительным фронтом канала, отрицательным фронтом канала и положительными и отрицательными фронтами сигналов от других каналов. После блока MUX предварительный делитель может

быть использован для фиксации каждого события, каждого второго, каждого четвертого и каждого восьмого события. Выход предварительного делителя является сигналом Capture для регистра CCR, и Capture1 для регистра CCR1, при этом в регистры CCR и CCR1 записывается текущее значение основного счетчика CNT.

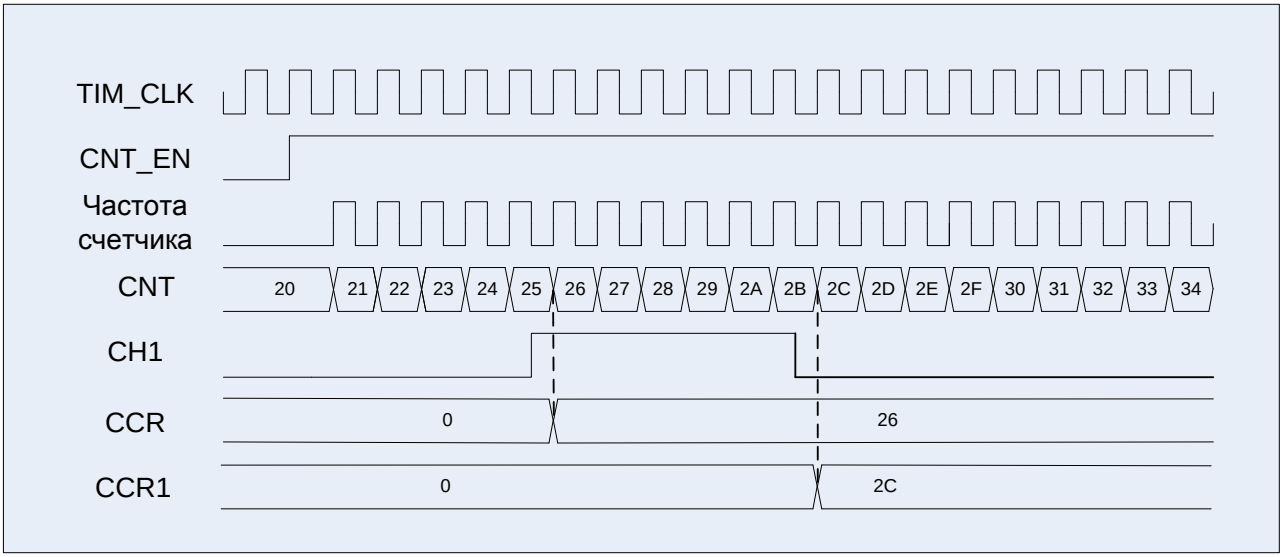


Рисунок 25–19 – Диаграмма захвата события со входа первого канала

На рисунке показан пример захвата значения основного счетчика в регистр CCR по положительному фронту на входе канала, а в регистр CCR1 – по отрицательному фронту на входе канала. В регистре IE можно разрешить выработку прерываний по событию захвата на определенном канале, а в регистре DMA_RE можно разрешить формирование запросов DMA.

25.5 Режим ШИМ

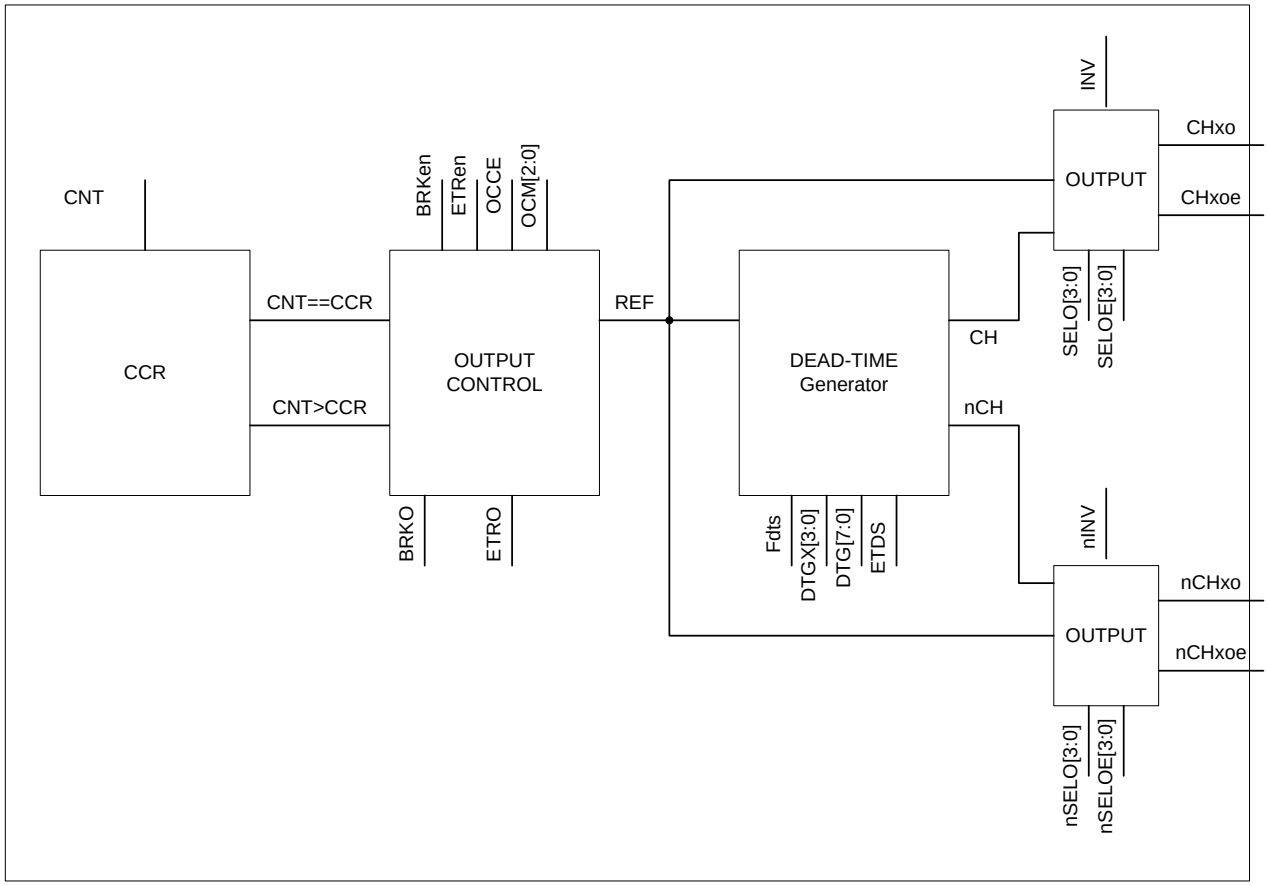


Рисунок 25–20 – Структурная схема блока формирования ШИМ

Для включения режима сравнения для определенного канала необходимо в регистре управления каналом CHy_CNTRL записать “0” в поле CAPnPWM. При работе в режиме ШИМ выходной сигнал может формироваться на основании сравнения значения в регистре CCR и основного счетчика CNT или регистров CCR, CCR1 и значения основного счетчика CNT. Полученный сигнал может без изменения выдаваться на выводы CHxO и nCHxO. Либо с применением схемы DEAD TIME Generator формируются управляющие сигналы с мертвой зоной. У каждого канала есть два выхода - прямой и инверсный. Для каждого выхода формируется как сигнал для выдачи, так и сигнал разрешения выдачи, т.е. если выход канала должен всегда выдавать тот или иной уровень, то на выводе разрешения выдачи CHxOE (для прямого) и на CHxNOE (для инверсного) должны формироваться “1”. Если канал работает на вход (например, режим захвата), то там всегда должен быть "0" для прямого канала. Сигналы OE формируются по тем же принципам, что и просто выходные уровни, но у них есть собственные сигналы разрешения вывода SELOE и nSELOE, в которых можно выбрать постоянный уровень, либо формируемый на основании REF.

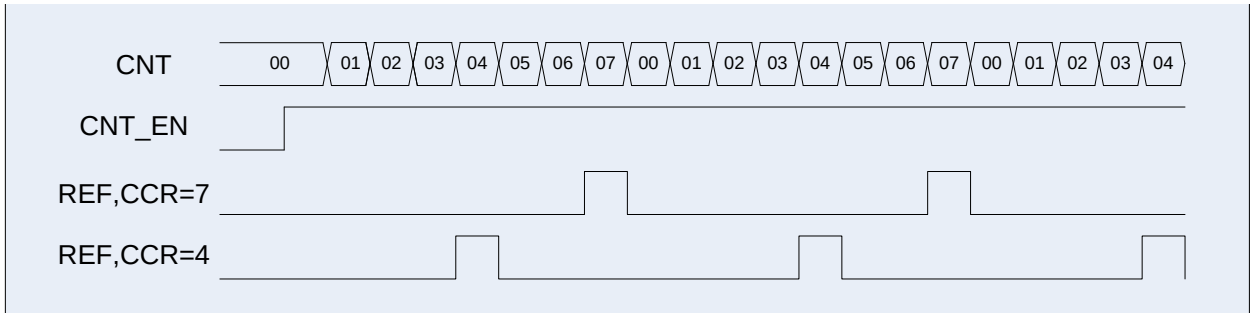


Рисунок 25–21 – Диаграмма работы схемы в режиме ШИМ, CCR1_EN=0

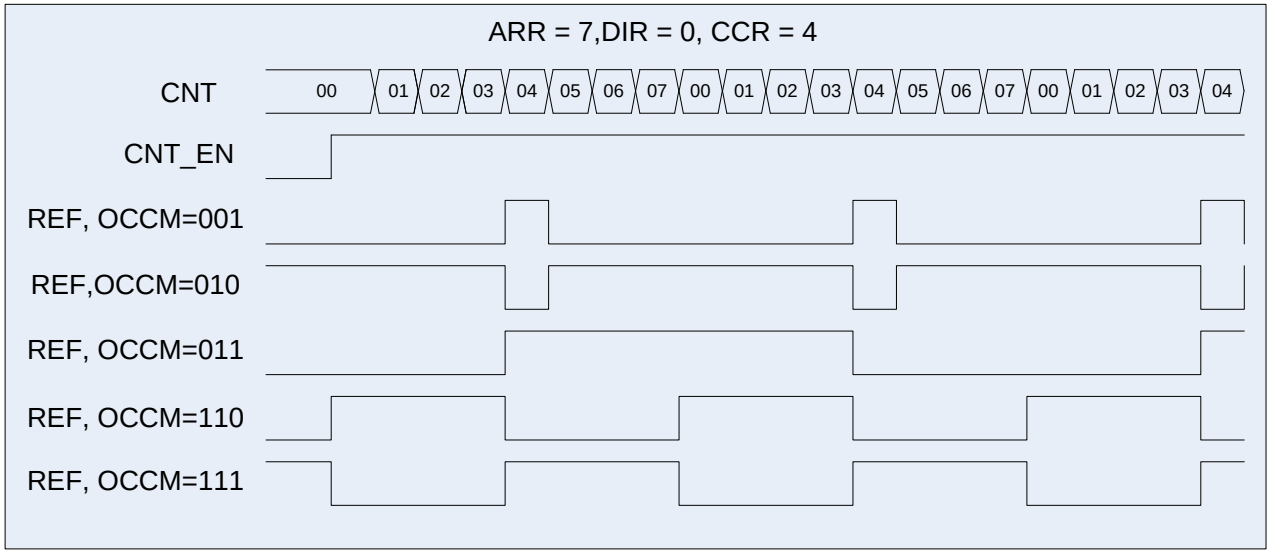


Рисунок 25–22 – Диаграмма работы схемы в режиме ШИМ, CCR1_EN=0

Сигнал REF может быть очищен с использованием внешнего сигнала с входа ETR, или внешнего, синхронизированного по PCLK сигналу со входа BRK.

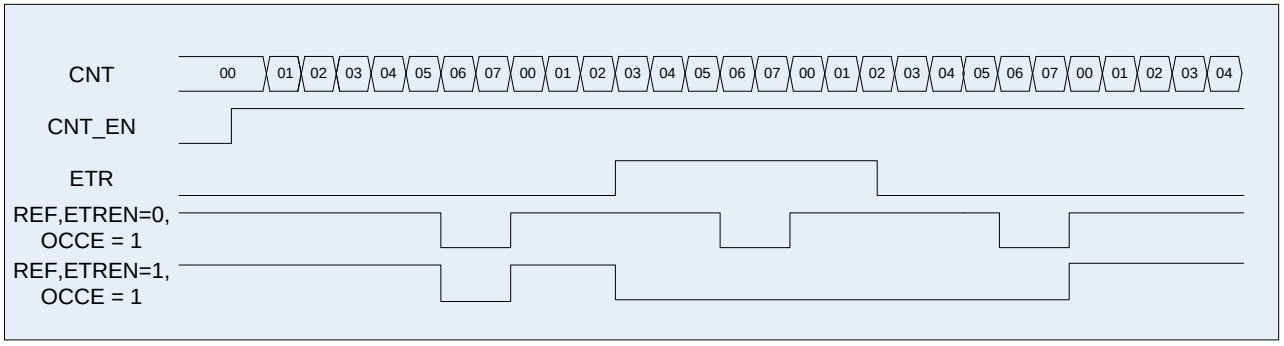


Рисунок 25–23 – Диаграмма работы схемы в режиме ШИМ, CCR1_EN = 0

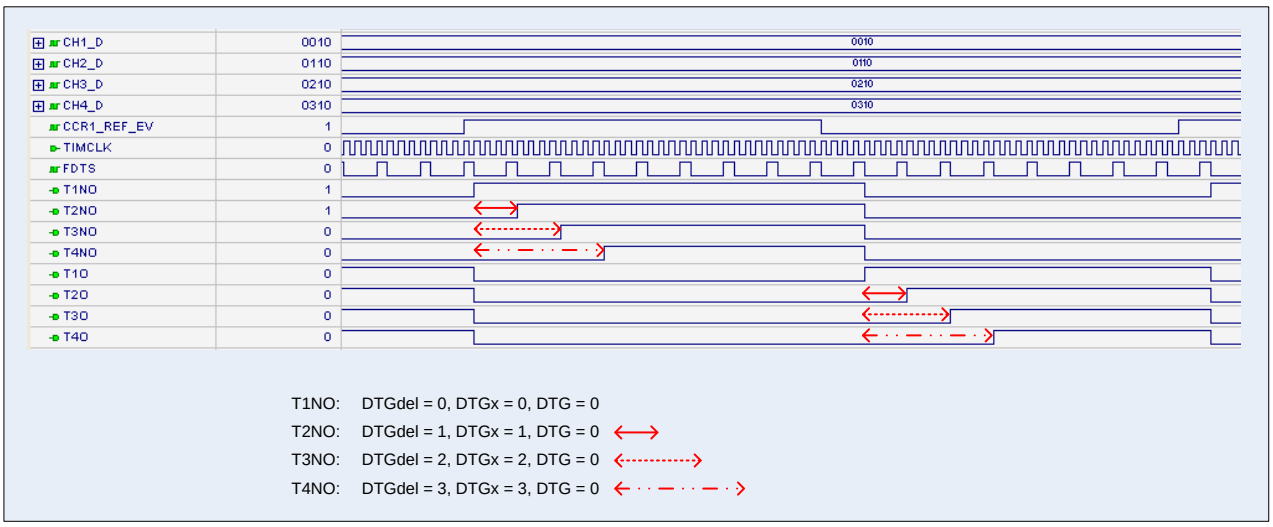


Рисунок 25–24 – Диаграмма работы схемы DTG

Если CCR1_EN = 1, тогда значение основного счетчика CNT сравнивается со значениями регистров CCR и CCR1, и в зависимости от запрограммированного формата выработки сигнала REF (регистры управления каналами таймера CHy_CNTRL поле OCCM) будет формироваться сигнал соответствующей формы.

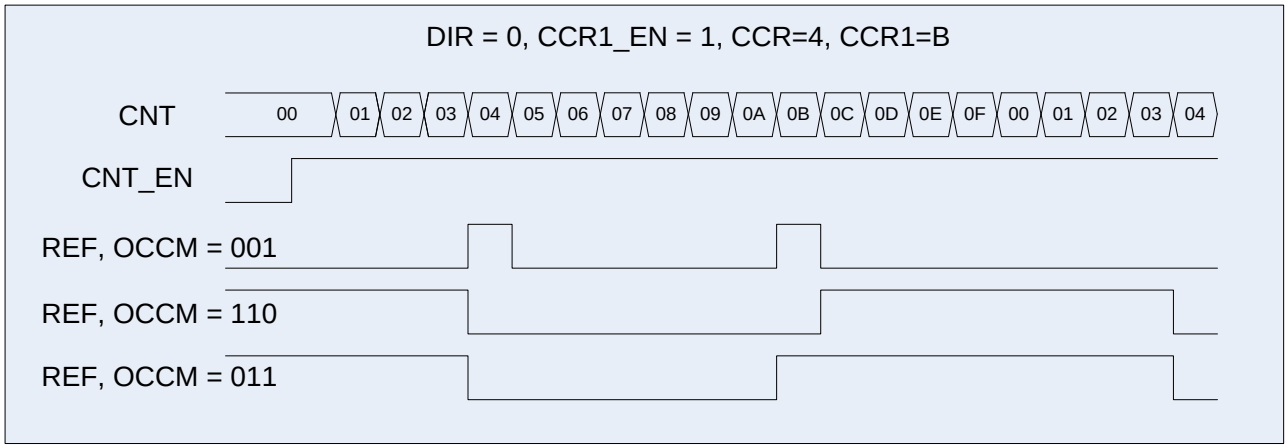


Рисунок 25–25 – Диаграмма работы схемы в режиме ШИМ, CCR1_EN = 1

При записи новых значений CCR и CCR1, если установлен бит CRRRLD, то регистры CCR1 и CCR получат новые значения только при CNT = 0, иначе запись осуществляется немедленно. Факт окончания записи обозначается взведением флага WR_CMPL.

25.6 Примеры

25.6.1 Обычный счетчик

```
MDR_RST_CLK->PER_CLOCK = 0xFFFFFFFF;
MDR_RST_CLK->TIM_CLOCK = 0x07000000;
MDR_TIMERx->CNTRL  = 0x00000000;
//Настраиваем работу основного счетчика
MDR_TIMERx->CNT = 0x00000000; //Начальное значение счетчика
MDR_TIMERx->PSG = 0x00000000; //Предделитель частоты
MDR_TIMERx->ARR = 0x0000000F; //Основание счета

MDR_TIMERx->IE = 0x00000002;    //Разрешение генерировать прерывание при CNT=ARR

MDR_TIMERx->CNTRL = 0x00000001;     //Счет вверх по TIM_CLK. Разрешение работы таймера.
```

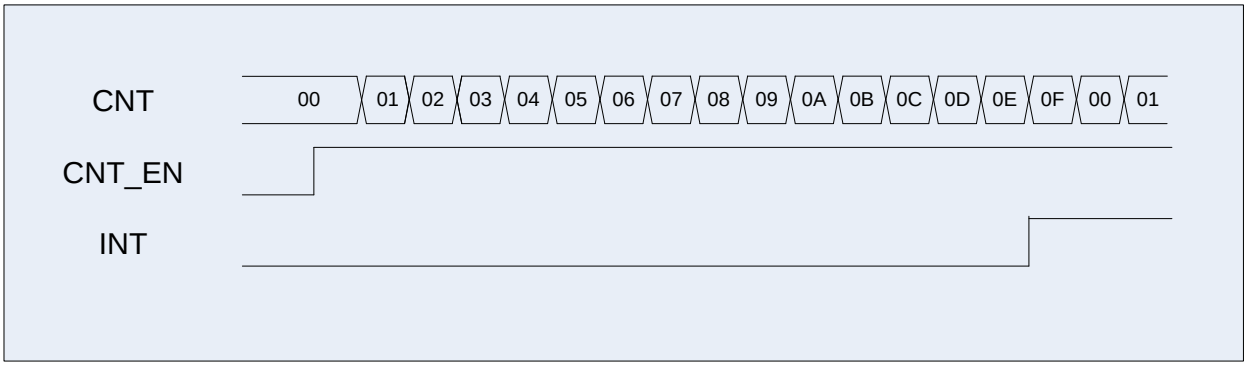


Рисунок 25–26 – Режим обычного счетчика

25.6.2 Режим захвата

```
MDR_RST_CLK->PER_CLOCK = 0xFFFFFFFF;   //Разрешение тактовой частоты таймеров
MDR_RST_CLK->TIM_CLOCK = 0x07000000;     //Включение тактовой частоты таймеров
TIMx->TIMx_CNTRL  = 0x00000000;//Режим инициализации таймера
//Настраиваем работу основного счетчика
MDR_TIMERx->CNT = 0x00000000; //Начальное значение счетчика
MDR_TIMERx->PSG = 0x00000000; //Предделитель частоты
MDR_TIMERx->ARR = 0x000000FF;//Основание счета

MDR_TIMERx->IE = 0x00001E00;    //Разрешение генерировать прерывание
                  //по переднему фронту на выходе CAP по всем каналам
//Режим работы каналов - захват
MDR_TIMERx->CHy_CNTRL[0] = 0x00008000;
MDR_TIMERx->CHy_CNTRL[1] = 0x00008002;
MDR_TIMERx->CHy_CNTRL[2] = 0x00008001;
MDR_TIMERx->CHy_CNTRL[3] = 0x00008003;

//Режим работы выхода канала – канал на выход не работает
MDR_TIMERx->CHy_CNTRL1[0]= 0x00000000;
MDR_TIMERx->CHy_CNTRL1[1]= 0x00000000;
MDR_TIMERx->CHy_CNTRL1[2]= 0x00000000;
MDR_TIMERx->CHy_CNTRL1[3]= 0x00000000;

MDR_TIMERx->CNTRL = 0x00000001;      //Счет вверх по TIM_CLK. Разрешение работы таймера
```

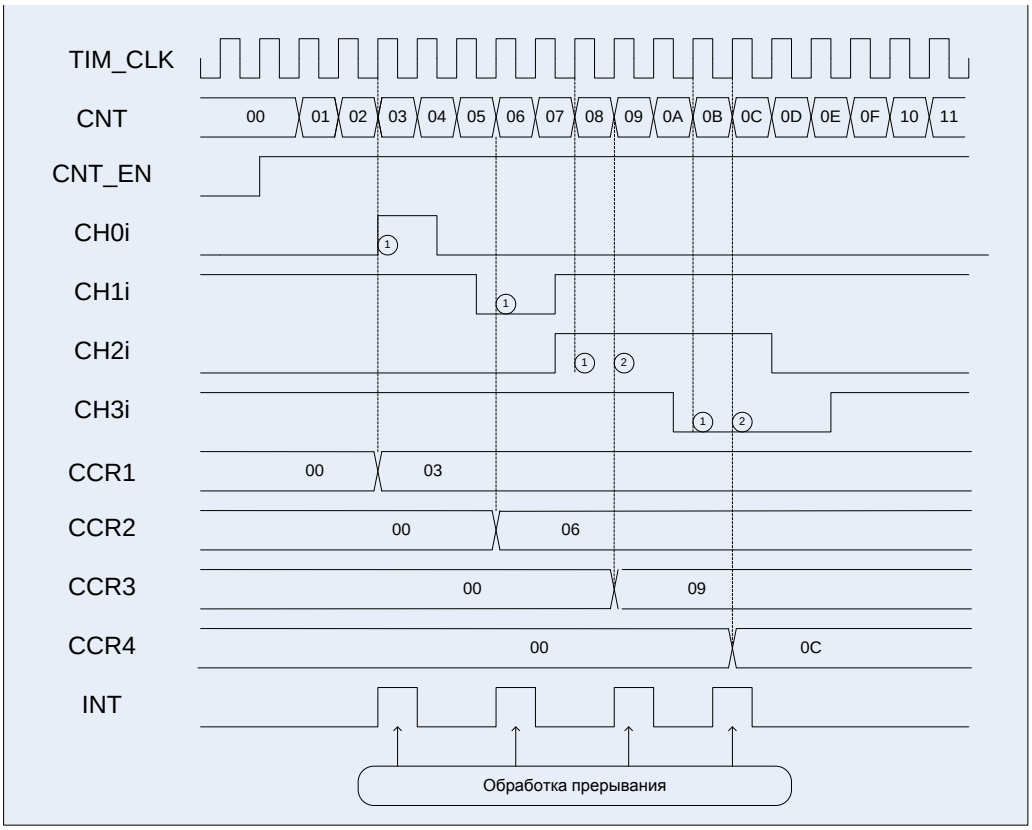


Рисунок 25–27 – Диаграммы примера работы в режиме захвата

25.6.3 Режим ШИМ

```
MDR_RST_CLK->PER_CLOCK = 0xFFFFFFFF;   //Разрешение тактовой частоты таймеров
MDR_RST_CLK->TIM_CLOCK = 0x07000000;     //Включение тактовой частоты таймеров
MDR_TIMERx->CNTRL  = 0x00000000;     //Режим инициализации таймера
//Настраиваем работу основного счетчика
MDR_TIMERx->CNT = 0x00000000; //Начальное значение счетчика
MDR_TIMERx->PSG = 0x00000000; //Предделитель частоты
MDR_TIMERx->ARR = 0x00000010; //Основание счета

MDR_TIMERx->IE = 0x000001E0;    //Разрешение генерировать прерывание
                //по переднему фронту на выходе REF по всем каналам
//Режим работы каналов - ШИМ
MDR_TIMERx->CHy_CNTRL[0] = 0x00000200;
MDR_TIMERx->CHy_CNTRL[1] = 0x00000200;
MDR_TIMERx->CHy_CNTRL[2] = 0x00000400;
MDR_TIMERx->CHy_CNTRL[3] = 0x00000600;

//Режим работы выхода канала – канал на выход не работает
MDR_TIMERx->CHy_CNTRL1[0]= 0x00000099;
MDR_TIMERx->CHy_CNTRL1[1]= 0x00000099;
MDR_TIMERx->CHy_CNTRL1[2]= 0x00000099;
MDR_TIMERx->CHy_CNTRL1[3]= 0x00000099;

//Разрешение работы таймера.
MDR_TIMERx->CNTRL = 0x00000001;      //Счет вверх по TIM_CLK.
```

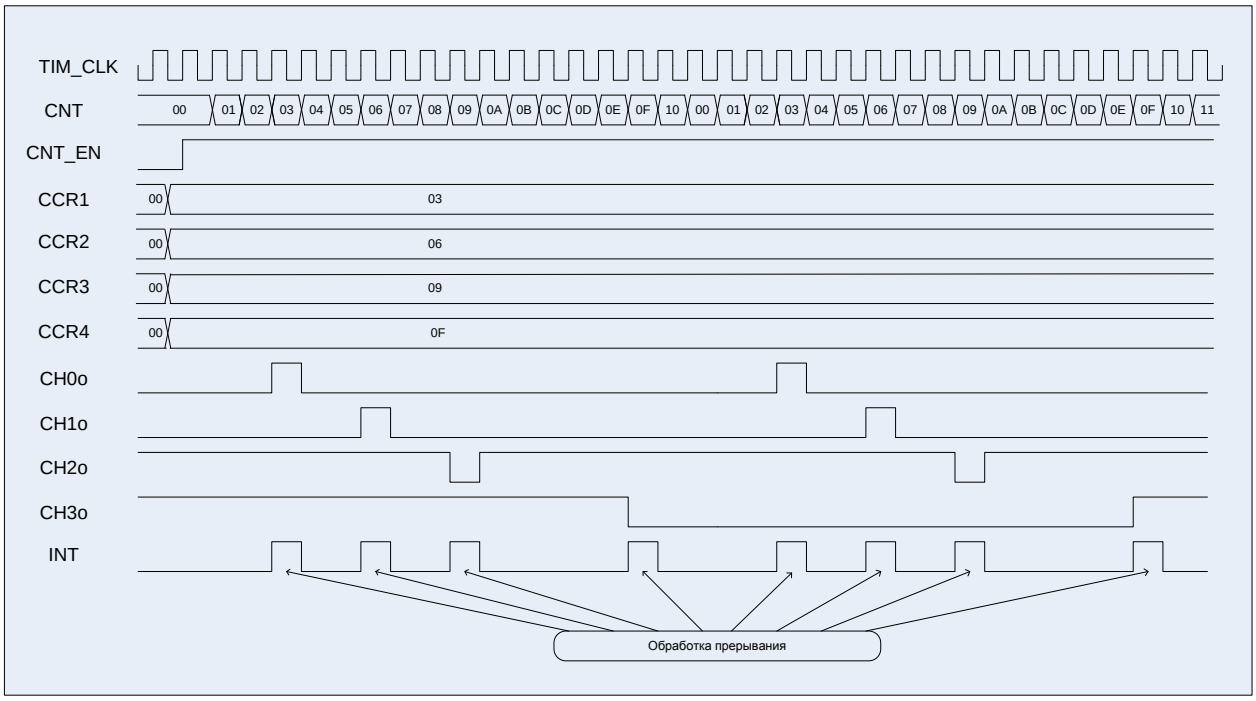


Рисунок 25–28 – Диаграммы примера работы в режиме ШИМ

25.7 Описание регистров блока таймера

Таблица 25-1 – Базовые адреса и смещения регистров управления таймера

Адрес	Название	Описание
0x4007_0000	MDR_TIMER1	Контроллер Timer1
0x4007_8000	MDR_TIMER 2	Контроллер Timer2
0x4008_0000	MDR_TIMER 3	Контроллер Timer3
Смещение		
0x00	MDR_TIMERx->CNT[15:0]	**MDR_TIMERx->CNT** Основной счетчик таймера
0x04	MDR_TIMERx->PSG[15:0]	**MDR_TIMERx->PSG** Делитель частоты при счете основного счетчика
0x08	MDR_TIMERx->ARR[15:0]	**MDR_TIMERx->ARR** Основание счета основного счетчика
0x0C	MDR_TIMERx->CNTRL[7:0]	**MDR_TIMERx->CNTRL** Регистр управления основного счетчика
0x10	CCR1[15:0]	MDR_TIMERx->CCRy Регистр сравнения, захвата для 1 канала таймера
0x14	CCR2[15:0]	MDR_TIMERx->CCRy Регистр сравнения, захвата для 2 канала таймера
0x18	CCR3[15:0]	MDR_TIMERx->CCRy Регистр сравнения, захвата для 3 канала таймера
0x1C	CCR4[15:0]	MDR_TIMERx->CCRy Регистр сравнения, захвата для 4 канала таймера
0x20	CH1_CNTRL[15:0]	**MDR_TIMERx->CHy_CNTRL** Регистр управления для 1 канала таймера
0x24	CH2_CNTRL[15:0]	**MDR_TIMERx->CHy_CNTRL** Регистр управления для 2 канала таймера
0x28	CH3_CNTRL[15:0]	**MDR_TIMERx->CHy_CNTRL** Регистр управления для 3 канала таймера
0x2C	CH4_CNTRL[15:0]	**MDR_TIMERx->CHy_CNTRL** Регистр управления для 4 канала таймера
0x30	CH1_CNTRL1[15:0]	**MDR_TIMERx->CHy_CNTRL1** Регистр управления 1 для 1 канала таймера
0x34	CH2_CNTRL1[15:0]	**MDR_TIMERx->CHy_CNTRL1** Регистр управления 1для 2 канала таймера
0x38	CH3_CNTRL1[15:0]	**MDR_TIMERx->CHy_CNTRL1** Регистр управления 1 для 3 канала таймера
0x3C	CH4_CNTRL1[15:0]	**MDR_TIMERx->CHy_CNTRL1** Регистр управления 1 для 4 канала таймера
0x40	CH1_DTG[15:0]	**MDR_TIMERx->CHy_DTG** Регистр управления DTG для 1 канала таймера
0x44	CH2_DTG[15:0]	**MDR_TIMERx->CHy_DTG** Регистр управления DTG для 2 канала таймера
0x48	CH3_DTG[15:0]	**MDR_TIMERx->CHy_DTG** Регистр управления DTG для 3 канала таймера
0x4C	CH4_DTG[15:0]	**MDR_TIMERx->CHy_DTG** Регистр управления DTG для 4 канала таймера
0x50	BRKETR_CNTRL[15:0]	

		MDR_TIMERx->BRKETR_CNTRL Регистр управления входом BRK и ETR
0x54	STATUS[15:0]	**MDR_TIMERx-**>STATUS Регистр статуса таймера
0x58	IE[15:0]	**MDR_TIMERx->IE** Регистр разрешения прерывания таймера
0x5C	DMA_RE[15:0]	**MDR**_TIMERx->DMA_RE Регистр разрешения запросов DMA от прерываний таймера
0x60	CH1_CNTRL2[15:0]	**MDR_TIMERx->CHy**_CNTRL2 Регистр управления 2 для 1 канала таймера
0x64	CH2_CNTRL2[15:0]	**MDR_TIMERx->CHy**_CNTRL2 Регистр управления 2 для 2 канала таймера
0x68	CH3_CNTRL2[15:0]	**MDR_TIMERx->CHy**_CNTRL2 Регистр управления 2 для 3 канала таймера
0x6C	CH4_CNTRL2[15:0]	**MDR_TIMERx->CHy**_CNTRL2 Регистр управления 2 для 4 канала таймера
0x70	CCR11[15:0]	**MDR_TIMERx->CCRy1** Регистр сравнения 1, захвата для 1 канала таймера
0x74	CCR21[15:0]	**MDR_TIMERx->CCRy1** Регистр сравнения 1, захвата для 2 канала таймера
0x78	CCR31[15:0]	**MDR_TIMERx->CCRy1** Регистр сравнения 1, захвата для 3 канала таймера
0x7C	CCR41[15:0]	**MDR_TIMERx->CCRy1** Регистр сравнения 1, захвата для 4 канала таймера

25.7.1 MDR_TIMERx->CNT

Таблица 25-2 – Основной счетчик таймера CNT

Номер	31...16	15... 0
Доступ	U	R/W
Сброс	0	0
	-	**CNT[15:0]**

Таблица 25-3 – Описание бит регистра CNT

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31…16	-	Зарезервировано
15…0	CNT[7:0]	Значение основного счетчика таймера

25.7.2 MDR_TIMERx->PSG

Таблица 25-4 – Делитель частоты при счете основного счетчика PSG

Номер	31...16	15... 0
Доступ	U	R/W
Сброс	0	0
	-	**PSG[15:0]**

Таблица 25-5 – Описание бит регистраPSG

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31…16	-	Зарезервировано
15…0	PSG[7:0]	Значение предварительного делителя счетчика. Основной счетчик считает на частоте: CLK = TIM_CLK/(PSG+1)

25.7.3 MDR_TIMERx->ARR

Таблица 25-6 – Основание счета основного счетчика ARR

Номер	31...16	15... 0
Доступ	U	R/W
Сброс	0	0
	-	**ARR[15:0]**

Таблица 25-7 – Описание бит регистра ARR

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31…16	-	Зарезервировано
15…0	ARR[7:0]	Основание счета для основного счетчика: CNT = [0…ARR]

25.7.4 MDR_TIMERx->CNTRL

Таблица 25-8 – Регистр управления основного счетчика CNTRL

Номер	31..12	11..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
Сброс	0	0	0	0	0	0	0	0
	-	**EVENT SEL[3:0]**	**CNT MODE[1:0]**	**FDTS [1:0]**	**DIR**	**WR CMPL**	**ARRB EN**	**CNT EN**

Таблица 25-9 – Описание бит регистра CNTRL

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31...11	-	Зарезервировано
11...8	EVENT_SEL [3:0]	Биты выбора источника событий: 0000 – всегда “0”; 0001 – CNT == ARR в таймере 1; 0010 – CNT == ARR в таймере 2; 0011 – CNT == ARR в таймере 3; 0100 – событие на первом канале «Режим 1»; 0101 – событие на втором канале «Режим 1»;; 0110 – событие на третьем канале «Режим 1»; 0111 – событие на четвертом канале «Режим 1»; 1000 – событие на ETR «Режим 2»
7...6	CNT_MODE [1:0]	Режим счета основного счетчика: 00 – счетчик вверх при DIR=0 (при PSG = 0) счетчик вниз при DIR=1 (при PSG = 0); 01 – счетчик вверх/вниз с автоматическим изменением DIR при PSG = 0; 10 – счетчик вверх при DIR=0 (при EVENT = 1) счетчик вниз при DIR=1 (при EVENT = 1); 11 – счетчик вверх/вниз с автоматическим изменением DIR (при EVENT = 1)
5...4	FDTS[1:0]	Частота семплирования данных FDTS: 00 – каждый TIM_CLK; 01 – каждый второй TIM_CLK; 10 – каждый третий TIM_CLK; 11 – каждый четвертый TIM_CLK
3	DIR	Направление счета основного счетчика: 0 – вверх, от 0 до ARR; 1 – вниз, от ARR до 0
2	WR_CMPL	Окончание записи, при задании нового значения регистров CNT, PSG и ARR: 0 – новые данные можно записывать; 1 – данные не записаны и идет запись
1	ARRB_EN	Разрешение мгновенного обновления 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

25.7.5 MDR_TIMERx->CCRy

Таблица 25-10 – Регистр сравнения/захвата для ‘y’ канала таймера CCRy

Номер	31...16	15... 0
Доступ	U	R/W
Сброс	0	0
	-	**CCR[15:0]**

Таблица 25-11 – Описание бит регистра CCRy

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31…16	-	Зарезервировано
15…0	CCR[15:0]	Значение CCR, с которым сравнивается CNT при работе в режиме ШИМ. Значение CNT при котором произошел факт захвата события, в режиме захвата.

25.7.6 MDR_TIMERx->CCRy1

Таблица 25-12 – Регистр сравнения/захвата для ‘y’ канала таймера CCRy1

Номер	31...16	15...0
Доступ	U	R/W
Сброс	0	0
	-	**CCR1[15:0]**

Таблица 25-13 – Описание бит регистра CCRy1

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31...16	-	Зарезервировано
15…0	CCR1[15:0]	Значение CCR1, с которым сравнивается CNT при работе в режиме ШИМ. Значение CNT при котором произошел факт захвата события, в режиме захвата.

25.7.7 MDR_TIMERx->CHy_CNTRL

Таблица 25-14 – Регистр управления для ‘y’ канала таймера CHy_CNTRL

Номер	31...16	15	14	13	12	11…9	8	7…6	5…4	3…0
Доступ	U	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	0
	-	**CAP nPWM**	**WR CMPL**	**ETREN**	**BRKEN**	**OCCM [2:0]**	**OCCE**	**CHPSC [1:0]**	**CHSEL [1:0]**	**CHFLTR [3:0]**

Таблица 25-15 – Описание бит регистра CHy_CNTRL

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31...16	-	Зарезервировано
15	CAP nPWM	Режим работы канала Захват или ШИМ: 1 – канал работает в режиме Захват; 0 – канал работает в режиме ШИМ
14	WR CMPL	Флаг окончания записи, при задании нового значения регистра CCR6 1 – данные не записаны и идет запись; 0 – новые данные можно записывать
13	ETREN	Разрешения сброса по выводу ETR: 0 – запрещен сброс; 1 – разрешен
12	BRKEN	Разрешение сброса по выводу BRK: 0 – запрещен сброс; 1 – разрешен
11…9	OCCM[2:0]	Формат выработки сигнала REF в режиме ШИМ: 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 вверх), CNT<CCR, иначе 0; 0, если DIR= 1 (счет вниз), CNT<CCR, иначе 1; 111 – 0, если DIR= 0 (счет вверх), CNT<CCR, иначе 1; 1, если DIR= 1 (счет вниз), 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 – 1, если DIR = 0 (счет вверх), CCR1< CNT< CCR,иначе 0; 0, если DIR = 1 (счет вниз), CCR < CNT < CCR1, иначе 1; 111 – 0, если DIR = 0 (счет вверх), CCR1< CNT < CCR, иначе 1; 1, если DIR = 1 (счет вниз), CCR< CNT< CCR1, иначе 0

8	OCCE	Разрешение работы ETR: 0 – запрет ETR; 1 – разрешение ETR
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: 00 – положительный фронт ; 01 – отрицательный фронт; 10 – положительный фронт от других каналов; Для первого канала от 2 канала; Для второго канала от 3 канала; Для третьего канала от 4 канала; Для четвертого канала от 1 канала; 11 – положительный фронт от других каналов; Для первого канала от 3 канала; Для второго канала от 4 канала; Для третьего канала от 1 канала; Для четвертого канала от 2 канала
3…0	CHFLTR[3:0]	Сигнал зафиксирован: 0000 – в 1 триггере на частоте TIM_CLK; 0001 – в 2 триггерах на частоте TIM_CLK; 0010 – в 4 триггерах на частоте TIM_CLK; 0011 – в 8 триггерах на частоте TIM_CLK; 0100 – в 6 триггерах на частоте FDTS/2; 0101 – в 8 триггерах на частоте FDTS/2; 0110 – в 6 триггерах на частоте FDTS/4; 0111 – в 8 триггерах на частоте FDTS/4; 1000 – в 6 триггерах на частоте FDTS/8; 1001 – в 8 триггерах на частоте FDTS/8; 1010 – в 5 триггерах на частоте FDTS/16; 1011 – в 6 триггерах на частоте FDTS/16; 1100 – в 8 триггерах на частоте FDTS/16; 1101 – в 5 триггерах на частоте FDTS/32; 1110 – в 6 триггерах на частоте FDTS/32; 1111 – в 8 триггерах на частоте FDTS/32

25.7.8 MDR_TIMERx->CHy_CNTRL1

Таблица 25-16 – Регистр управления 1 для ‘y’ 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	0	0	0	0	0	0
	-	**NINV**	**NSELO [1:0]**	**NSELOE [1:0]**	**-**	**INV**	**SELO [1:0]**	**SELOE [1:0]**

Таблица 25-17 – Описание бит регистра CHy_CNTRL1

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31...13	-	Зарезервировано
12	NINV	Режим выходной инверсии инверсного канала nCHy: 0 – выход не инвертируется; 1 – выход инвертируется
11..10	NSELO[1:0]	Режим работы выхода инверсного канала nCHy: 00 – всегда на выход выдается 0, канал на выход не работает; 01 – всегда на выход выдается 1, канал всегда работает на выход; 10 – на выход выдается сигнал REF; 11 - на выход выдается сигнал с DTG
9…8	NSELOE[1:0]	Режим работы инверсного канала nCHy на выход 00 – всегда на nCHyOE выдается 0, канал на выход не работает; 01 – всегда на nCHyOE выдается 1, канал всегда работает на выход; 10 – на nCHyOE выдается сигнал REF, при REF = 0 третье состояние, при REF = 1 выход; 11 - на nCHyOE выдается сигнал с DTG, при nCHyOE = 0 третье состояние, при nCHyOE = 1 выход
7…5	-	Зарезервировано
4	INV	Режим выходной инверсии прямого канала CHy: 0 – выход не инвертируется; 1 – выход инвертируется
3…2	SELO[1:0]	Режим работы выхода прямого канала CHy: 00 – всегда на выход выдается 0, канал на выход не работает; 01 – всегда на выход выдается 1, канал всегда работает на выход; 10 – на выход выдается сигнал REF; 11 – на выход выдается сигнал с DTG
1…0	SELOE[1:0]	Режим работы прямого канала CHy на выход: 00 – всегда на CHyOE выдается 0, канал на выход не работает; 01 – всегда на CHyOE выдается 1, канал всегда работает на выход; 10 – на CHyOE выдается сигнал REF, при REF = 0 третье состояние, при REF = 1 выход; 11 – на CHyOE выдается сигнал с DTG, при CHyOE = 0 третье состояние, при CHyOE = 1 выход

25.7.9 MDR_TIMERx->CHy_CNTRL2

Таблица 25-18 – Регистр управления 2 для ‘y’ канала таймера CHy_CNTRL2

Номер	31... 4	3	2	1…0
Доступ	U	R/W	R/W	R/W
Сброс	0	0	0	00
	-	**CRRRLD**	**CCR1_EN**	**CHSEL [1:0]**

Таблица 25-19 – Описание бит регистра CHy_CNTRL2

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31...4	-	Зарезервировано
3	CRRRLD	Разрешение обновления регистров CCR и CCR1: 0 – обновление возможно в любой момент времени; 1 – обновление будет осуществлено только при CNT = 0
2	CCR1_EN	Разрешение работы регистра CCR1: 0 – CCR1 не используется; 1 – CCR1 используется
1…0	CHSEL1[1:0]	Выбор события по входному каналу для CAP1: 00 – положительный фронт по Chi; 01 – отрицательный фронт по Chi; 10 – отрицательный фронт от других каналов: - для первого канала от 2 канала; - для второго канала от 3 канала; - для третьего канала от 4 канала; - для четвертого канала от 1 канала. 11 – отрицательный фронт от других каналов: - для первого канала от 3 канала; - для второго канала от 4 канала; - для третьего канала от 1 канала; - для четвертого канала от 2 канала

25.7.10 MDR_TIMERx->CHy_DTG

Таблица 25-20 – Регистр CHy_DTG управления DTG

Номер	31...16	15…8	7…5	4	3…0
Доступ	U	R/W	U	R/W	R/W
Сброс	0	0	0	0	0
	-	**DTG[7:0]**	**-**	**EDTS**	**DTGx[3:0]**

Таблица 25-21 – Описание бит регистра CHy_DTG

№ 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. Задержка DTGdel = DTG*(DTGx+1)
7…5	-	Зарезервировано
4	EDTS	Частота работы DTG: 0 – TIM_CLK; 1 – FDTS
3…0	DTGx [3:0]	Предварительный делитель частоты DTGx

25.7.11 MDR_TIMERx->BRKETR_CNTRL

Таблица 25-22 – 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	0	0	0	0
	-	**ETR FILTER [3:0]**	**ETR PSC [1:0]**	**ETR INV**	**BRK INV**

Таблица 25-23 – Описание бит регистра BRKETR_CNTRL

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31...8	-	Зарезервировано
7…4	ETR FILTER[3:0]	Цифровой фильтр на входе ETR. Сигнал зафиксирован: 0000 – в 1 триггере на частоте TIM_CLK; 0001 – в 2 триггерах на частоте TIM_CLK; 0010 – в 4 триггерах на частоте TIM_CLK; 0011 – в 8 триггерах на частоте TIM_CLK; 0100 – в 6 триггерах на частоте FDTS/2; 0101 – в 8 триггерах на частоте FDTS/2; 0110 – в 6 триггерах на частоте FDTS/4; 0111 – в 8 триггерах на частоте FDTS/4; 1000 – в 6 триггерах на частоте FDTS/8; 1001 – в 8 триггерах на частоте FDTS/8; 1010 – в 5 триггерах на частоте FDTS/16; 1011 – в 6 триггерах на частоте FDTS/165; 1100 – в 8 триггерах на частоте FDTS/16; 1101 – в 5 триггерах на частоте FDTS/32; 1110 – в 6 триггерах на частоте FDTS/32;

		1111 – в 8 триггерах на частоте FDTS/32
3…2	ETRPSC[1:0]	Асинхронный пред. делитель внешней частоты: 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

25.7.12 MDR_TIMERx->STATUS

Таблица 25-24 – Регистр статуса таймера STATUS

Номер	31…17	16…13	12…9	8…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
	-	**CCR CAP1 EVENT [3:0]**	**CCR REF EVENT [3:0]**	**CCR CAP EVENT [3:0]**	**BRK EVENT**	**ETR FE EVENT**	**ETR RE EVENT**	**CNT ARR EVENT**	**CNT ZERO EVENT**

Таблица 25-25 – Описание бит регистра STATUS

№ 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 переднего фронта на входе CAP1 каналов таймера: 0 – нет события; 1 – есть событие. Сбрасывается записью 0, если запись 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 REF каналов таймера 0 – нет события; 1 – есть событие. Сбрасывается записью 0, если запись одновременно с новым событием. Приоритет у нового события. Бит 0 – первый канал. Бит 3 – четвертый канал

8…5	CCR CAP EVENT[3:0]	Событие переднего фронта на входе CAP каналов таймера: 0 – нет события; 1 – есть событие. Сбрасывается записью 0, если запись одновременно с новым событием. Приоритет у нового события. Бит 0 – первый канал. Бит 3 – четвертый канал
4	BRK EVENT	Состояние входа BRK, синхронизированное по PCLK: 0 – BRK = 0; 1 – BRK = 1. Сбрасывается записью 0, при условии наличия 0 на входе BRK
3	ETR FE EVENT	Событие заднего фронта на входе ETR: 0 – нет события; 1 – есть событие. Сбрасывается записью 0, если запись 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 одновременно с новым событием. Приоритет у нового события
1	CNT ARR EVENT	Событие совпадения CNT с ARR: 0 – нет события; 1 – есть событие. Сбрасывается записью 0, если запись одновременно с новым событием совпадения. Приоритет у нового события. Если с момента совпадения до момента программного сброса 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 одновременно с новым событием совпадения. Приоритет у нового события. Если с момента совпадения до момента программного сброса CNT не изменил состояния, то флаг повторно не взводится

25.7.13 MDR_TIMERx->IE

Таблица 25-26 – Регистр разрешения прерывания таймера IE

Номер	31…17	16…13	12…9	8…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
	-	**CCR CAP1 EVENT IE [3:0]**	**CCR REF EVENT IE [3:0]**	**CCR CAP EVENT IE [3:0]**	**BRK EVENT IE**	**ETR FE EVENT IE**	**ETR RE EVENT IE**	**CNT ARR EVENT IE**	**CNT ZERO EVENT IE**

Таблица 25-27 – Описание бит регистра IE

№ 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 переднего фронта на выходе CAP1 каналов таймера: 0 – нет прерывания; 1 – прерывание разрешено. Бит 0 – первый канал. Бит 3 – четвертый канал
12…9	CCR REF EVENT IE[3:0]	Флаг разрешения прерывания по событию переднего фронта на выходе REF каналов таймера: 0 – нет прерывания; 1 – прерывание разрешено. Бит 0 – первый канал. Бит 3 – четвертый канал
8…5	CCR CAP EVENT IE [3:0]	Флаг разрешения прерывания по событию переднего фронта на выходе CAP каналов таймера: 0 – нет прерывания; 1 – прерывание разрешено. Бит 0 – первый канал. Бит 3 – четвертый канал
4	BRK EVENT IE	Флаг разрешения по состоянию входа BRK, синхронизированному по PCLK: 0 – нет прерывания; 1 – прерывание разрешено
3	ETR FE EVENT IE	Флаг разрешения прерывания по заднему фронту на входе ETR: 0 – нет прерывания; 1 – прерывание разрешено
2	ETR RE EVENT IE	Флаг разрешения прерывания по переднему фронту на входе ETR: 0 – нет прерывания; 1 – прерывание разрешено
1	CNT ARR EVENT IE	Флаг разрешения прерывания по событию совпадения CNT и ARR: 0 – нет прерывания; 1 – прерывание разрешено
0	CNT ZERO EVENT IE	Флаг разрешения прерывания по событию совпадения CNT и нуля: 0 – нет прерывания; 1 – прерывание разрешено

25.7.14 MDR_TIMERx->DMA_RE

Таблица 25-28 – Регистр DMA_RE разрешения запросов DMA от прерываний таймера

Номер	31…17	16…13	12…9	8…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
	-	**CCR CAP1 EVENT RE [3:0]**	**CCR REF EVENT RE [3:0]**	**CCR CAP EVENT RE [3:0]**	**BRK EVENT RE**	**ETR FE EVENT RE**	**ETR RE EVENT RE**	**CNT ARR EVENT RE**	**CNT ZERO EVENT RE**

Таблица 25-29 – Описание бит регистра DMA_RE

№ 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 переднего фронта на выходе CAP1 каналов таймера: 0 – нет запроса DMA; 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 переднего фронта на выходе REF каналов таймера: 0 – нет запроса DMA; 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 переднего фронта на выходе CAP каналов таймера: 0 – нет запроса DMA; 1 – запрос DMA разрешен. Бит 0 – первый канал. Бит 3 – четвертый канал
4	BRK EVENT RE	Флаг разрешения по состоянию входа BRK, синхронизированному по PCLK: 0 – нет запроса DMA; 1 – запрос DMA разрешен
3	ETR FE EVENT RE	Флаг разрешения запроса DMA по заднему фронту на входе ETR: 0 – нет запроса DMA; 1 – запрос DMA разрешен
2	ETR RE EVENT RE	Флаг разрешения запроса DMA по переднему фронту на входе ETR: 0 – нет запроса DMA; 1 – запрос DMA разрешен
1	CNT ARR EVENT RE	Флаг разрешения запроса DMA по событию совпадения CNT и ARR: 0 – нет запроса DMA; 1 – запрос DMA разрешен

0	CNT ZERO EVENT RE	Флаг разрешения запроса DMA по событию совпадения CNT и нуля: 0 – нет запроса DMA; 1 – запрос DMA разрешен

26 Контроллер MDR_ADC

В микроконтроллере реализовано два 12-ти разрядных АЦП. С помощью АЦП можно оцифровать сигнал от 16-ти 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.

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 CLK. В качестве синхросигнала может выступать частота процессора CPU_CLK, либо частота ADC_CLK. Выбор частоты осуществляется с помощью бита Cfg_REG_CLKS. Часттота CPU_CLK формируется из частоты процессорного ядра делением на коэффициент Cfg_REG_DIVCLK[3:0]. Максимальная частота CLK не может превышать 14 МГц.

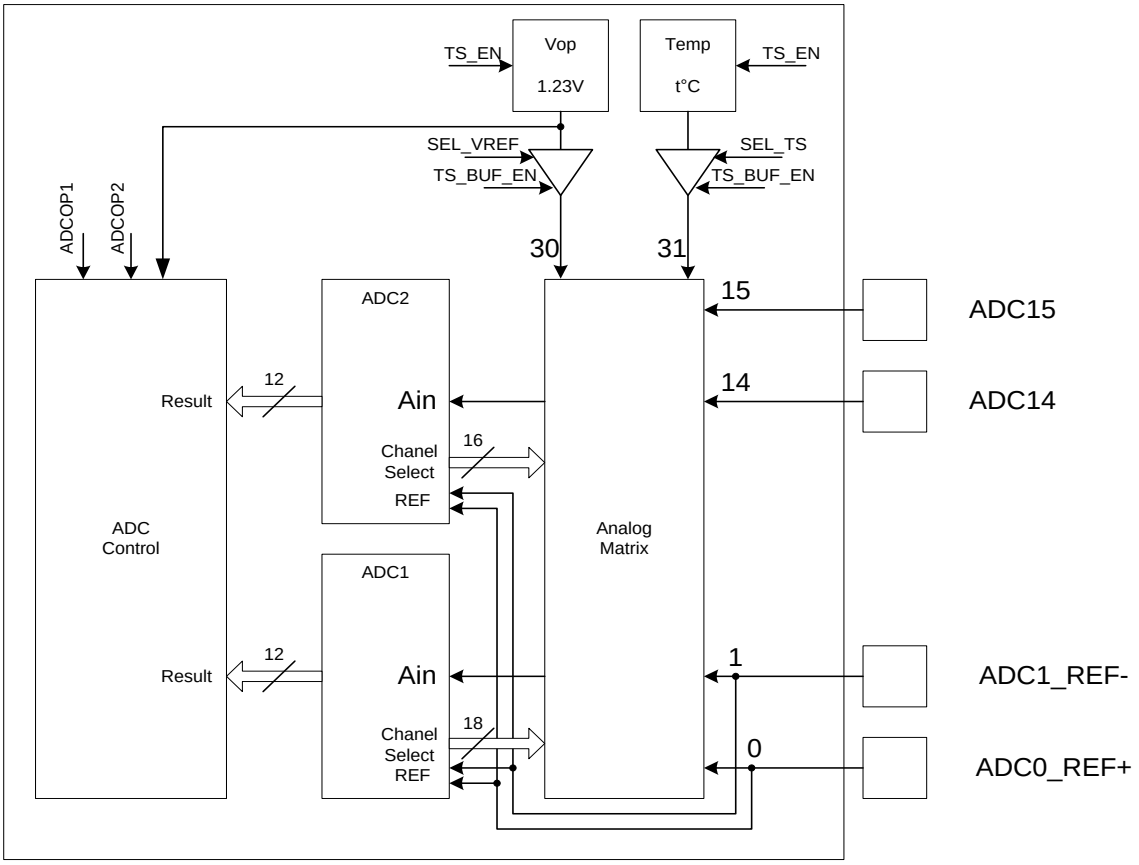


Рисунок 26–1 – Структурная схема контроллера АЦП

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 единицу. Для преобразования необходимо, чтобы выводы, используемые АЦП у порта D, были сконфигурированы как аналоговые и были отключены какие-либо внутренние подтяжки.

26.1 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 перобразования 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.

26.2 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. Преобразование может осуществляться при внутренней опоре бит Cfg_M_REF = 0 и внешней Cfg_M_REF = 1. В этом случая 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_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_SAMPLE.

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.

26.3 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.

26.4 Датчик опорного напряжения

С помощью первого АЦП можно осуществить преобразования источника 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_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.

26.5 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_OP 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.

26.6 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 CPU_CLK, независимо от того на какой частоте ADC_CLK или CPU_CLK 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.

26.7 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.

Приведенная ниже формула позволяет определить максимальное выходное сопротивление источника RAIN для обеспечения качественного преобразования:

RAIN < (TS/(fC ADC*CADC*ln(2N)))-RADC

где: TS – время заряда внутренней емкости в тактах
fC ADC – рабочая частота АЦП
CADC – внутренняя емкость АЦП (~ 15–20 пФ)
N – требуемая точность, в разрядах
RADC – входное сопротивление тракта АЦП (~500 Ом)

Если необходимо обеспечить преобразование с точностью 12 разрядов ± ¼ LSB, то N = 14. Если необходимо обеспечить преобразование с точностью 10 разрядов ± 1 LSB, то N=10. Время заряда TS = определяется битами DelayGo[2:0] и схемой самого АЦП и представлена в Таблица 26-1. Время зарядки внутренней емкости задается битами DelayGo[2:0] определяется в тактах CPU_CLK, независимо от того на какой частоте ADC_CLK или CPU_CLK идет само преобразование.

Таблица 26-1 – Время заряда внутренней емкости АЦП и время преобразования

DelayGo[2:0]	Дополнительная задержка перед началом преобразования	Общее время T_S заряда емкости АЦП перед началом преобразования	Общее время преобразования АЦП
000	1 x CPU_CLK	4 x CLK + 1 x CPU_CLK	28 x CLK + 1 x CPU_CLK
001	2 x CPU_CLK	4 x CLK + 2 x CPU_CLK	28 x CLK + 2 x CPU_CLK
010	3 x CPU_CLK	4 x CLK + 3 x CPU_CLK	28 x CLK + 3 x CPU_CLK
011	4 x CPU_CLK	4 x CLK + 4 x CPU_CLK	28 x CLK + 4 x CPU_CLK
100	5 x CPU_CLK	4 x CLK + 5 x CPU_CLK	28 x CLK + 5 x CPU_CLK
101	6 x CPU_CLK	4 x CLK + 6 x CPU_CLK	28 x CLK + 6 x CPU_CLK
110	7 x CPU_CLK	4 x CLK + 7 x CPU_CLK	28 x CLK + 7 x CPU_CLK
111	8 x CPU_CLK	4 x CLK + 8 x CPU_CLK	28 x CLK + 8 x CPU_CLK

Помимо точности определяемой временем зарядки внутренней емкости АЦП, точность преобразования имеет ошибки, связанные с технологическими разбросами схемы и шумами, и определяемые параметрами EDLADC, EILADC и EOFFADC.

Для корректного задания режимов работы АЦП в регистре ADCx_CFG необходимо сделать до задания бита Go или Cfg_REG_SAMPLE, иначе новая конфигурация будет действовать со следующего преобразования.

26.8 Описание регистров блока контроллера АЦП

Таблица 26-2 – 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

26.8.1 MDR_ADC->ADC1_CFG

Таблица 26-3 – 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 26-4 – Описание бит регистра ADC1_CFG

№ 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 CPU_CLK; 0001 – 2 такта CPU_CLK; … 1111 – 16 тактов CPU_CLK
27...25	Delay Go [2:0]	Дополнительная задержка перед началом преобразования после выбора канала: 000 – 1 такт CPU_CLK; 001 – 2 такта CPU_CLK; … 111 – 8 тактов CPU_CLK
24...21	TR[3:0]	Подстройка опорного напряжения. Смотрите диаграмму зависимости источника опорного напряжения от подстройки (Рисунок 26–2)
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; 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 – CPU_CLK = HCLK; 0001 – CPU_CLK = HCLK/2; 0010 – CPU_CLK = HCLK/4; 0011 – CPU_CLK = HCLK/8; … 1011 – CPU_CLK = HCLK/2048 Остальные CPU_CLK = HCLK;
11	Cfg M_REF	Выбор источника опорных напряжений: 0 – внутренне опорное напряжение (от AUcc и AGND); 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 REG CLKS	Выбор источника синхросигнала CLK работы ADC: 0 – CPU_CLK; 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

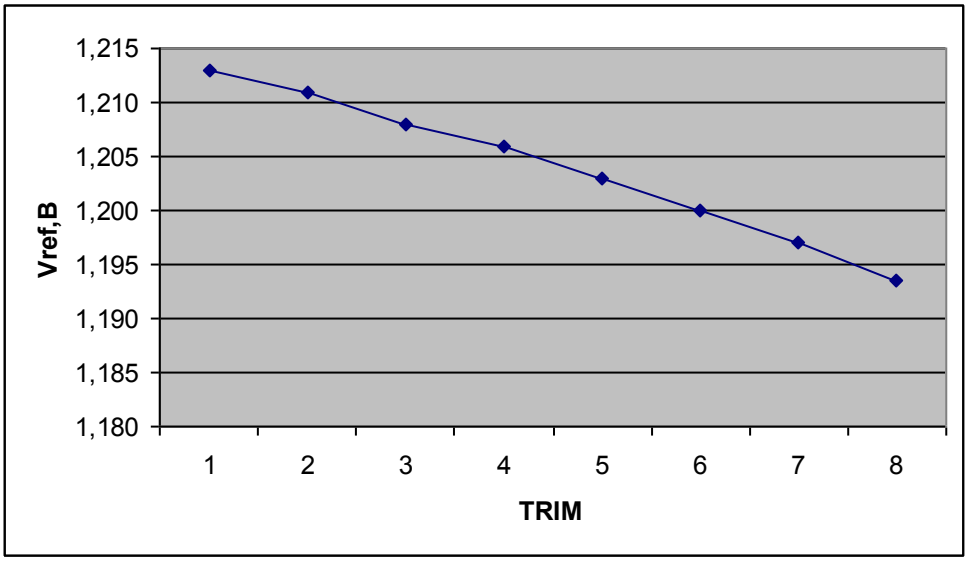


Рисунок 26–2 – Зависимость источника опорного напряжения от подстройки

26.8.2 MDR_ADC->ADC2_CFG

Таблица 26-5 – 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 26-6 – Описание бит регистра ADC2_CFG

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31...28	-	Зарезервировано
27...25	Delay Go [2:0]	Задержка перед началом следующего преобразования после завершения предыдущего при последовательном переборе каналов: 000 – 0 тактов CPU_CLK 001 – 1 такт CPU_CLK … 111 – 7 тактов CPU_CLK
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 – CPU_CLK = HCLK; 0001 – CPU_CLK = HCLK/2; 0010 – CPU_CLK = HCLK/4; 0011 – CPU_CLK = HCLK/8; … 1011 – CPU_CLK = HCLK/2048 Остальные CPU_CLK = HCLK;
11	Cfg M_REF	Выбор источника опорных напряжений: 0 – внутренне опорное напряжение (от AUcc и AGND); 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).
2	Cfg REG CLKS	Выбор источника синхросигнала CLK работы ADC: 0 – CPU_CLK; 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

26.8.3 MDR_ADC->ADCx_H_LEVEL

Таблица 26-7 – Регистр ADCx_H_LEVEL

Номер	31...12	11…0
Доступ	U	R/W
Сброс	0	0
	-	**REG H LEVEL [11:0]**

Таблица 26-8 – Описание бит регистра ADCx_H_LEVEL

№ 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.

26.8.4 MDR_ADC->ADCx_L_LEVEL

Таблица 26-9 – Регистр ADCx_L_LEVEL

Номер	31...12	11…0
Доступ	U	R/W
Сброс	0	0
	-	**REG L LEVEL [11:0]**

Таблица 26-10 – Описание бит регистра ADCx_L_LEVEL

№ 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

26.8.5 MDR_ADC->ADCx_RESULT

Таблица 26-11 – 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 26-12 – Описание бит регистра ADCx_RESULT

№ 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

26.8.6 MDR_ADC->ADCx_STATUS

Таблица 26-13 – 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 26-14 – Описание бит регистра ADCx_STATUS

№ 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 верней 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
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

26.8.7 MDR_ADC->ADCx_CHSEL

Таблица 26-15 – Регистр ADCx_CHSEL

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

Таблица 26-16 – Описание бит регистра ADCx_CHSEL

№ 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

27 Контроллер MDR_DAC

В микроконтроллере 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 AU_{CC}. В режиме работы с внешней опорой Cfg_M_REFx=1 ЦАП формирует выходное напряжение в диапазоне от 0 до значения DACx_REF.

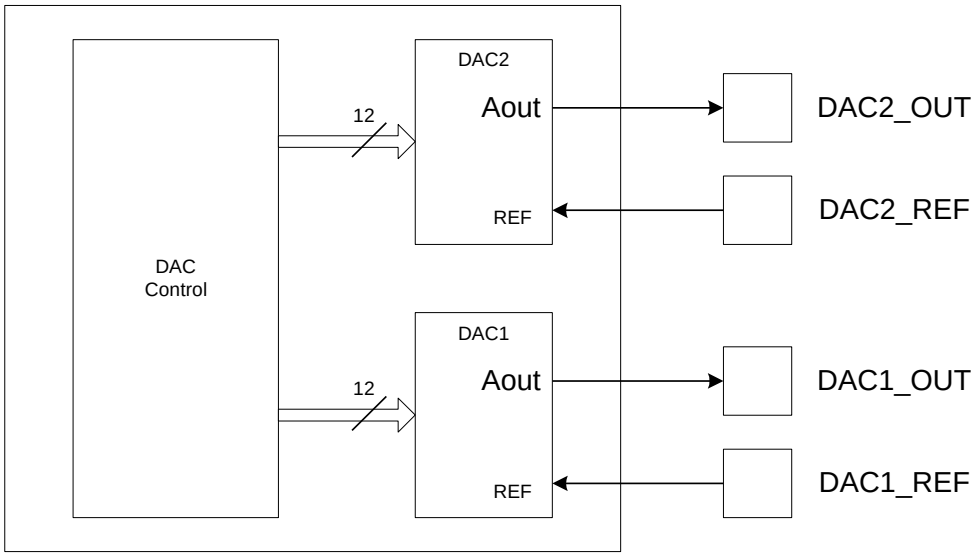


Рисунок 27–1 – Структурная схема контроллера ЦАП

27.1 Описание регистров блока контроллера ЦАП

Таблица 27-1 – 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

27.1.1 MDR_DAC->CFG

Таблица 27-2 – 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 27-3 – Описание бит регистра CFG

№ 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 AUCC; 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 AUCC; 1 – в качестве опорного напряжения используется напряжение на входе опорного напряжения DAC1_REF

27.1.2 MDR_DAC->DAC1_DATA

Таблица 27-4 – Регистр DAC1_DATA

Номер	31…28	27...16	15…12	11...0
Доступ	U	R/W	U	R/W
Сброс	0	0	0	0
	-	**DAC1_DATA[11:0]**	**-**	**DAC0_DATA[11:0]**

Таблица 27-5 – Описание бит регистра DAC1_DATA

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31…28	-	Зарезервировано
27...16	DAC1 DATA[11:0]	Данные DAC1 при Cfg_SYNC_A=1. При чтении всегда равны нулю. Читать из DAC2_DATA
15...12	-	Зарезервировано
11…0	DAC0 DATA[11:0]	Данные DAC0

27.1.3 MDR_DAC->DAC2_DATA

Таблица 27-6 – Регистр DAC2_DATA

Номер	31…28	27...16	15…12	11...0
Доступ	U	R/W	U	R/W
Сброс	0	0	0	0
	-	**DAC0_DATA[11:0]**	**-**	**DAC1_DATA[11:0]**

Таблица 27-7 – Описание бит регистра DAC21_DATA

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31…28	-	Зарезервировано
27...16	DAC0 DATA[11:0]	Данные DAC0 при Cfg_SYNC_A=1. При чтении всегда равны нулю. Читать из DAC1_DATA
15...12	-	Зарезервировано
11…0	DAC1 DATA[11:0]	Данные DAC1

Примечание – Если бит конфигурации Cfg_SYNC_A установлен, то данные для DAC1 и DAC2 задаются записью в один из регистров DACx_DATA.

28 Контроллер схемы компаратора MDR_COMP

В микроконтроллере 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 микроконтроллера 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.

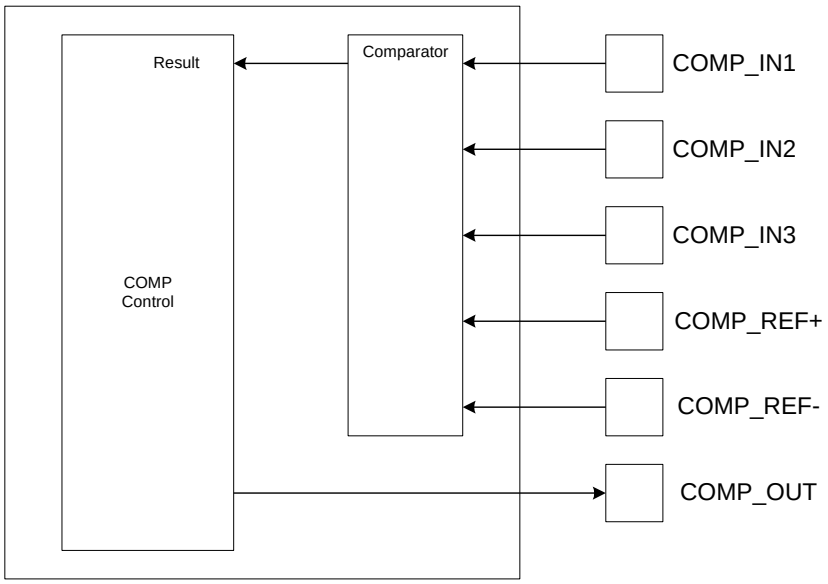
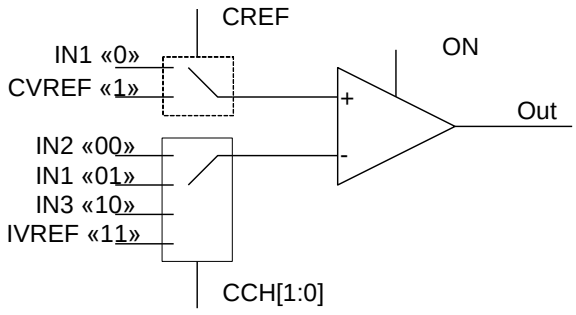


Рисунок 28–1 – Структура блока компаратора



*IVREF – выход внутреннего источника опорного напряжения 1,2 В.

Рисунок 28–2 – Структура мультиплексирования входов компаратора

28.1 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.

28.2 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.

28.3 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.

28.4 Формирование внутренней шкалы напряжений

Внутренняя шкала напряжений формируется на резистивном делителе (см Рисунок 28–3), который включается битом CVREN=1.

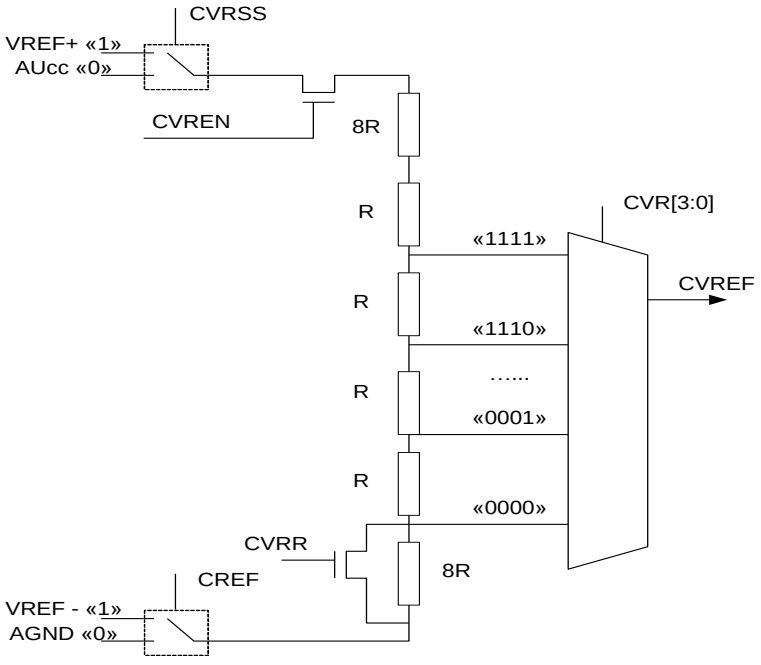


Рисунок 28–3 – Структура блока формирования CVREF

При этом в качестве опорного напряжения делителя может выступать питание микросхемы AUCC (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 приведены в Таблица 28-1, как справочные данные. Реальные значения в конкретном кристалле могут отличаться за счет технологического разброса параметров.

Таблица 28-1 – Формирование внутренней шкалы напряжений CVREF

CVRR	CVR[3:0]	Отношение резисторов	Напряжение CREF при U_{CC}=,.3 В, В	Входной импеданс VREF+, Ом	Примечание
0	0000	8/32	0.83	12K	
	0001	9/32	0.93	13K	
	0010	10/32	1.03	13.8K	
	0011	11/32	1.13	14.4K	
	0100	12/32	1.24	15K	
	0101	13/32	1.34	15.4K	
	0110	14/32	1.44	15.8K	
	0111	15/32	1.55	15.9K	
	1000	16/32	1.65	16K	
	1001	17/32	1.75	15.9K	
	1010	18/32	1.86	15.8K	
	1011	19/32	1.96	15.4K	
	1100	20/32	2.06	15K	
	1101	21/32	2.17	14.4K	
	1110	22/32	2.27	13.8K	
	1111	23/32	2.37	12.9K	
1	0000	0/24	0.00	0.5K	
	0001	1/24	0.14	1.9K	
	0010	2/24	0.28	3.7K	
	0011	3/24	0.41	5.3K	
	0100	4/24	0.55	6.7K	
	0101	5/24	0.69	7.9K	
	0110	6/24	0.83	9K	
	0111	7/24	0.96	9.9K	
	1000	8/24	1.10	10.7K	
	1001	9/24	1.24	11.3K	
	1010	10/24	1.38	11.7K	
	1011	11/24	1.51	11.9K	
	1100	12/24	1.65	12K	
	1101	13/24	1.79	11.9K	
	1110	14/24	1.93	11.7K	
	1111	15/24	2.06	11.3K	

Результат работы компаратора сигнал Out может быть проинвертирован с помощью бита INV и выдан на вывод микросхемы OUT_COMP. Также результат сравнения доступен внутри микроконтроллера. 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_Ich фиксирует событие появления положительного сигнала сравнения и

устанавливается в 1 до тех пор, пока не будет считан регистр COMP_RESULT_LATCH.

28.5 Описание регистров блока контроллера компаратора

Таблица 28-2 – 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

28.5.1 MDR_COMP->CFG

Таблица 28-3 – 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
	-	**CMPIE**	**Ready**	**INV**	**CCH[1:0]**	**CREF**	**CVR[3:0]**	**CVREN**	**CVRSS**	**CVRR**	**ON**

Таблица 28-4 – Описание бит регистра CFG

№ 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 CREF

7…4	CVR [3:0]	Биты выбора сигнал управления мультиплексора выбора CVREF. Смотрите Таблица 28-1 – Формирование внутренней шкалы напряжений CVREF
3	CVREN	Бит разрешения работы источника CVREF: 0 – не разрешен; 1 – разрешен
2	CVRSS	Бит выбора опоры CVREF: 0 – источник CVREF работает в границах AVdd AGND; 1 – источник CVREF работает в границах Vref+ Vref-
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

28.5.2 MDR_COMP->RESULT

Таблица 28-5 – Регистр RESULT

Номер	31…3	2	1	0
Доступ	U	R/W	R/W	R/W
Сброс	0	0	0	0
	-	**Rst_Ich**	**Rslt_As**	**Rslt_Sy**

Таблица 28-6 – Описание бит регистра RESULT

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31…3	-	Зарезервировано
2	Rst_Ich	Значение компарирования хранится до момента считывания из регистра COMP_RESULT_LATCH, после чего сбрасывается. Взводится по переднему фронту сигнала с компаратора
1	Rslt_As	Значение компарирования непосредственно из компаратора
0	Rslt_Sy	Значение результата компарирования, синхранизированное с частотой HCLK

28.5.3 MDR_COMP->RESULT_LATCH

Таблица 28-7 – Регистр RESULT_LATCH

Номер	31…1	0
Доступ	U	R/W
Сброс	0	0
	-	**Rst_Ich**

Таблица 28-8 – Описание бит регистра RESULT_LATCH

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31…1	-	Зарезервировано
0	Rst_Ich	Значение компарирования хранится до момента считывания из

		регистра COMP_RESULT_LATCH, после чего сбрасывается. Взводится по переднему фронту сигнала с компаратора

29 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 3 скоростях:

- нормальная: 100 Kbps;
- быстрая: 400 Kbps;
- очень быстрая: 3.5 Mbps.

29.1 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 обоих 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. «Сигнал START» и «Сигнал STOP»).

29.2 Протокол I2C

Нормальная передача по интерфейсу I2C содержит 4 фазы:

- сигнал START;
- передача адреса;
- передача данных;
- сигнал STOP.

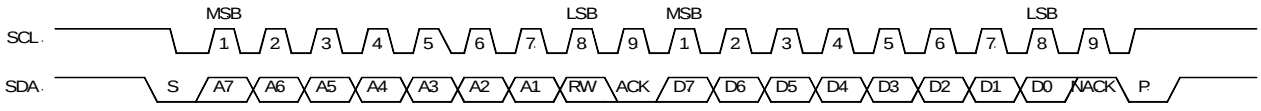


Рисунок 29–1 – Передача по I2C

29.3 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.

29.4 Передача адреса

Первым байтом данных, передаваемым мастером сразу после сигнала START, является адрес ведомого. Это 7-ми битный 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, выставляя 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.

29.5 Передача данных

После успешного подтверждения приема адреса одним ведомым устройством может быть начата передача данных в направлении, задаваемым битом RW в посылке 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.

29.6 Сигнал STOP

Мастер может завершить соединение путем создания сигнала STOP. Сигнал STOP или P бит определяется переходом линии SDA из низкого состояния в высокое, когда SCL находится в высоком состоянии.

29.7 Описание регистров контроллера I2C

Таблица 29-1 – 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

29.7.1 MDR_I2C->PRL

Таблица 29-2 – Регистр PRL

Номер	31…8	7… 0
Доступ	U	R/W
Сброс	0	0
	-	**PR[7:0]**

Таблица 29-3 – Описание бит регистра PRL

№ 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

29.7.2 MDR_I2C->PRH

Таблица 29-4 – Регистр PRH

Номер	31…8	7… 0
Доступ	U	R/W

Сброс	0	0
	-	**PR[15:8]**

Таблица 29-5 – Описание бит регистра PRH

№ 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

29.7.3 MDR_I2C->CTR

Таблица 29-6 – 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 29-7 – Описание бит регистра CTR

№ 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

29.7.4 MDR_I2C->RXD

Таблица 29-8 – Регистр RXD

Номер	31…8	7… 0
Доступ	U	R/W
Сброс	0	0
	-	**RXD[7:0]**

Таблица 29-9 – Описание бит регистра RXD

№ 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

29.7.5 MDR_I2C->STA

Таблица 29-10 – Регистр STA

Номер	31…8	7	6	5	4…2	1	0
Доступ	U	R/W	R/W	R/W	U	R/W	R/W
Сброс	0	0	0	0	0	0	0
	-	**Rx ACK**	**BUSY**	**LOST ARB**	**-**	**TR PROG**	**INT**

Таблица 29-11 – Описание бит регистра STA

№ 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 быт выставляется если: – получен Stop bit, но он не был инициализирован этим контроллером; – Если 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

29.7.6 MDR_I2C->TXD

Таблица 29-12 – Регистр TXD

Номер	31…8	7… 0
Доступ	U	R/W
Сброс	0	0
	-	**TXD[7:0]**

Таблица 29-13 – Описание бит регистра TXD

№ 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

29.7.7 MDR_I2C->CMD

Таблица 29-14 – 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 29-15 – Описание бит регистра CMD

№ 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

30 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-разрядный 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.

30.1 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 30–1.

30.2 Программируемые параметры

Следующие ключевые параметры могут быть заданы программно:

- режим функционирования периферийного устройства – ведущее или ведомое;
- разрешение или запрещение функционирования;
- формат информационного кадра;
- скорость передачи данных;
- фаза и полярность тактового сигнала;
- размер блока данных – от 4 до 16 бит;
- маскирование прерываний.

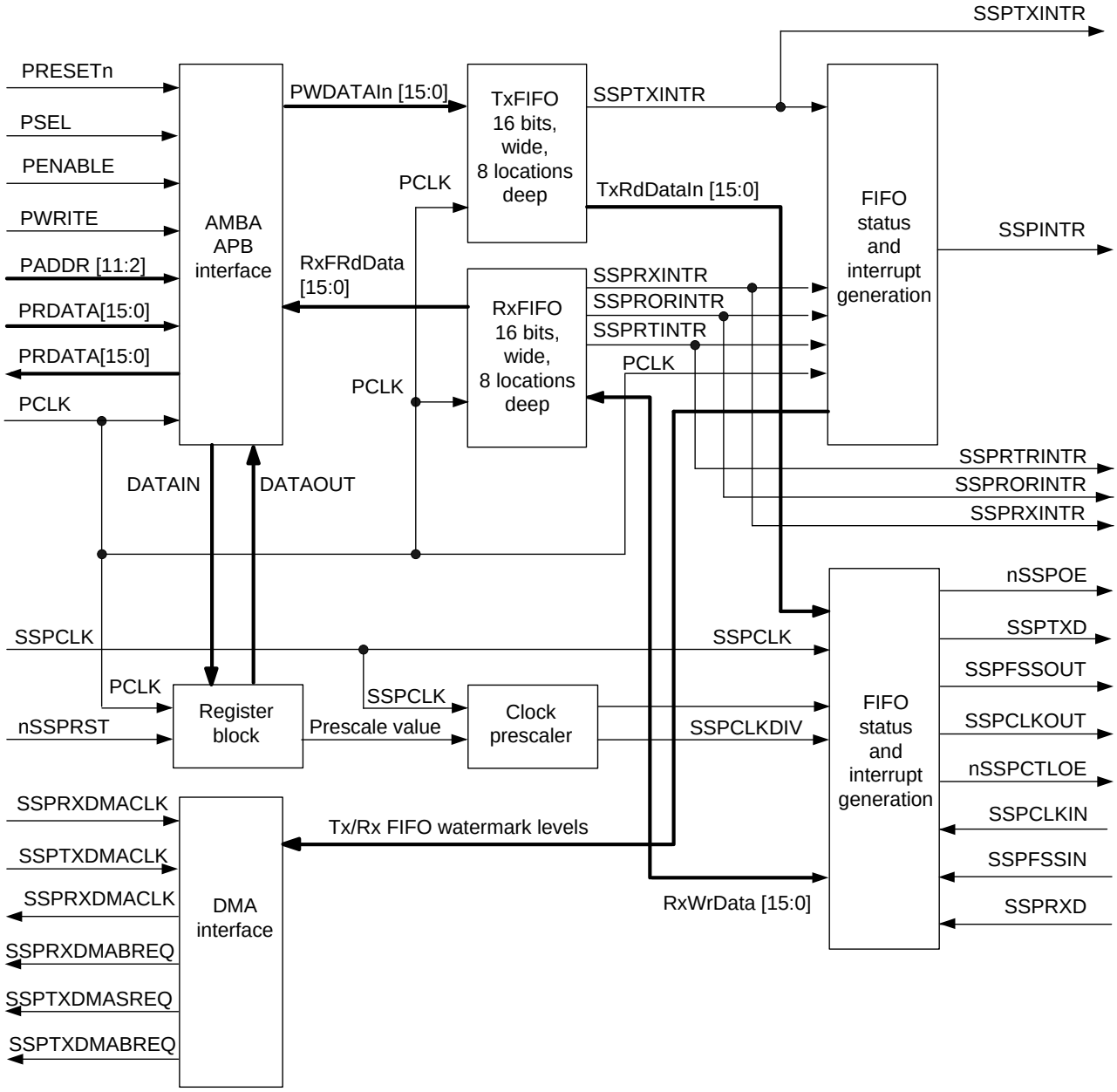


Рисунок 30–1 – Структурная схема модуля SSP

30.3 Характеристики интерфейса SPI

Последовательный синхронный интерфейс SPI фирмы Motorola обеспечивает:
- полнодуплексный обмен данными по четырехпроводной линии;
- программное задание фазы и полярности тактового сигнала.

30.4 Характеристики интерфейса Microwire

Интерфейс Microwire фирмы National Semiconductor обеспечивает:
- полудуплексный обмен данными с использованием восьмибитных управляющих последовательностей.

30.5 Характеристики интерфейса SSI

Интерфейс SSI фирмы Texas Instruments обеспечивает:
- полнодуплексный обмен данными по четырехпроводной линии;
- возможность перевода линии передачи данных в третье (высокоимпедансное) состояние.

30.6 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).

30.6.1 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.

30.6.2 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.

30.6.3 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 в до тех пор, пока не будут считаны центральным процессором.

30.6.4 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, передаётся 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.

30.6.5 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 подаётся на контроллер прерываний NVIC, при этом появляется дополнительная возможность маскирования устройства в целом, что облегчает построение модульных драйверов устройств.

30.6.6 Интерфейс прямого доступа к памяти

Модуль обеспечивает интерфейс с контроллером DMA согласно схеме взаимодействия приемопередатчика и контроллера DMA.

30.6.7 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.

30.6.8 Разрешение работы приемопередатчика

Разрешение осуществляется путем установки бита SSE регистра управления CR1. Буфер FIFO передатчика может быть либо проинициализирован путем записи в него до восьми 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.

30.6.9 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:

FSSPCLK <= FPCLK

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:

FSSPCLK(min) => 2 x FSSPCLKOUT(max) [for master mode]
FSSPCLK(min) => 12 x FSSPCLKIN(max) [for slave mode].

Аналогично, максимальная допустимая частота сигнала SSPCLK определяется следующей системой соотношений, которые должны выполняться одновременно:

FSSPCLK(max) <= 254 x 256 x FSSPCLKOUT(min) [for master mode]
FSSPCLK(max) <= 254 x 256 x FSSPCLKIN(min) [for slave mode].

30.6.10 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).

30.6.11 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).

30.6.12 Формирование тактового сигнала обмена данными

Тактовый сигнал обмена данными формируется путем деления частоты тактового сигнала SSPCLK. На первом этапе формирования частота этого сигнал 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 программирование через CR0.

Частота выходного тактового сигнала обмена данными SSP_CLK определяется следующим соотношением:

FSSPCLKOUT = FSSPCLK / (CPSDVR * (1+SCR)).

Например, в случае, если частота сигнала SSPCLK составляет 3,6864 МГц, а значение CPSDVSR = 2, частота сигнала SSP_CLK лежит в интервале от 7,2 кГц до 1,8432 МГц.

30.6.13 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, передаёт 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.

30.6.14 Формат синхронного обмена SSI фирмы Texas Instruments

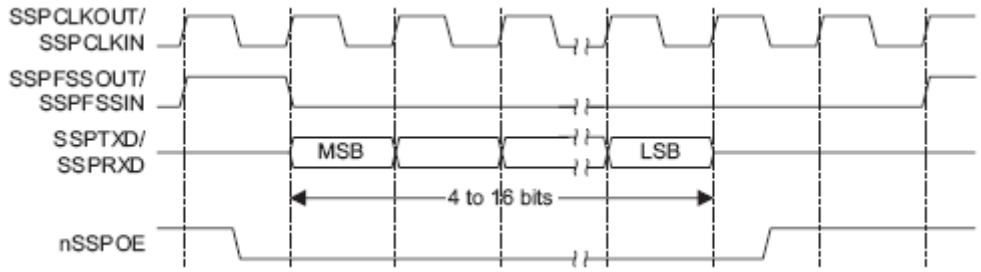


Рисунок 30–2 – Формат синхронного обмена протокола SSI (единичный обмен)

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 (Рисунок 30–2, Рисунок 30–3).

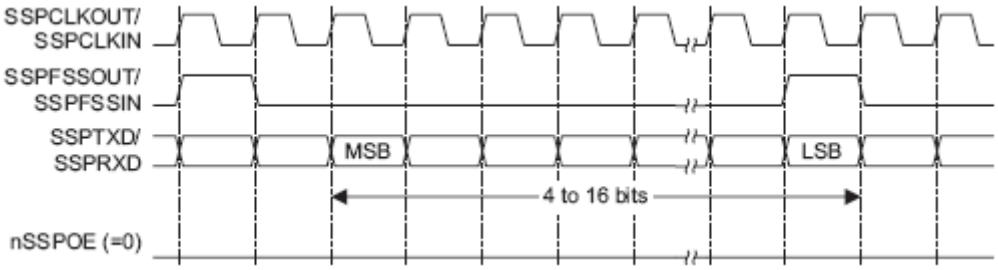


Рисунок 30–3 – Формат синхронного обмена протокола SSI (непрерывный обмен)

30.6.15 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.

30.6.16 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 30–4, Рисунок 30–5).

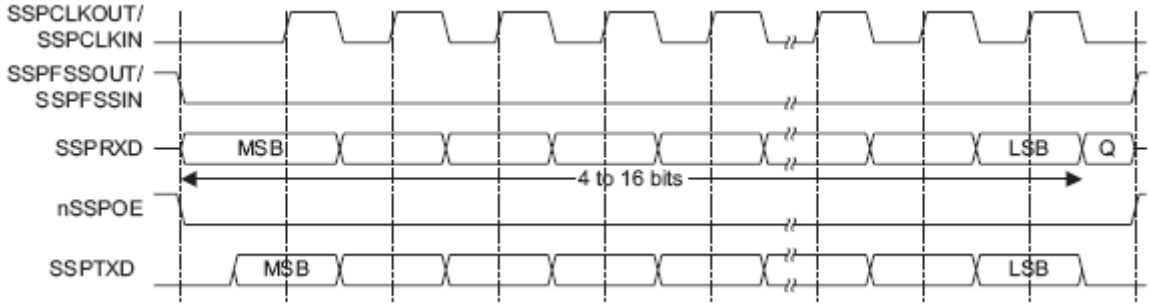


Рисунок 30–4 – Формат синхронного обмена протокола SPI, SPO=0, SPH=0 (одиночный обмен)

Примечание – На рисунке буквой Q обозначен сигнал с неопределенным уровнем.

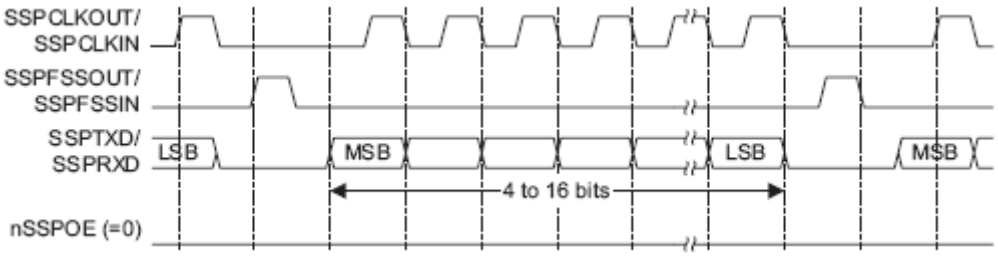


Рисунок 30–5 – Формат синхронного обмена протокола SPI, SPO=0, SPH=0 (непрерывный обмен)

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.

30.6.17 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 30–6 – одиночный и непрерывный обмен.

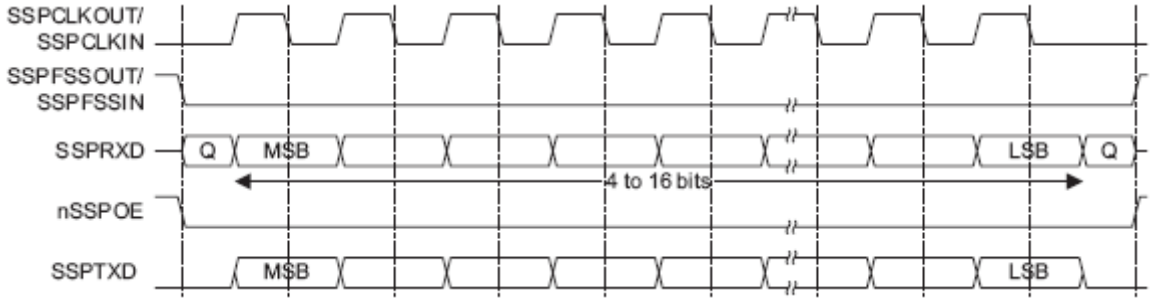


Рисунок 30–6 – Формат синхронного обмена протокола SPI, SPO=0, SPH=1

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 полутакта 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.

30.6.18 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: Рисунок 30–7 – одиночный обмен и Рисунок 30–8 – непрерывный обмен.

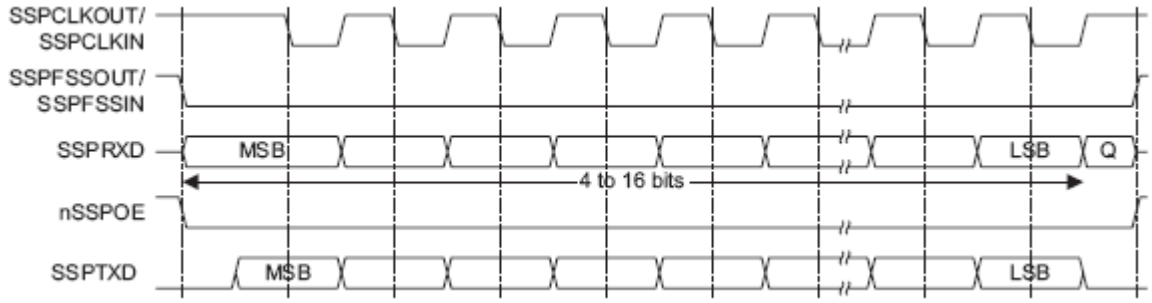


Рисунок 30–7 – Формат синхронного обмена протокола SPI, SPO=1, SPH=0 (одиночный обмен)

Примечание – На рисунке буквой Q обозначен сигнал с неопределенным уровнем.

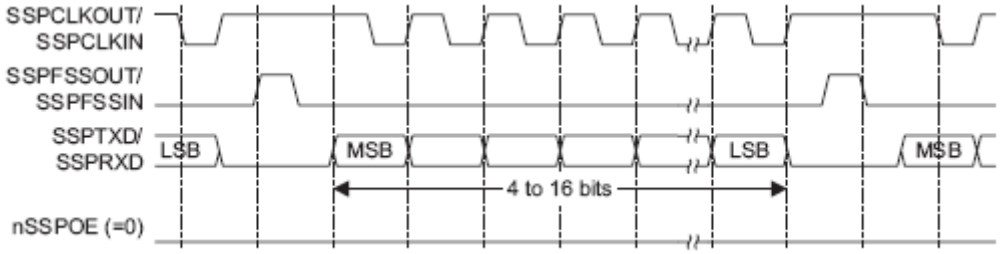


Рисунок 30–8 – Формат синхронного обмена протокола SPI, SPO=1, SPH=0 (непрерывный обмен)

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.

30.6.19 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 30–9 – одиночный и непрерывный обмен.

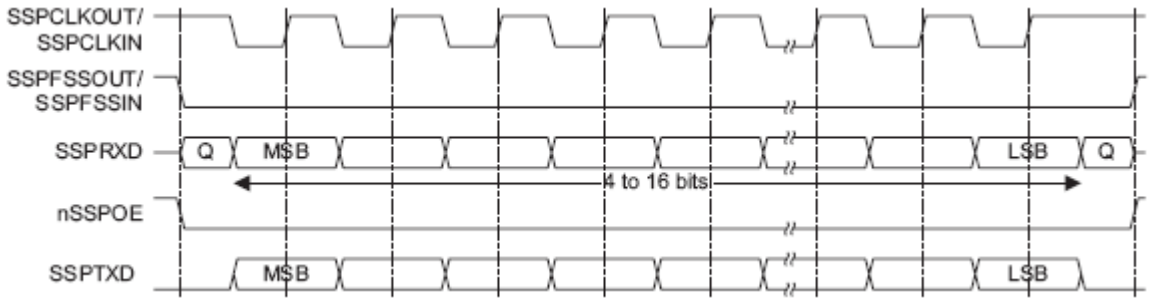


Рисунок 30–9 – Формат синхронного обмена протокола SPI, SPO=1, SPH=1

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.

30.6.20 Формат синхронного обмена Microwire фирмы National Semiconductor

Временные диаграммы последовательного синхронного обмена в режиме Microwire показаны на Рисунок 30–10 – одиночный обмен и на Рисунок 30–11 – непрерывный обмен.

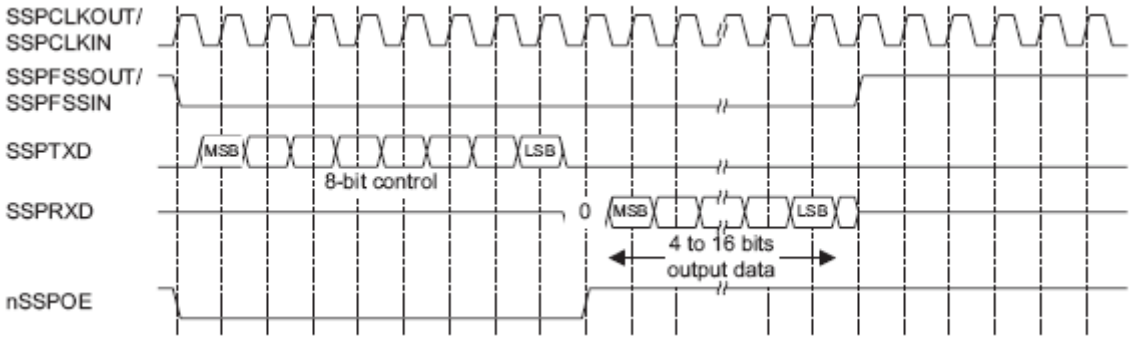


Рисунок 30–10 – Формат синхронного обмена протокола Microwire (одиночный обмен)

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 так же, 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.

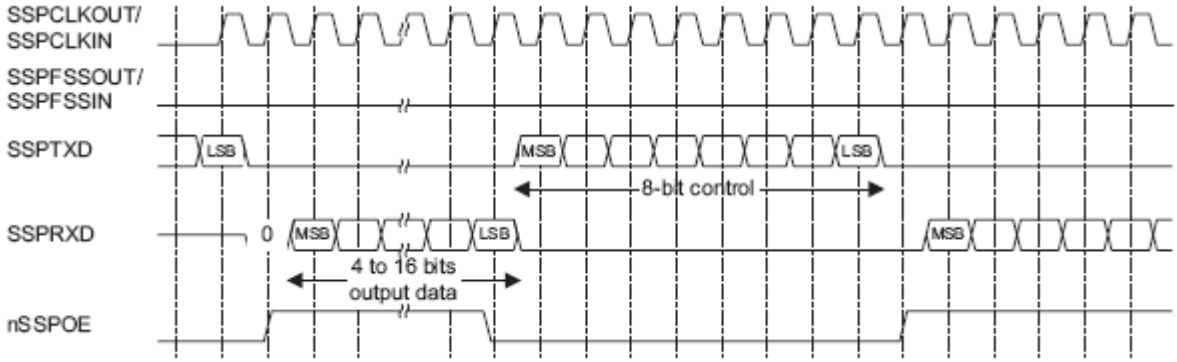


Рисунок 30–11 – Формат синхронного обмена протокола Microwire (непрерывный обмен)

30.6.20.1 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_CKL, должны гарантировать достаточное время установки и удержания сигнала SSP_FSS по отношению к переднему фронту сигнала SSP_CLK.

Данные требования иллюстрирует Рисунок 30–12. 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.

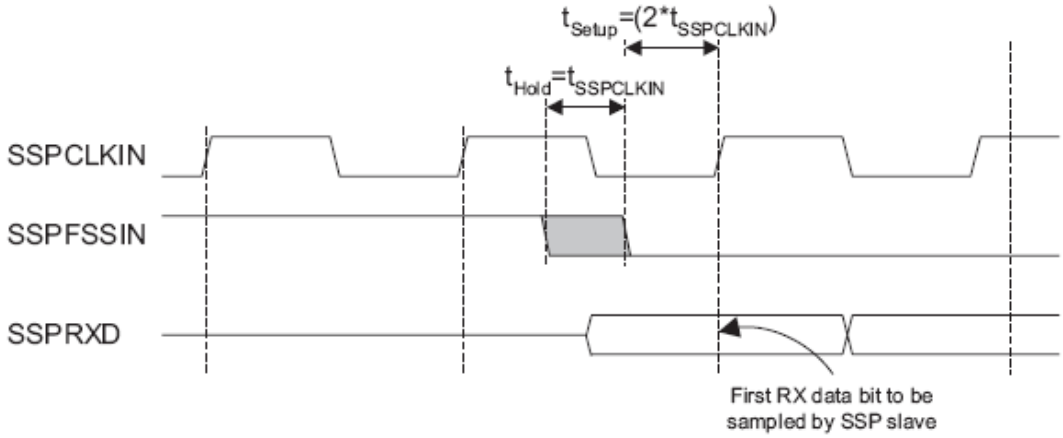


Рисунок 30–12 – Формат Microwire, требования к времени установки и удержания сигнала

30.6.21 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 30–13, Рисунок 30–14 и Рисунок 30–15).

Примечание – Модуль SSP не поддерживает динамическое изменение режима «ведущий – ведомый». Каждый приемопередатчик должен быть изначально сконфигурирован в одном из этих режимов.

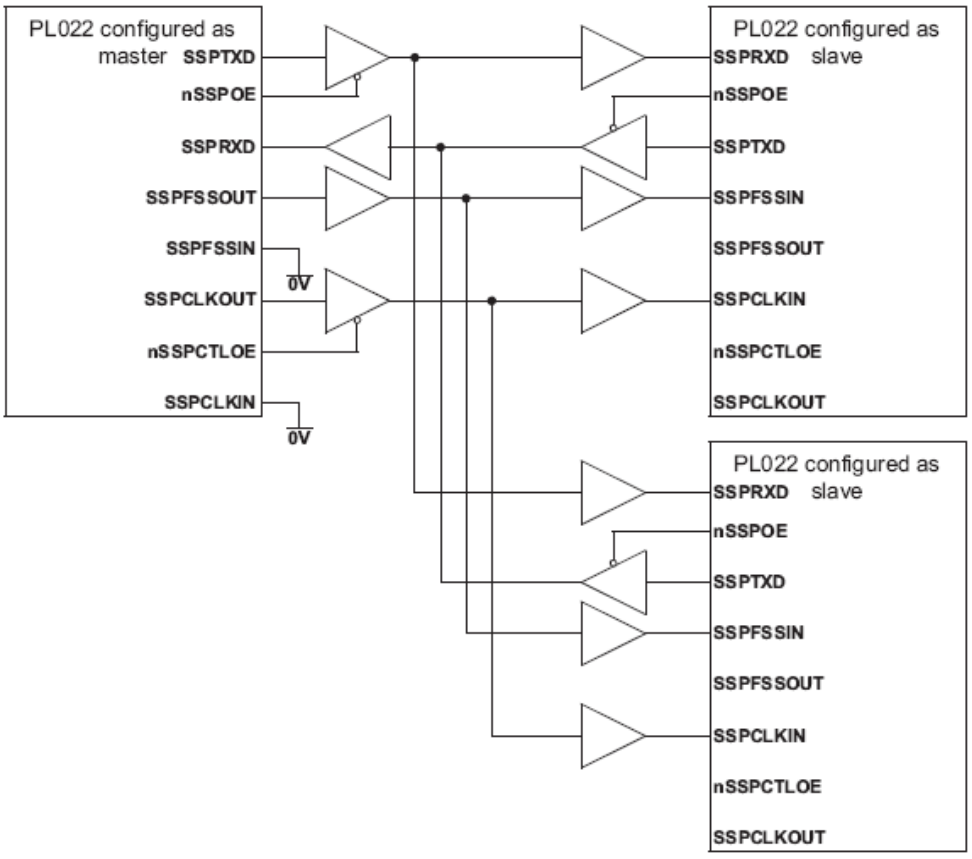


Рисунок 30–13 – Ведущее устройство SSP подключено к двум ведомым

Рисунок 30–13 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.

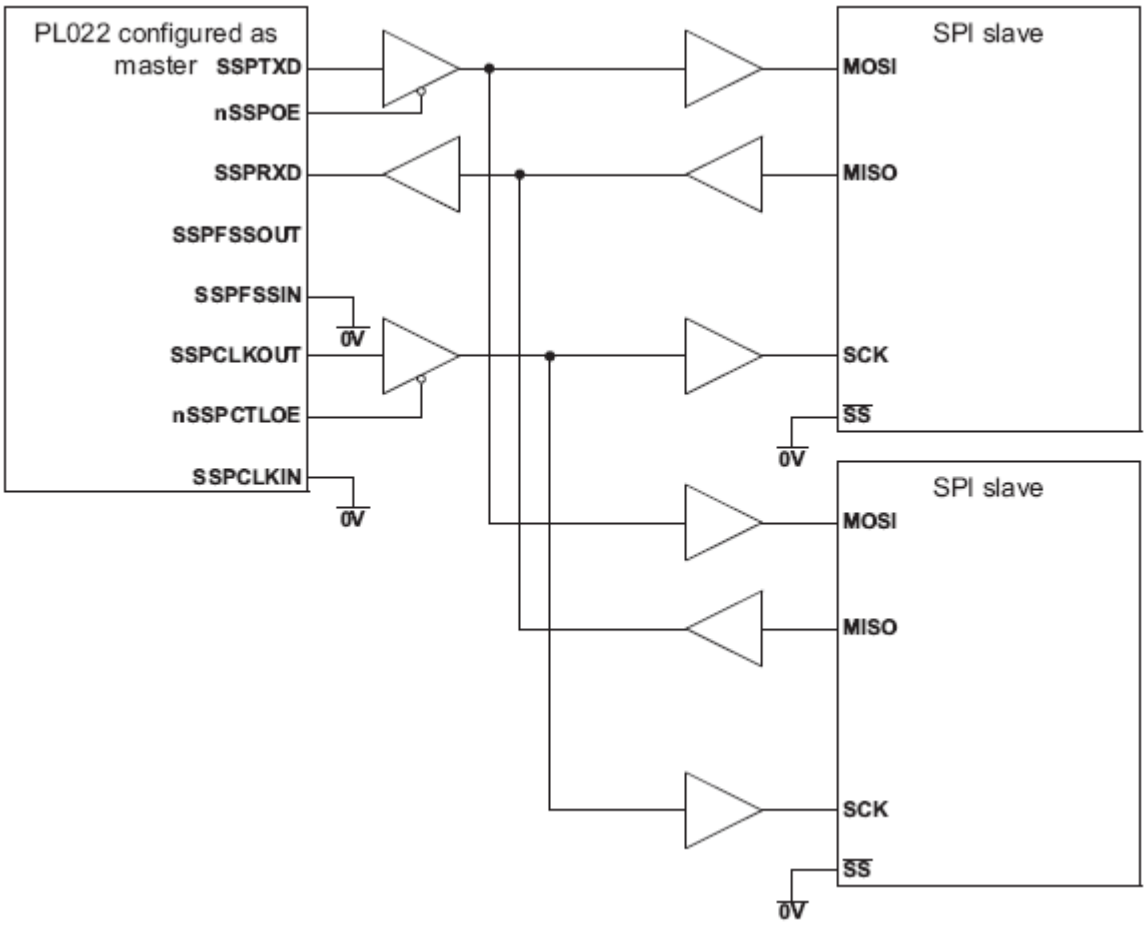


Рисунок 30–14 – Ведущее устройство SSP подключено к двум ведомым, поддерживающим SPI

Рисунок 30–14 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 линю SSP_RXD ведущего устройства одновременно осуществляется только одним из ведомых по соответствующей линии MISO.

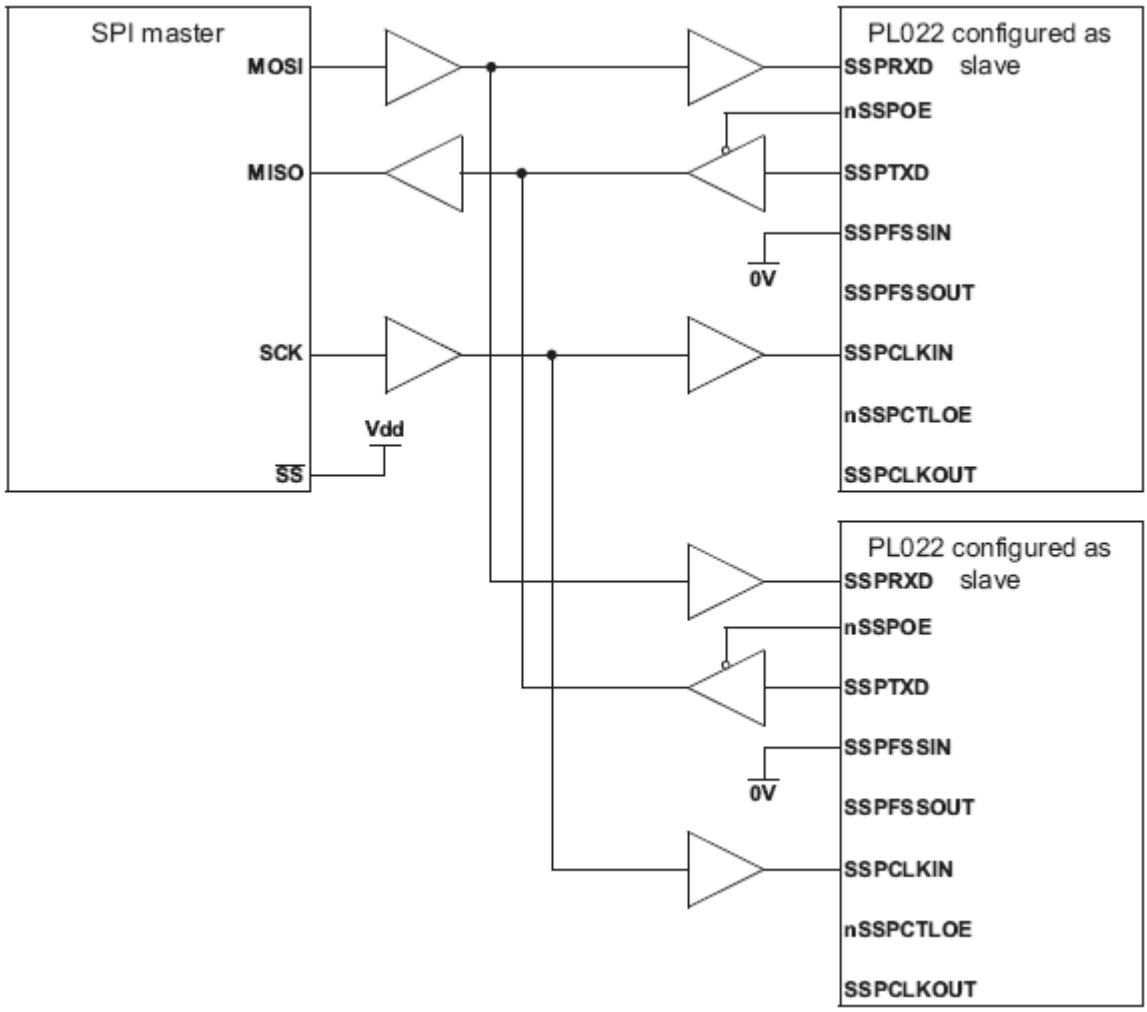


Рисунок 30–15 – Ведущее устройство, протокол SPI, подключено к двум ведомым модулям SSP

Рисунок 30–15 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.

30.6.22 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 Таблица 30-1 приведены значения порогов заполнения буферов приемника и передатчика, необходимых для срабатывания запросов блочного обмена DMABREQ.

Таблица 30-1 – 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)
½	4	4

Рисунок 30–16 показывает временные диаграммы одноэлементного и блочного запросов DMA, в том числе действие сигнала DMACLR. Все сигналы должны быть синхронизированы с PCLK.

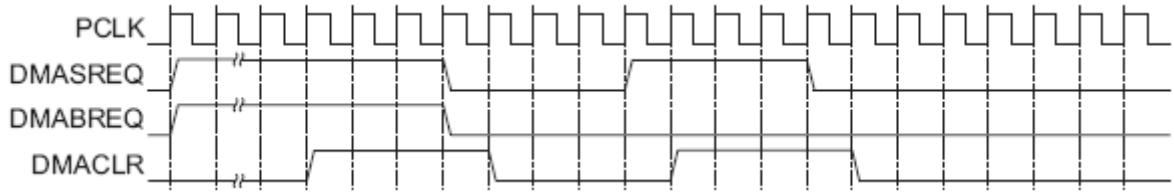


Рисунок 30–16 – Временные диаграммы обмена в режиме DMA

30.7 Программное управление модулем

30.7.1 Общая информация

В микроконтроллере реализовано два модуля SSP, базовые адреса каждого модуля указаны далее в Таблица 30-2. 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.

30.7.2 Описание регистров контроллера SSP

Данные о регистрах модуля SSP приведены ниже (Таблица 30-2).

Таблица 30-2 – Обобщенные данные о регистрах модуля SSP

Базовый адрес	Наимено-вание	Тип	Значение после сброса	Раз-мер, бит	Описание
0x4004_0000	MDR_SSP1				Контроллер SSP1
0x400A_0000	MDR_SSP2				Контроллер SSP2
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
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.

30.7.3 MDR_SSPx->CR0

Регистр управления 0

Регистр CR0 содержит пять битовых полей, предназначенных для управления блоками модуля SSP. Назначение разрядов регистра представлены ниже (Таблица 30-3).

Таблица 30-3 – Формат регистра CR0

№ 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 * (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. раздел «Формат SPI фирмы Motorola»
6	SPO	Полярность сигнала SSPCLKOUT (используется только в режиме обмена SPI фирмы Motorola). См. раздел «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 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

30.7.4 MDR_SSPx->CR1

Регистр управления 1

Регистр CR1 содержит четыре битовых поля, предназначенных для управления блоками модуля SSP. Назначение разрядов регистра представлено в Таблица 30-4.

Таблица 30-4 – Регистр CR1

№ 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 данных 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

30.7.5 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.

Назначение разрядов регистра SSPDR описано в таблице, смледующей ниже (Таблица 30-5).

Таблица 30-5 – Формат регистра DR

№ 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

30.7.6 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.

Назначение бит регистра SSPCPSR представлено в таблице ниже (Таблица 30-6).

Таблица 30-6 – Регистр SR

№ 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

30.7.7 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 30-7 отображает назначение бит регистра SSPSR.

Таблица 30-7 – Регистр CPSR

№ 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

30.7.8 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.

Назначение бит регистра IMSC показано в таблице ниже.

Таблица 30-8 – Регистр IMSC

№ 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 попуток его чтения в течение времени таймаута). 1 – не маскирована. 0 – маскирована
0	RORIM	Маска прерывания по переполнению буфера приемника. 1 – не маскирована. 0 – маскирована

30.7.9 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 30-9 отображает назначение бит в регистре RIS.

Таблица 30-9 – Регистр RIS

№ 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

30.7.10 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.

Назначение бит в регистре SSPMIS представлено в таблице ниже (Таблица 30-10).

Таблица 30-10 – Регистр MIS

№ 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

30.7.11 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.

Назначение бит в регистре SSPICR представлено в таблице ниже (Таблица 30-11).

Таблица 30-11 – Регистр ICR

№ 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

30.7.12 MDR_SSPx->DMACR

Регистр управления прямым доступом к памяти

Регистр доступен по чтению и записи. После сброса все биты регистра обнуляются.

Назначение бит регистра UARTDMACR представлено в таблице ниже (Таблица 30-12).

Таблица 30-12 – Регистр DMACR

№ 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

30.8 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.

30.8.1 SSPRXINTR

Прерывание по заполнению буфера FIFO приемника формируется в случае, если буфер приемника содержит четыре или более несчитанных слов данных.

30.8.2 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.

30.8.3 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.

30.8.4 SSPRTINTR

Прерывание по таймауту приемника возникает в случае, если буфер FIFO приемника не пуст, и на вход приемника не поступало новых данных в течение периода времени, необходимого для передачи 32 бит. 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.

30.8.5 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.

31 Контроллер MDR_UART

Модуль универсального асинхронного приемопередатчика (UART – Universal Synchronous Asynchronous Receiver Transmitter) представляет собой периферийное устройство микроконтроллера.

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).

31.1 Основные сведения

Основные сведения о модуле представлены в следующих разделах:

- основные характеристики;
- программируемые параметры;
- отличия от приемопередатчика 16C650.

31.1.1 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.

Поддержка функция управления модемом (линии CTS, DCD, DSR, RTS, DTR и RI).

Возможность организации аппаратного управления потоком данных.

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.

31.1.2 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.

31.1.3 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 состоянии 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.

31.2 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 индустриального стандарта - контроллера UART 16C650;
- позволяет осуществлять обмен информацией с максимальной скоростью:
  - в режиме UART – до 921600 бит/с;
  - в режиме IrDA – до 460800 бит/с;
  - в режиме IrDA с пониженным энергопотреблением – до 115200 бит/с.

Режим работа приемопередатчика и скорость обмена данными контролируются регистром управления линией UARTLCR_H 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.

Поддерживаются входные линии состояния модема:

- «готовность к приему» (Clear To Send, CTS);
- «обнаружен информационный сигнал» (Data Carrier Detected, DCD);
- «источник данных готов» (Data Set Ready, DSR);
- «индикатор вызова» (Ring Indicator, RI).

Также поддерживаются выходные линии:

- «запрос на передачу» (Request to Send, RTS);
- «приемник данных готов» (Data Terminal Ready, DTR).

Доступна возможность аппаратного управления потоком данных по линиям nUARTCTS и nUARTRTS.

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.

31.3 Описание функционирования блока UART

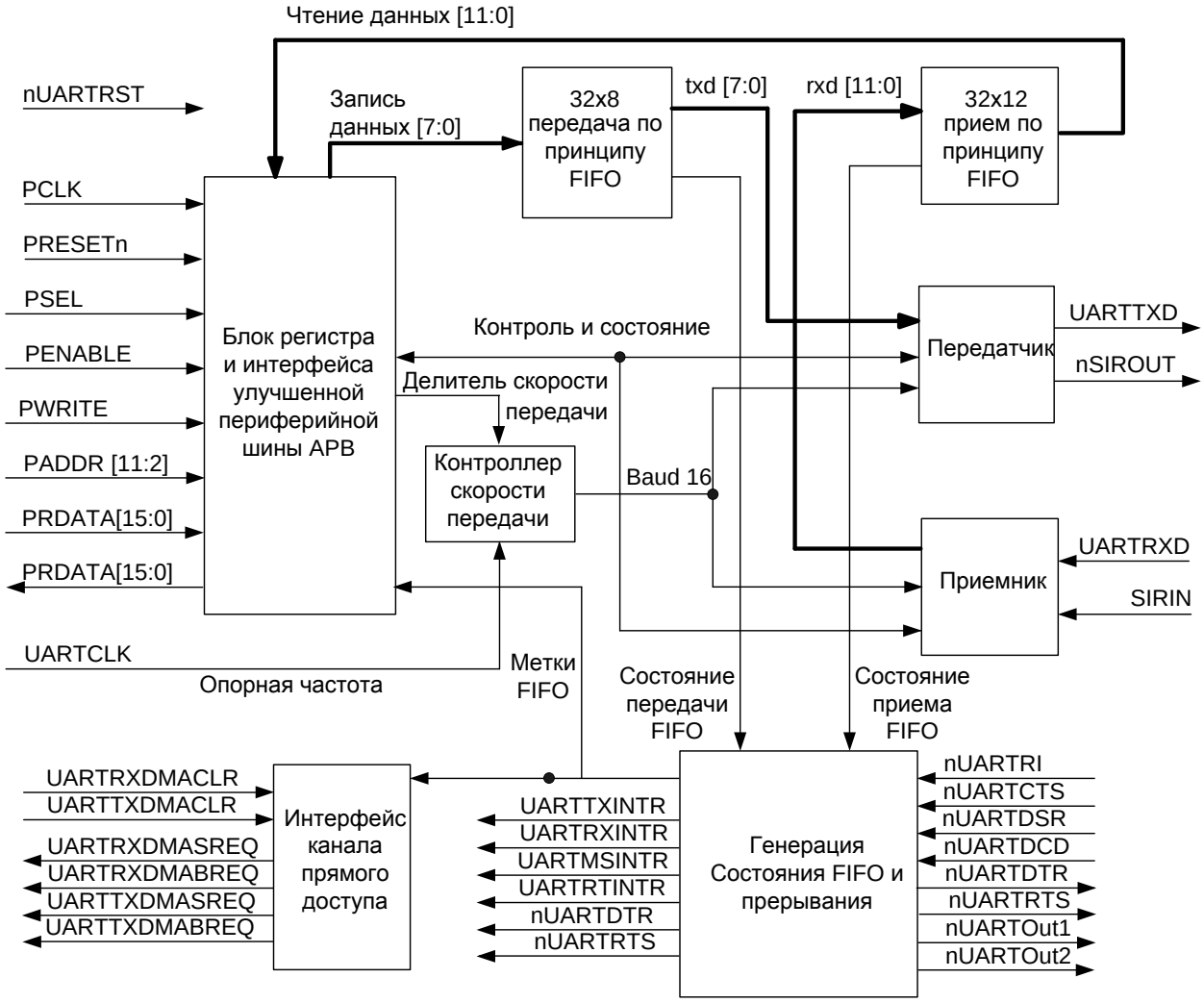


Рисунок 31–1 – Блок-схема универсального асинхронного приёмопередатчика (UART)

31.3.1 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.

31.3.2 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.

31.3.3 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.

31.3.4 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.

31.3.5 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.

31.3.6 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. раздел «Прерывания».

31.3.7 Интерфейс прямого доступа к памяти

Модуль обеспечивает интерфейс с контроллером DMA согласно схеме взаимодействия приемопередатчика и контроллера DMA.

31.3.8 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.

31.4 Описание функционирования ИК кодека IrDA SIR

Структурная схема кодека представлена ниже (Рисунок 31–2).

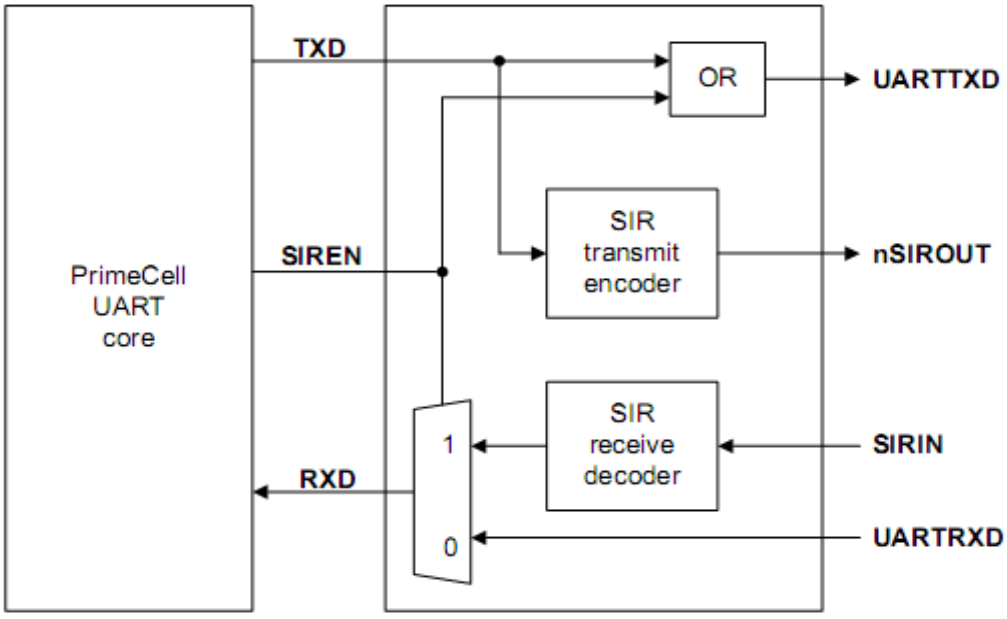


Рисунок 31–2 – Структурная схема кодека IrDA

31.4.1 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, 3 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.

31.4.2 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.

31.5 Описание работы UART

31.5.1 Сброс модуля

Приемопередатчик и кодек могут быть сброшены общим сигналом сброса процессора. Значения регистров после сброса описаны в разделе «Программное управление модулем».

31.5.2 Тактовые сигналы

Частота тактового сигнала UARTCLK должна обеспечивать поддержку требуемого диапазона скоростей передачи данных:

F_UARTCLK(min) >= 16 * baud_rate_max;
F_UARTCLK(max) <= 16 * 65535 * 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.

Также существует ограничение на соотношение между тактовыми частотами CPU_CLK и UARTCLK. Частота UARTCLK должна быть не более, чем в 5/3 раз выше частоты CPU_CLK.

F_UARTCLK <= 5/3 * F_ CPU_CLK.

Например, при работе в режиме UART с максимальной скоростью передачи данных 921600 бод, при частоте UARTCLK 14,7456 МГц, частота CPU_CLK должна быть не менее 8,85276 МГц. Это гарантирует, что контроллер UART будет иметь достаточно времени для записи принятых данных в буфер FIFO.

31.5.3 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.

31.5.4 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:

Коэффициент деления = UARTCLK / (16 * скорость передачи данных)
= BRD_I + BRD_F,

где

BRD_I – целая часть, а BRD_F – дробная часть коэффициента деления.

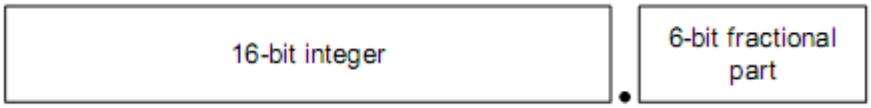


Рисунок 31–3 – Коэффициент деления

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 = integer(BRD_F * 2n + 0.5),

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.

31.5.5 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 31-1 – Назначение бит слова данных в FIFO-буфере приемника).

31.5.6 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 31-1 описывает назначение всех бит слова данных в FIFO-буфере приемника.

Таблица 31-1 – 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

31.5.7 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.

31.5.8 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.

31.5.8.1 Проверка по шлейфу

Проверка по шлейфу (замыкание выхода передатчика на вход приемника) выполняется путем установки в 1 бита LBE в регистре управления контроллером UARTCR.

31.5.9 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 единица 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:

F_UARTCLK / F_IrLPBaud16,

где номинальное значение IrLPBaud16 составляет 1.8432 МГц. Коэффициент деления должен быть выбран так, чтобы выполнялось соотношение:

1,42 МГц < F_IrLPBaud16 < 2,12 МГц.

31.5.9.1 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.

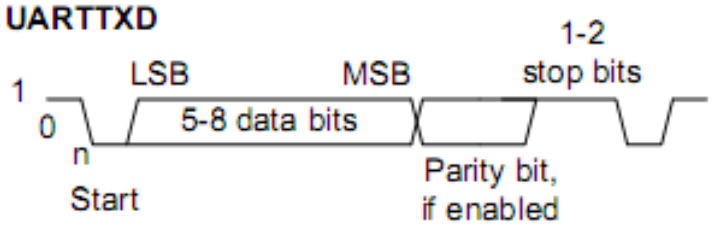


Рисунок 31–4 – Кадр передачи данных

31.5.10 Модуляция данных IrDA

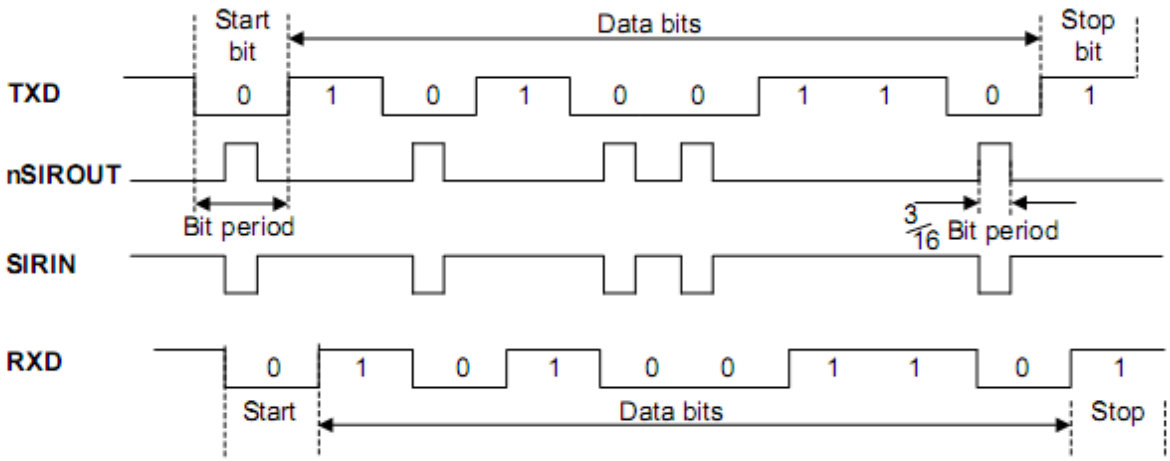


Рисунок 31–5 – Модуляция данных IrDA

31.6 Линии управления модемом

Модуль универсального асинхронного приемопередатчика может использоваться как в режиме оконечного оборудования (DTE), так и в режиме оборудования передачи данных (DCE). Сигналы модема в режиме DTE показаны ранее (см. Рисунок 31–1).

Назначение сигналов в режимах DTE и DCE представлено в таблице ниже.

Таблица 31-2 – Назначение управления модемом в режимах DTE и DCE

Сигнал	Назначение	
	Режим оконечного оборудования	**Режим оборудования передачи данных**
nUARTCTS	Готов к передаче данных	Запрос передачи данных
nUARTDSR	Источник данных готов	Приемник данных готов
nUARTDCD	Обнаружен информационный сигнал	-
nUARTRI	Индикатор вызова	-
nUARTCTS	Запрос передачи данных	Готов к передаче данных
nUARTDTR	Приемник данных готов	Источник данных готов
nUARTOUT1	-	Обнаружен информационный сигнал
nUARTOUT2	-	Индикатор вызова

31.6.1 Аппаратное управление потоком данных

Программно активизируемый режим аппаратного управления потоком данных позволяет контролировать (приостанавливать и возобновлять) информационный обмен с помощью сигналов nUARTRTS и nUARTCTS. Рисунок 31–6 иллюстрирует взаимодействие двух устройств последовательной связи с аппаратным управлением потоком данных.

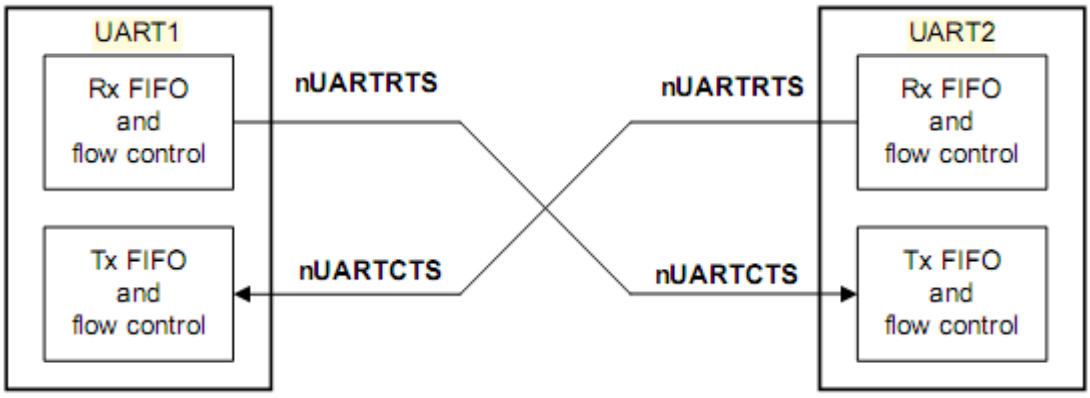


Рисунок 31–6 – Взаимодействие двух устройств последовательной связи с аппаратным управлением потоком данных

Если разрешено управление потоком данных по сигналу RTS, линия nUARTRTS переводится в активное состоянии только после того, как в FIFO буфере приема появляется заданное количество свободных элементов.

Если разрешено управление потоком данных по сигналу CTS, передача данных осуществляется только после перевода линии nUARTCTS в активное состояние.

Режим аппаратного управления потоком данных задается путем установки значений бит RTSEn и CTSEn в регистре управления UARTCR. Таблица 31-3 показывает необходимые установки для различных режимов управления потоком данных.

Таблица 31-3 – Режимы управления потоком данных

CTSEn	RTSEn	Описание
1	1	Разрешено управление потоком данных по CTS и RTS
1	0	Управления потоком данных осуществляется по линии CTS
0	1	Управления потоком данных осуществляется по линии RTS
0	0	Управления потоком данных запрещено

Примечание – В случае если выбран режим управления потоком данных по RTS, программное обеспечение не может использовать бит RTSEn регистра UARTCR для проверки состояния линии RTS.

31.6.2 Управление потоком данных по линии RTS

Логика управления потоком данных по RTS использует данные о превышении пороговых уровней заполнения буфера FIFO приемника. В случае выбора режимов с управлением по RTS, сигнал на линии nUARTRTS переводится в активное состояние только после того, как в FIFO буфере приема появляется заданное количество свободных элементов. После достижения порогового уровня заполнения буфера приемника сигнал nUARTRTS снимается (переводится в пассивное состояние), указывая, таким образом, на отсутствие свободного места для сохранения принятых данных. При этом дальнейшая передача данных должна быть прекращена по завершении передачи текущего символа.

Обратно в активное состояние сигнал nUARTRTS переводится после считывания данных из приемного буфера FIFO в количестве, достаточном для того, чтобы заполнение буфера оказалось ниже порогового уровня.

В случае, если управление потоком данных по RTS запрещено, однако работа приемопередатчика UART разрешена, прием будет осуществляться до полного заполнения буфера FIFO, либо до завершения передачи данных.

31.6.3 Управление потоком данных по линии CTS

В случае выбора одного из режимов с управлением потоком данных по CTS передатчик осуществляет проверку состояния линии nUARTCTS перед началом передачи очередного байта данных. Передача осуществляется только в случае, если данная линия активна, и продолжается до тех пор, пока активное состояние линии сохраняется и буфер передатчика не пуст.

При переходе линии nUARTCTS в неактивное состояние модуль завершает выдачу текущего передаваемого символа, после чего передача данных прекращается.

Если управление потоком данных по CTS запрещено, и при этом работа приемопередатчика UART разрешена – данные будут выдаваться до опустошения буфера FIFO передатчика.

31.7 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 31-4 показывает значения параметров срабатывания запросов блочного обмена UARTRXDMABREQ и UARTTXDMABREQ в зависимости от порога заполнения буфера.

Таблица 31-4 – 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 31–7). Все сигналы должны быть синхронизированы с CPU_CLK. В интересах ясности изложения предполагается, что синхронизация сигналов запроса DMA в контроллере DMA не производится.

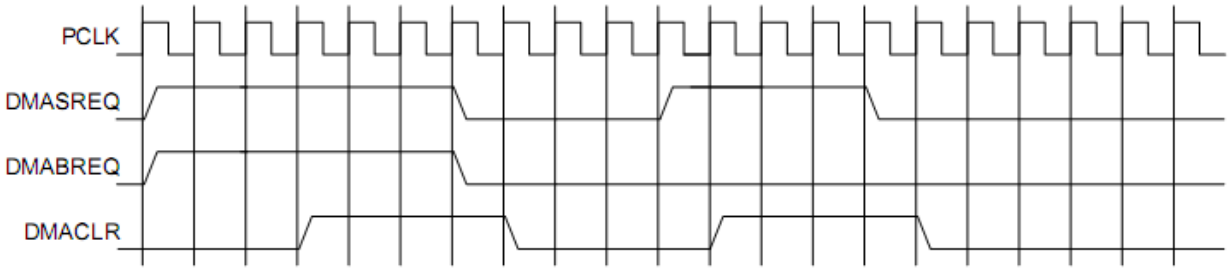


Рисунок 31–7 – Временные диаграммы одноэлементного и блочного запросов DMA

31.8 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;
- UARTTXINTR – прерывание от передатчика;
- UARTRTINTR – прерывание по таймауту приемника;
- UARTMSINTR – прерывание по состоянию модема:
  - UARTRIINTR, изменение состояния линии nUARTRI;
  - UARTCTSINTR, изменение состояния линии nUARTCTS;
  - UARTDCDINTR, изменение состояния линии nUARTDCD;
  - UARTDSRINTR, изменение состояния линии nUARTDSR.
- UARTEINTR – ошибка:
  - UARTOEINTR, переполнение буфера;
  - UARTBEINTR, прерывание приема – разрыв линии;
  - UARTPEINTR, ошибка контроля четности;
  - UARTFEINTR, ошибка в структуре кадра.
- UARTINTR – логическое ИЛИ сигналов UARTRXINTR, UARTTXINTR, UARTRTINTR, UARTMSINTR 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 UARTTXINTR выведены отдельно от прерываний по изменению состояния устройства. Это позволяет использовать сигналы запроса UARTRXINTR и UARTTXINTR 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 UARTMIS.

31.8.1 UARTMSINTR

Прерывание по состоянию модема возникает в случае изменения любой из линий состояний модема (nUARTCTS, nUARTDCD, nUARTDSR, nUARTRI). Сброс прерывания осуществляется путем записи 1 в соответствующий (в зависимости от линии состояния модема, вызвавшей прерывание) разряд регистра сброса прерывания UARTICR.

31.8.2 UARTRXINTR

Состояние прерывания от приемника может измениться в случае возникновения одного из следующих событий:

- буфер FIFO разрешон 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.

31.8.3 UARTTXINTR

Состояние прерывания от передатчика может измениться в случае возникновения одного из следующих событий:

- буфер FIFO разрешон 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.

31.8.4 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.

31.8.5 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 UARTMIS.

Сброс прерывания осуществляется путем записи соответствующих бит в регистр сброса прерывания UARTICR. За прерываниями по обнаружению ошибки закреплены биты с 7 по 10.

31.8.6 UARTINTR

Все описанные сигналы запроса на прерывание скомбинированы в общую линию путем объединения по схеме ИЛИ сигналов UARTRXINTR, UARTTXINTR, UARTRTINTR, UARTMSINTR и UARTEINTR с учетом маскирования. Общий выход может быть подключен к системному контроллеру прерывания, что позволит ввести дополнительное маскирование запросов на уровне периферийных устройств.

31.9 Программное управление модулем

31.9.1 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 31-5) определяет режим доступа к регистру в соответствии с обозначениями:
  - RW – чтение и запись;
  - RO – только чтение;
  - WO – только запись.

31.9.2 Обобщенные данные о регистрах устройства

Данные о регистрах модуля универсального асинхронного приемопередатчика приведены в таблице ниже (Таблица 31-5).

Таблица 31-5 – Обобщенные данные о регистрах устройства

Смещение	Наименование	Тип	Значение после сброса	Размер, бит	Описание
0x40030000	MDR_UART1				Контроллер UART1
0x40038000	MDR_UART2				Контроллер UART2
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					Зарезервироавано
0x018	FR	RO	0b-10010---	9	**MDR_UARTx->FR** Регистр флагов
0x01C					Зарезервироавано
0x020	ILPR	RW	0x00	8	**MDR_UARTx->ILPR** Регистр управления ИК обменом в

					режиме пониженного энергопотребления
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

31.9.3 MDR_UARTx->DR

Регистр данных

В ходе передачиданных:

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 считывается путем чтения из регистра UARTDR принятых данных совместно с соответствующими битами состояния. Информация о состоянии также может быть получена путем чтения регистра UARTRSR/UARTECR (Таблица 31-6).

Таблица 31-6 – Формат регистра UARTDR

№ бита	Сигнал	Назначение
15…12		Резерв
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.
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.

31.9.4 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 31-7 показывает назначение бит регистра UARTRSR/UARTECR.

Таблица 31-7 – Регистр UARTRSR/UARTECR

№ 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
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

31.9.5 MDR_UARTx->FR

Регистр флагов

После сброса биты регистра флагов TXFF, RXFF и BUSY устанавливаются в 0, а биты TXFE и RXFE – в 1. В таблице ниже представлена информация о назначении бит регистра UARTFR.

Таблица 31-8 – Регистр UARTFR

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15…9		Резерв. Не модифицируйте. При чтении заполняются нулями
8	RI	Инверсия линии nUARTRI
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
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	DCD	Инверсия линии nUARTDCD
1	DSR	Инверсия линии nUARTDSR
0	CTS	Инверсия линии nUARTCT

31.9.6 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. Назначение разрядов регистра показано в Таблица 31-9.

Требуемое значение коэффициента деления для формирования сигнала IrLPBaud16 вычисляется по формуле:

ILPDVSR = F_UARTCLK / F_IrLPBaud16,

где номинальное значение частоты F_IrLPBaud16 составляет 1,8432 МГц.

Коэффициент деления должен быть установлен таким образом, чтобы выполнялось соотношение:

1,42 МГц < F_IrLPBaud16 < 2,12 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 31-9 – Регистр UARTILPR

№ 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.

31.9.7 MDR_UARTx->IBRD

Регистр целой части делителя скорости передачи данных

Назначение бит регистра UARTBIRD показано ниже.

Таблица 31-10 – Регистр UARTBIRD

№ 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

31.9.8 MDR_UARTx->FBRD

Регистр дробной части делителя скорости передачи данных,

Таблица 31-11 показывает назначение бит регистра.

Таблица 31-11 – Регистр UARTBFRD

№ 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:

BAUDDIV = FUARTCLK/(16 * Baud_rate),

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:

Коэффициент деления = (4*106)/(16*230400) = 1,085

Таким образом, BRDI = 1, BRDF = 0,085

Следовательно, значение, записываемое в регистр UARTBFRD, равно

m = integer((0,085*64)+0.5) = 5

Реальное значение коэффициента деления = 1+5/64 = 1,078

Реальная скорость передачи данных = (4*106)/(16*1,078) = 231911 бит/с

Ошибка установки скорости = (231911-230400)/230400 * 100 % = 0,656 %

Максимальная ошибка установки скорости передачи данных с использованием шестиразрядного регистра UARTBFRD = 1/64*100 % = 1,56 %. Такая ошибка возникает в случае m = 1, при этом разница накапливается в течение 64 тактовых интервалов.

В следующей таблице (Таблица 31-12) 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 31-12 – 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 31-13 – 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

31.9.9 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 31-14 показывает назначение разрядов регистра UARTLCR_H.

Таблица 31-14 – Регистр UARTLCR_H

№ 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 31-15 показывает таблицу истинности для бит управления контролем четности PEN, EPS и SPS регистра управления линией UARTLCR_H.

Таблица 31-15 – 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.

31.9.10 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.

Назначение разрядов регистра управления показано в следующей таблице.

Таблица 31-16 – Регистр управления UARTCR

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15	CTSEn	Разрешение управления потоком данных по CTS. 1 – разрешено, данные передаются в линию только при активном значении сигнала nUARTCTS.
14	RTSEn	Разрешение управления потоком данных по RTS. 1 – разрешено. Запрос данных от внешнего устройства осуществляется только при наличии свободного места в буфере FIFO приемника
13	Out2	Инверсия сигнала на линии состояния модема nUARTOut2. В режиме оконечного оборудования (DTE) эта линия может использоваться в качестве линии «сигнал вызова» (RI)
12	Out1	Инверсия сигнала на линии состояния модема nUARTOut1. В режиме оконечного оборудования (DTE) эта линия может использоваться в качестве линии «обнаружен информационный сигнал» (DCD)
11	RTS	Инверсия сигнала на линии состояния модема nUARTRTS
10	DTR	Инверсия сигнала на линии состояния модема nUARTDTR
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 – запрещено. 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 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

Примечание – 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.

Примечание – 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.

31.9.11 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 UARTTXINTR или UARTRXINTR, 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 31-17 – Регистр UARTIFLS

№ 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
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

31.9.12 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.

Назначение бит регистра UARTIMSC показано в Таблица 31-18.

Таблица 31-18 – Регистр UARTIMSC

№ 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
4	RXIM	Маска прерывания от приемника UARTRXINTR. 1 – установлена; 0 – сброшена
3	DSRMIM	Маска прерывания UARTDSRINTR по изменению состояния линии nUARTDSR: 1 – установлена; 0 – сброшена
2	DCDMIM	Маска прерывания UARTDCDINTR по изменению состояния линии nUARTDCD: 1 – установлена; 0 – сброшена
1	CTSMIM	Маска прерывания UARTCTSINTR по изменению состояния линии nUARTCTS: 1 – установлена; 0 – сброшена
0	RIMIM	Маска прерывания UARTRIINTR по изменению состояния линии nUARTRI: 1 – установлена; 0 – сброшена

31.9.13 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.

Назначение бит регистра UARTRIS представлено в следующей таблице.

Таблица 31-19 – Регистр UARTRIS

№ 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
5	TXRIS	Состояние прерывания от передатчика UARTTXINTR
4	RXRIS	Состояние прерывания от приемника UARTRXINTR
3	DSRRMIS	Состояние прерывания UARTDSRINTR по изменению линии nUARTDSR
2	DCDRMIS	Состояние прерывания UARTDCDINTR по изменению линии nUARTDCD
1	CTSRMIS	Состояние прерывания UARTCTSINTR по изменению линии nUARTCTS
0	RIRMIS	Состояние прерывания UARTRIINTR по изменению линии nUARTRI

31.9.14 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.

Назначение бит регистра UARTMIS представлено в таблице ниже.

Таблица 31-20 – Регистр UARTMIS

№ 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	DSRMMIS	Маскированное состояние прерывания UARTDSRINTR по изменению линии nUARTDSR
2	DCDMMIS	Маскированное состояние прерывания UARTDCDINTR по изменению линии nUARTDCD
1	CTSMMIS	Маскированное состояние прерывания UARTCTSINTR по изменению линии nUARTCTS
0	RIMMIS	Маскированное состояние прерывания UARTRIINTR по изменению линии nUARTRI

31.9.15 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.

Назначение бит регистра UARTICR представлено в таблице.

Таблица 31-21 – Регистр UARTICR

№ 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	DSRMIC	Сброс прерывания UARTDSRINTR по изменению линии nUARTDSR
2	DCDMIC	Сброс прерывания UARTDCDINTR по изменению линии nUARTDCD
1	CTSMIC	Сброс прерывания UARTCTSINTR по изменению линии nUARTCTS
0	RIMIC	Сброс прерывания UARTRIINTR по изменению линии nUARTRI

31.9.16 MDR_UARTx->DMACR

Регистр управления прямым доступом к памяти

Регистр доступен по чтению и записи. После сброса все биты регистра обнуляются.

Назначение бит регистра UARTDMACR представлено в таблице.

Таблица 31-22 – Регистр UARTDMACR

№ 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

32 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).

32.1 Описание функционирования контроллера

32.1.1 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 32–1).

При заполнении FIFO буфера команд следует учитывать, что команда передается старшин битом младшего байта вперед. Младший байт должен быть формата 0b01xxxxxx.

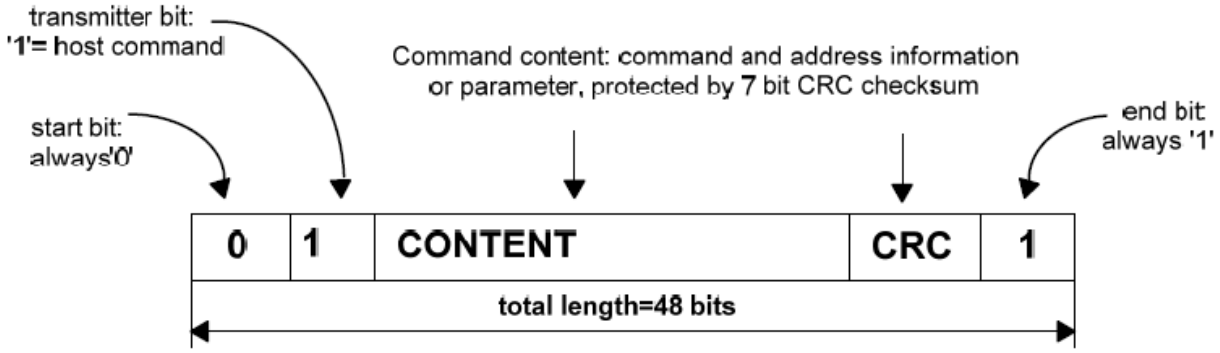


Рисунок 32–1 – Формат передаваемой команды по спецификации SDIO в режиме ведущего

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 */
```

32.1.2 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 32–2).

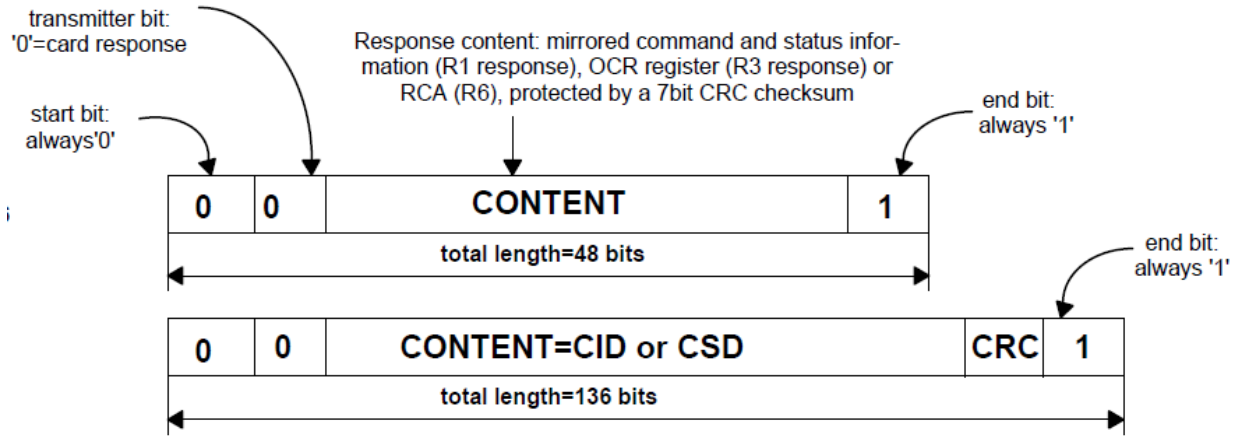


Рисунок 32–2 – Формат принимаемой команды по спецификации SDIO в режиме ведущего

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 */
```

32.1.3 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 на Рисунок 32–3.

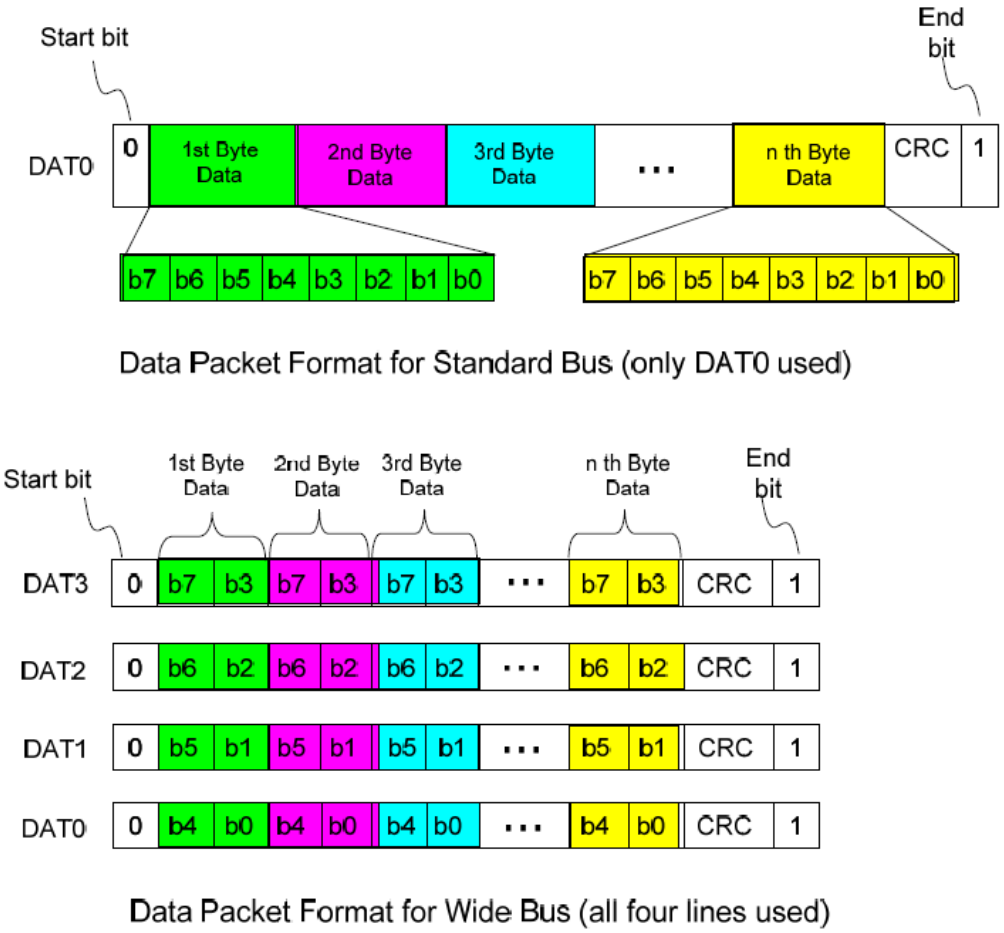


Рисунок 32–3 – Формат данных по спецификации SDIO в режиме ведущего

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».

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 DATActive 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 */
```

32.1.4 Прием данных (по спецификации SDIO)

Для приема данных по интерфейсу SDIO следует настроить режим работы контроллера (регистр SD_CR), обнулить регистры контрольной суммы (SD_DAT0CRC- SD_DAT3CRC), задать количество бит которые надо принять по линиям данных (регистр SD_DATtransfer), а затем выставить бит WORK1 в «1».

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 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».

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 DATActive 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 */
```

32.2 Схемы включения контроллера

32.2.1 Подключение ROM/RAM к контроллеру

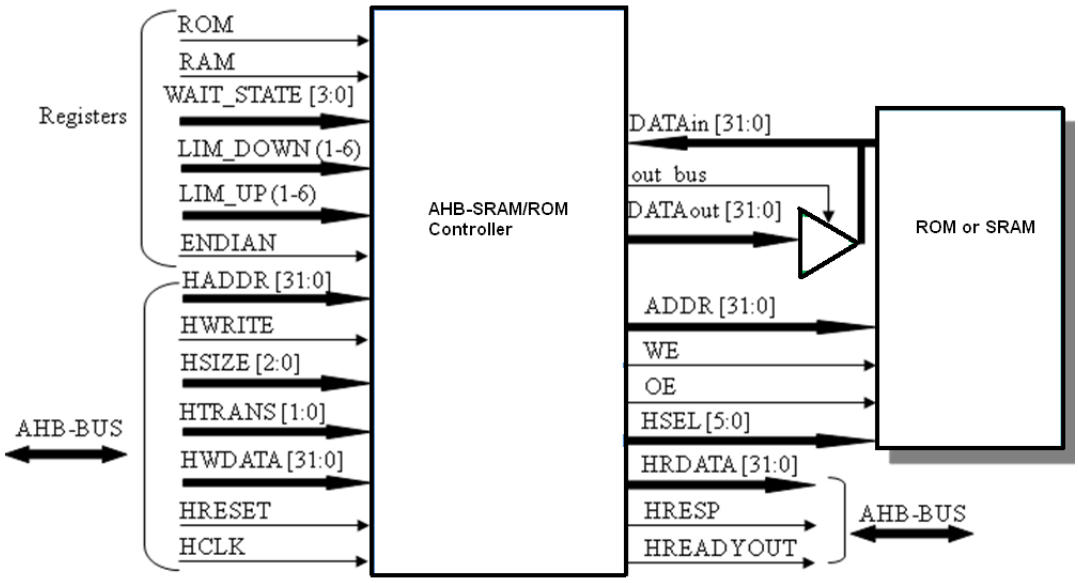


Рисунок 32–4 – Схема подключения ROM/RAM к контроллеру

32.2.2 Подключение SD card к контроллеру

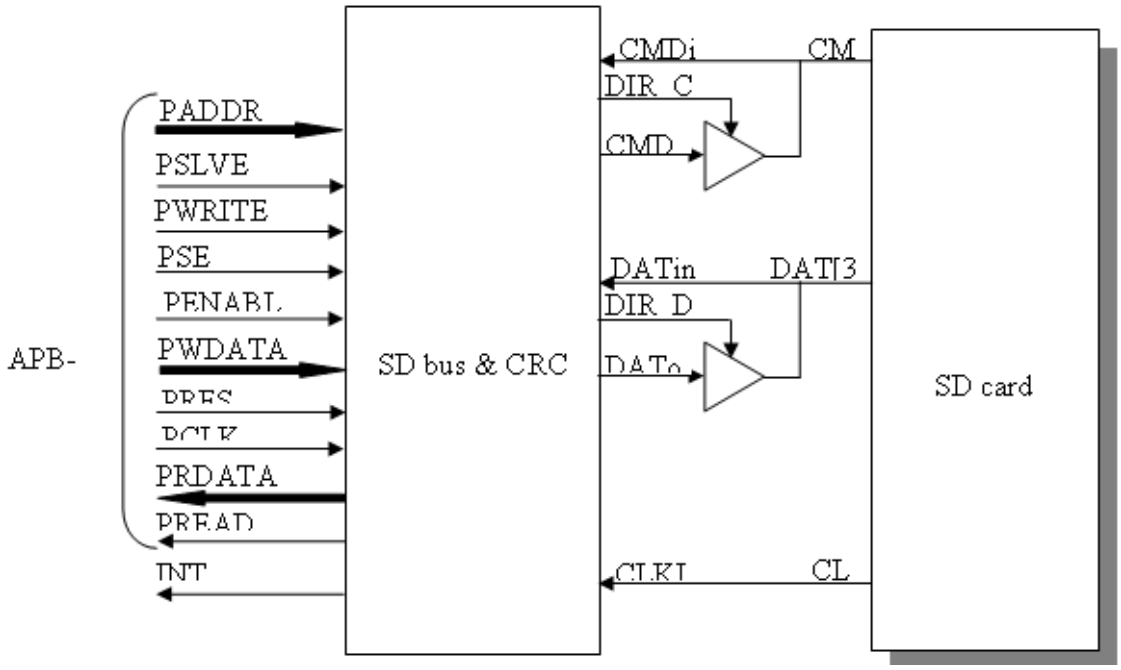


Рисунок 32–5 – Схема подключения хоста к SD-контроллеру

32.2.3 Подключение NAND_Flash к контроллеру

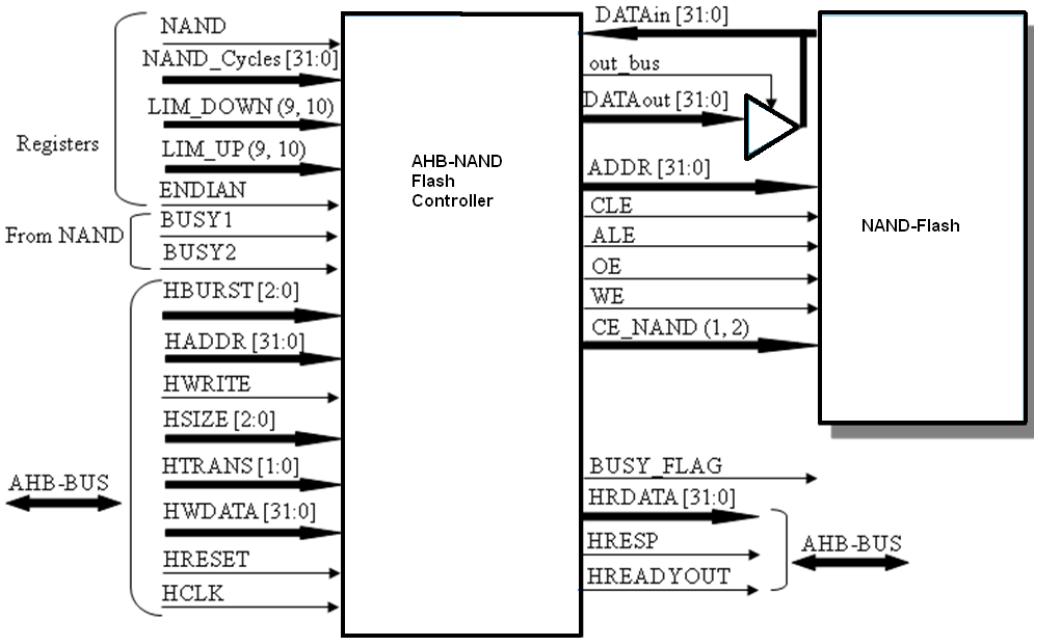


Рисунок 32–6 – Схема подключения NAND флеш к контроллеру

32.3 Регистры SD

32.3.1 Регистр управления

SD_CR=0x0000;

Таблица 32-1

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	23	22	21	20	19	18	17	16
Доступ*	U	U	U	U	U	U	U	R/W
Сброс								0
	-			**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 32-2

№	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31..18		Зарезервировано
19	CLKOE	Сигнал разрешения выдачи сигнала CLK 0 – ведущий 1 – 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 счётчика передаваемых битов команды ненулевое значение. Если бит установлен, но FIFO для передачи команды пусто или FIFO приёма команды полно, то транзакция не начнётся до тех пор, пока FIFO не будут подготовлено для приёма 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** 1 – посылать ответ 0 – не посылать ответ

17	WIDTHDAT	**Разрядность шины данных** 1 – шина одноразрядная 0 –-шина четырёхразрядная
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.
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 определённого 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 счётчика передаваемых битов данных ненулевое значение. Если бит установлен, но FIFO для передачи данных пусто или FIFO приёма данных полно, то транзакция не начнётся до тех пор, пока FIFO не будет подготовлено для приёма 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

32.3.2 Регистр состояния

SD_SR=0x0004;

Таблица 32-3

Номер	7	6	5	4	3	2	1	0
Доступ*	U	U	U	U	R	R	R	R
Сброс					0	0	1	1
					FIFODAT_FULL	**FIFOCMD_FULL**	**FIFODAT_EMPTY**	**FIFOCMD_EMPTY**

Таблица 32-4

№	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31..4		Зарезервировано
0	FIFOCMD_EMPTY	**Состояние FIFO приёма/передачи команд** 0 – FIFO не пусто 1 – FIFO пусто
1	FIFODAT_EMPTY	**Состояние FIFO приёма/передачи данных.** 0 – FIFO не пусто 1 – FIFO пусто
2	FIFOCMD_FULL	**Состояние FIFO приёма/передачи команд.** 0 – FIFO не полно 1 – FIFO полно
3	FIFODAT_FULL	**Состояние FIFO приёма/передачи данных.** 0 – FIFO не полно 1 – FIFO полно

32.3.3 Регистр команд

SD_CMDDR=0x0008;

Таблица 32-5

Номер	31							0
Доступ*	R/W							R/W
Сброс	0							0
	RX_FIFO/TX_FIFO							

Таблица 32-6

№	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31..0	RX_FIFO/TX_FIFO	8x32 бита FIFO приёма и передачи

32.3.4 Регистр данных

SD_DATDR=0x000C;

Таблица 32-7

Номер	31							0
Доступ*	R/W							R/W
Сброс	0							0
	RX_FIFO/TX_FIFO							

Таблица 32-8

№	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31..0	RX_FIFO/TX_FIFO	128x32 бита FIFO приёма и передачи

32.3.5 Регистр контрольной суммы линии команд

SD_CMDCRC=0x0010;

Таблица 32-9

Номер	6							0
Доступ*	R/W							R/W
Сброс	0							0
	SD_CMDCRC							

Таблица 32-10

№	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31..7		Зарезервировано
6..0	SD_CMDCRC	**CRC регистр линии команд** Если разрешено вычисление контрольной суммы при приёме/передаче битом CRCEN_CMD, то регистр содержит подсчитанное значение CRC последовательно принятой/переданной команды. Перед началом подсчёта регистр необходимо инициализировать нулевым значением. CRC рассчитывается последовательно с использованием полинома CRC7 (x^7+x^3+1). Чтение регистра в момент незавершённой транзакции приведёт к ошибочному результату.

32.3.6 Регистры контрольных сумм линий данных

SD_DAT0CRC=0x0014;
SD_DAT1CRC=0x0018;
SD_DAT2CRC=0x001C;
SD_DAT3CRC=0x0020;

Таблица 32-11

Номер	15							0
Доступ*	R/W							R/W
Сброс	0							0
	SD_DATCRC							

Таблица 32-12

№	**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**	**CRC регистры линий данных** Если разрешено вычисление контрольной суммы при приёме/передаче битом CRC_DATEN, то регистр содержит подсчитанное значение CRC последовательно принятых/переданных данных. Перед началом подсчёта регистр необходимо инициализировать нулевым значением. CRC рассчитывается последовательно с использованием полинома CRC16 ($x^{16}+x^{12}+x^5+1$). Чтение регистра в момент незавершённой транзакции приведёт к ошибочному результату.

32.3.7 Регистр-счётчик принимаемых/ передаваемых бит линии команд

SD_CMDtransfer=0x0024;

Таблица 32-13

Номер	16							0
Доступ*	R/W							R/W
Сброс	0							0
	SD_CMDtransfer							

Таблица 32-14

№	**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 приёма/передачи счётчик уменьшается на единицу при каждом принятом/переданном бите, пока не достигнет нулевого значения. Максимальное количество доступных бит для приёма/передачи 65536. Количество бит не обязательно кратно 32.

32.3.8 Регистр-счётчик принимаемых/ передаваемых бит линий данных

SD_DATtransfer=0x0028;

Таблица 32-15

Номер	16							0
Доступ*	R/W							R/W
Сброс	0							0
	SD_DATtransfer							

Таблица 32-16

№	**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 приёма/передачи счётчик уменьшается на четыре при каждом принятом/переданном полубайте, пока не достигнет нулевого значения. Поэтому значение счётчика должно быть кратно четырём. Максимальное количество доступных бит для приёма/передачи 65536. Количество бит не обязательно кратно 32.

33 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.

33.1 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.

33.1.1 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 thread режиме.

33.1.2 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 – В микроконтроллерах серии 1986BE9x данное прерывание не реализовано.

NMI не может быть:

- замаскировано или предотвращено от активации из другого исключения;
- прерывает любые исключения, кроме RESET.

33.1.3 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.

33.1.4 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.

33.1.5 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.

33.1.6 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.

33.1.7 SVCall

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.

33.1.8 PendSV

Исключение PendSV является прерыванием запросом сервисов системного уровня. В приложениях с использованием ОС PendSV используется для переключения контекстов, когда нет других активных исключений.

33.1.9 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.

33.2 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 33-1 – 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.

33.3 Обработчики исключений

Для обработки исключений используются:

- Процедуры обработки прерываний (Interrupt Service Routines – ISRs) Прерывания с IRQ0 по IRQ31 обрабатываются процедурами ISR.
- **Обработчики ошибок (Fault Handlers)** Обрабатывают исключения Hard fault, memory management fault, usage fault и bus fault.
- **Системные обработчики (System handlers)** Обрабатывают исключения NMI, PendSV, SVCall и SysTick.

33.4 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 33–1). Младший бит всех векторов должен быть равен 1, указывая на то, что обработчик выполняется в Thumb режиме.

Exception number	IRQ number	Offset	Vector
45	29	0x00B4	IRQ29
.		.	.
.		.	.
.		.	.
		0x004C	
18	2	0x0048	IRQ2
17	1	0x0044	IRQ1
16	0	0x0040	IRQ0
15	-1	0x003C	Systick
14	-2	0x0038	PendSV
13			Reserved
12			Reserved for Debug
11	-5	0x002C	SVCall
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 33–1 – 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 0x3fffff80. Подробнее в описании регистра VTOR.

33.5 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.

33.5.1 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.

33.6 Вход в обработчик и выход из обработчика

33.6.1 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 разделе «Группировка приоритетов прерываний». В случае, если внутри обработчика исключения возникает прерывание более высокого приоритета, возникает ситуация, называемая вложенным исключением. Подробнее данный вопрос рассмотрен в разделе «Вход в процедуру обработки исключения».

33.6.2 Возврат

Возврат из программы-обработчика осуществляется по завершении обработки исключительной ситуации, с одновременным выполнением следующих условий:

- в системе отсутствуют необработанные исключения с достаточным приоритетом;
- завершенный 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 разделе «Возврат из обработчика исключения».

33.6.3 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.

33.6.4 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.

33.6.5 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 разделе «Запоздавшее исключение (late-arriving exception)».

33.6.6 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 качества 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 33-1).

Процессор устанавливает биты EXC_RETURN [31:4] в 0xFFFFFFF. Загрузка данного значения в PC указывает на завершение процедуры обработки исключения и заставляет процессор выполнить необходимые действия для возврата из обработчика.

Таблица 33-2 – Возврат из обработчика исключения

EXC_RETURN[3:0]	Описание
bXXX0	Зарезервирован
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

33.7 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.

33.7.1 Типы отказов

В Таблица 33-3 представлены 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 разделе “Конфигурируемый регистр отказов”.

Таблица 33-3 – Отказы

Отказ	Обработ-чик	Наименование бита регистра	Регистр отказа
Ошибка доступа к шине при чтении вектора	Тяжелый отказ	VECTTBL	«Регистр состояния тяжелых 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	Обработ-чик	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, не поддерживаемого процессором.

33.7.2 Эскалация отказов и тяжелые отказы

Всем типам исключительных ситуаций по отказу, за исключением тяжелых отказов (hard fault) можно задать приоритет обработки, см. «SCB->SHP[x]». Выполнение данных обработчиков можно программно запретить, см. «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 тяжелый. 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 тяжелых 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 тяжелого отказа.

33.7.3 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 Таблица 33-4.

Таблица 33-4 – Регистры состояния и адреса отказа

Обработчик	Регистр состояния	Регистр адреса	Описание регистров
Тяжелый отказ	HFSR	-	“Регистр состояния тяжелых 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”

33.7.4 Блокировка

Процессор переходит в состояние блокировки в случае, если тяжелый отказ возник во время выполнения программы-обработчика тяжелого отказа.

После перехода в состояние блокировки процессор перестает выполнять какие-либо команды. В этом состоянии он будет находиться до момента сброса.

33.8 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.

33.8.1 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.

33.8.2 Ожидание прерывания

Инструкция ожидания прерывания WFI (wait for interrupt) после своего выполнения немедленно переводит процессор в режим пониженного энергопотребления.

33.8.3 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.

33.8.4 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.

33.8.5 Выход из состояния ожидания

Условия выхода процессора из режима ожидания зависят от причины, по которой он был переведен в этот режим.

33.8.5.1 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.

33.8.5.2 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”.

33.8.6 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 33-5 – 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 Таймера (DSP)
IRQ10	0x0068	I2C	Сигнал INT при EN_INT
IRQ11	0x006C	POWER	Сигнал прерывания от POWER Detecor
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 Крипто модуля (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 BKP 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

34 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 в Таблица 34-1.

Таблица 34-1 – 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]				
…					

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.

34.1 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;
- 4-битные поля регистра приоритета прерываний отображаются на массив 4-разрядных 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 34-2 показанывает отображение прерываний (номеров запросов IRQ) на регистры прерываний и соответствующие переменные CMSIS, для которых предусмотрено по одному биту на прерывание.

Таблица 34-2 – 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

34.1.1 NVIC->ISER[x]

Регистр ISER0 предназначен для разрешения прерываний (запись) и определения, какие из прерываний разрешены (чтение).

Таблица 34-3 – Регистр разрешения прерываний

Номер	31…0
Доступ	R/W
Сброс	0
	SETENA bits

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.

34.1.2 NVIC->ICER[x]

Регистр запрета прерываний

Регистр ICER0 предназначен для запрета прерываний (запись) и определения, какие из прерываний разрешены (чтение) (Таблица 34-4).

Таблица 34-4 – 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.

34.1.3 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 34-5).

Таблица 34-5 – 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.

34.1.4 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 34-6 – 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.

34.1.5 NVIC->IABR[x]

Регистр активных прерываний

Регистр ICPR0 показывает, какие из прерываний находятся в активном состоянии. Этот регистр доступен только для чтения (Таблица 34-7).

Таблица 34-7 – 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.

34.1.6 NVIC->IP[x]

Регистры приоритета прерываний

Регистры IPR0-IPR7 представляют собой набор 4-битных 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 ниже.

Таблица 34-8 – 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 DIV 4;

- смещение данных в регистре в зависимости от значения N MOD 4 равно:
  0 – биты регистра [7:0];
  1 – биты регистра [15:8];
  2 – биты регистра [23:16];
  3 – биты регистра [31:24].

34.1.7 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 34-9 – Регистр программного формирования прерывания

Номер	31...9	8...0
Доступ	U	R/W
Сброс	0	0
	-	**INTID**

INTID – идентификатор формируемого прерывания в диапазоне 0 – 239.
Например: значение b000000011 соответствует прерыванию IRQ3.

34.2 Прерывания, срабатывающие по уровню сигнала

Процессор способен обрабатывать прерывания, сформированные по уровню сигнала.

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. раздел “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.

34.3 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. NVIC->ISPR[x]) или соответствующего значения в регистр STIR (см. 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.

34.4 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 разделе 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 34-10 – 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.

35 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 35-1).

Таблица 35-1 – Обобщенная информация о регистрах блока управления системой

Адрес	Имя	Тип	Доступ	Значение после сброса	Описание
0xE000E000	InterruptType				
0x008	ACTLR	RW	Привилеги-рованный	0x00000000	Дополнительный регистр управления
0xE000ED00	SCB				Блок управления системой
0x000	CPUID	RO	Привилеги-рованный	0x412FC230	Регистр идентифика ции процессора
0x004	ICSR	RW	Привилеги-рованный	0x00000000	Регистр управления прерываниями
0x008	VTOR	RW	Привилеги-рованный	0x00000000	Регистр смещения таблицы векторов прерываний
0x00C	AIRCR	RW	Привилеги-рованный	0xFA050000	Регистр управления прерываниями и программного сброса
0x010	SCR	RW	Привилеги-рованный	0x00000000	Регистр управления системой
0x014	CCR	RW	Привилеги-рованный	0x00000200	Регистр конфигурации и управления
0x018	SHPR1	RW	Привилеги-рованный	0x00000000	Регистр №1 приоритета системных обработчиков
0x01C	SHPR2	RW	Привилеги-рованный	0x00000000	Регистр №2 приори-тета системных обработчиков
0x020	SHPR3	RW	Привилеги-рованный	0x00000000	Регистр №3 приоритета системных обработчиков
0x024	SHCRS	RW	Привилеги-рованный	0x00000000	Регистр управления и состояния системных обработчиков
0x028	CFSR	RW	Привилеги-рованный	0x00000000	Регистр состояния отказов с конфигурируемым приоритетом
0x028	MMSR	RW	Привилеги-рованный	0x00	Регистр состояния отказов доступа к памяти
0x029	BFSR	RW	Привилеги-рованный	0x00	Регистр состояния отказов доступа к шине
0x02A	UFSR	RW	Привилеги-рованный	0x0000	Регистр состояния отказов, вызванных ошибками

Адрес	Имя	Тип	Доступ	Значение после сброса	Описание
					программирования
0x02C	HFSR	RW	Привилеги-рованный	0x00000000	Регистр состояния тяжелого отказа
0x034	MMAR	RW	Привилеги-рованный	Не определено	Регистр адреса отказа доступа к памяти
0x038	BFAR	RW	Привилеги-рованный	Не определено	Регистр адреса отказа доступа к шине

35.1 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].

35.1.1 InterrupType->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 в Таблица 35-2.

Таблица 35-2 – Дополнительный регистр управления

Номер	31...3	2	1	0
Доступ	U	R/W	R/W	R/W
Сброс	0	0	0	0
	-	**DISFOLD**	**DISDEFWBUF**	**DISMCYCINT**

DISFOLD – установка разряда в 1 запрещает вложение условных инструкций (IT folding) (см. ниже “О вложении условных инструкций”).

DISDEFWBUF – 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 IT-блоке, 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.

35.1.2 SCB->CPUID

Регистр идентификации процессора

Регистр CPUID содержит информацию о модели процессора, версии и варианте его реализации. Подробная информация о регистре представлена в Таблица 35-3.

Таблица 35-3 – 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.

35.1.3 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 35-4 – Регистр управления прерываниями

Номер	31...29	28	27	26	25	24	23	22	21...12	11	10	9	8...0
Доступ	U	R/W	R/W	R/W	R/W	U	R/W	R/W	R/W	R/W	U	U	R/W
Сброс	0	0	0	0	0	0	0	0	0	0	0	0	0
	-	**PENDSVSET**	**PENDSVCLR**	**PENDSTSET**	**PENDSTCLR**	-	**Reserved for Debug**	**ISRPENDING**	**VECTPENDING**	**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 – ожидает.

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.

Запись 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.

Запись 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.

Запись 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.

ISRPENDING (RO) – флаг наличия в системе прерываний (за исключением отказов), ожидающих обслуживания. 0 – ожидающие обслуживания прерывания отсутствуют, 1 – присутствуют.

VECTPENDING (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.

VECTACTIVE (RO) – содержит номер активного исключения. 0 – режим приложения, другое число – номер текущего обслуживаемого исключения. Для получения номера запроса прерывания (IRQ) из значения VECTACTIVE 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.

35.1.4 SCB->VTOR

Регистр смещения таблицы векторов прерываний

Регистр VTOR содержит смещение базового адреса таблицы векторов прерываний относительно адреса 0x00000000.

Таблица 35-5 – Регистр смещения таблицы векторов прерываний

Номер	31	30	29	7	6...0
Доступ	U	U	R/W	R/W	R/W
Сброс	0	0	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.

35.1.5 SCB->AIRCR

Регистр управления прерываниями и программного сброса

Регистр AIRCR 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 35-6 – Регистр управления прерываниями и программного сброса

Номер	31...16	15	14…11	10…8	7...3	2	1	0
Доступ	R/W	R/W	U	R/W	U	R/W	R/W	R/W
Сброс	0	0	0	0	0	0	0	0
	On Read: VECTKEYSTAT, On Write: VECTKEY	**ENDIANESS**	**-**	**PRIGROUP**	**-**	**SYSRESETREQ**	**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. Зависимость этого разбиения от значения PRIGROUP показана в Таблица 35-7.

Таблица 35-7 – 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**
b011	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.

35.1.6 SCB->SCR

Регистр управления системой

Регистр SCR позволяет определить требования к переходу в режим и выходу из режима пониженного энергопотребления.

Таблица 35-8 – 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 энергопотреблении 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.

35.1.7 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 тяжелых 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. NVIC->STIR).

Таблица 35-9 – 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**	**BFHFNMIGN**	-	**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.

BFHFNMIGN разрешает обработчикам с уровнем приоритета -1 и -2 игнорировать отказы доступа к шине, вызванные инструкциями чтения и записи. Бит влияет на функционирование обработчиков тяжелых 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. 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. «Возврат из обработчика исключения»).

35.1.8 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 35-10 – 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 35-11 – 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 35-12 – 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 35-13 – 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.

35.1.9 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 35-14 – 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**	**SVCALLPENDED**	**BUSFAULTPENDED**	**MEMFAULTPENDED**	**USGFAULTPENDED**	**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 тяжелого 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.

35.1.10 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 35-15 – Регистр состояния отказов с конфигурируемым уровнем приоритета

Номер	31...16	15...8	7...0
Доступ	RO	RO	RO
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.

35.1.11 Поле MMFSR

Регистр состояния отказов доступа к памяти

Регистр MMFSR содержит набор флагов, указывающих на различные причины отказа доступа к памяти.

Таблица 35-16 – Регистр состояния отказов доступа к памяти

Номер	7	6	5	4	3	2	1	0
Доступ	RO	U	U	RO	RO	U	RO	RO
Сброс	0	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 тяжелого 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 разрешон 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.

35.1.12 Поле BFSR

Регистр состояния отказов доступа к шине

Регистр BFSR содержит набор флагов, указывающих на различные причины отказа доступа к шине:

Таблица 35-17 – Регистр состояния отказов доступа к шине

Номер	7	6	5	4	3	2	1	0
Доступ	RO	U	U	RO	RO	RO	RO	RO
Сброс	0	0	0	0	0	0	0	0
	BFRVALID			**STKERR**	**UNSTKERR**	**IMPRECISERR**	**PRECISERR**	**IBUSERR**

BFARVALID 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 тяжелого 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.

35.1.13 Поле UFSR

Регистр состояния отказов, вызванных ошибками программирования

Регистр UFSR содержит набор флагов, указывающих на различные причины отказа.

Таблица 35-18 – Регистр состояния отказов, вызванных ошибками программирования

Номер	15...10	9	8	7...4	3	2	1	0
Доступ	U	RO	RO	U	RO	RO	RO	RO
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. “SCB->CCR
Регистр конфигурации и управления”).

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. “SCB->CCR
Регистр конфигурации и управления”). Инструкции LDM, STM, LDRD, и STRD, пытающиеся обратиться по невыровненному адресу, вызывают исключение всегда, вне зависимости от значения бита UNALIGN_TRP.

NOCP попытка обращения к сопроцессору. Процессор не поддерживает инструкции, требующие наличия сопроцессора. 0 – не было, 1 – 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 – не было, 1 – процессор попытался выполнить неверной 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.

35.1.14 SCB->HFSR

Регистр состояния тяжелого отказа

Регистр HFSR содержит сведения о причинах вызова обработчика тяжелого отказа. Особенностью данного регистра является то, что для сброса в 0 его разрядов необходимо записать в них значение 1.

Таблица 35-19 – Регистр состояния тяжелого 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 тяжелого 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 тяжелого отказа. Если бит равен 1, значение счетчика команд PC, 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.

35.1.15 SCB->MMFAR

Регистр адреса отказа доступа к памяти

Регистр MMFAR содержит адрес, при обращении по которому возникла ошибка управления памятью.

Таблица 35-20 – Регистр адреса отказа доступа к памяти

Номер	31...0
Доступ	RO
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. “Регистры состояния и адреса отказа”.

35.1.16 SCB->BFAR

Регистр адреса отказа доступа к шине

Регистр BFAR содержит адрес, при обращении по которому возникла ошибка доступа к шине.

Таблица 35-21 – Регистр адреса отказа доступа к шине

Номер	31...0
Доступ	RO
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, вызвавшем отказ.

Флаги регистра BFSR содержат информацию о причине отказа, а также сообщают, является ли значение BFAR корректным. Подробнее см. “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.

36 Сторожевые таймеры

36.1 Описание регистров блока сторожевых таймеров

Таблица 36-1 – Обобщенная информация о регистрах блока сторожевых таймеров

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_PRL[11:0]	Регистр основания счета сторожевого таймера
0x0C	IWDG_SR[1:0]	Регистр статуса сторожевого таймера

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

36.1.1 IWDG_KR

Таблица 36-2 – Регистр Ключа

Номер	15							0
Доступ*	W							W
Сброс	0							0
	KEY[15:0]							

Таблица 36-3 – Описание бит регистра Ключа

№	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 0000h)** Эти биты должны перезаписываться программно через определённые интервалы ключевым значением AAAAh, в противном случае сторожевой таймер генерирует сброс, если таймер достиг значения нуля. Запись ключевого значения 5555h разрешает доступ по записи к регистрам IWDG_PR и IWDG_RLR. Запись ключевого значения CCCCh разрешает работу сторожевого таймера (за исключением, если сторожевой таймер разрешается аппаратно битами конфигурации).

36.1.2 IWDG_PR

Таблица 36-4 – Регистр делителя частоты сторожевого таймера

Номер	7	6	5	4	3	2	1	0
Доступ*	U	U	U	U	U	R/W	R/W	R/W
Сброс						0	0	0
	-	**-**	**-**	**-**	**-**	**PR2**	**PR1**	**PR0**

Таблица 36-5 – Описание бит регистра делителя частоты сторожевого таймера

№	**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 4 001 – делитель на 8 010 – делитель на 16 011 – делитель на 32 100 – делитель на 64 101 – делитель на 128 110 – делитель на 256 111 – делитель на 256 Чтение и запись этого регистра правомерна только, если бит PVU=0 в регистре IWDG_SR

36.1.3 IWDG_RLR

Таблица 36-6 – Регистр основания счета сторожевого таймера

Номер	11							0
Доступ*	R/W							R/W
Сброс	1							1
	RLR[11:0]							

Таблица 36-7 – Описание бит регистра основания счета сторожевого таймера

№	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31..12		Зарезервировано
11..0	RLR[11:0]	**Значение перезагрузки сторожевого таймера** Значение этих битов по доступу защищено с помощью регистра IWDG_KR. Эти биты записываются программно и определяют значение, загружаемое в сторожевой таймер в момент записи значение AAAAh в регистр IWDG_KR. Сторожевой таймер декрементируется, начиная с этого значения. Период таймаута сторожевого таймера функция от этого значения и делителя частоты. Чтение и запись этого регистра правомерна только, если бит RVU=0 в регистре IWDG_SR.

36.1.4 IWDG_SR

Таблица 36-8 – Регистр статуса сторожевого таймера

Номер	7	6	5	4	3	2	1	0
Доступ*	U	U	U	U	U	U	R	R
Сброс							0	0
	-	**-**	**-**	**-**	**-**	**-**	**RVU**	**PVU**

Таблица 36-9 – Описание бит регистра статуса сторожевого таймера

№	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31..2		Зарезервировано
1	RVU	**Флаг обновления значения сторожевого таймера** Этот бит устанавливается аппаратно и служит признаком того, что обновляется значение сторожевого таймера из регистра перезагрузки. Этот бит сбрасывается, если обновление завершено. Значение регистра перезагрузки может быть обновлено только, если этот бит равен нулю.
0	PVU	**Флаг обновления делителя частоты сторожевого таймера** Этот бит устанавливается аппаратно и служит признаком того, что обновляется значение делителя частоты сторожевого таймера. Этот бит сбрасывается, если обновление завершено. Значение регистра делителя частоты может быть обновлено только, если этот бит равен нулю.

36.1.5 WWDG_CR

Таблица 36-10 – Регистр управления

Номер	7	6	5	4	3	2	1	0
Доступ*	R/S	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	0	1	1	1	1	1	1	1
	WDGA	**T6**	**T5**	**T4**	**T3**	**T2**	**T1**	**T0**

Таблица 36-11 – Описание бит регистра управления

№	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31..18		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
6..0	T[6:0]	**Значение семиразрядного счётчика (от старших разрядов к младшим)** Эти биты содержат значение сторожевого таймера, который декрементируется каждые 4096×2^{WDGTB} циклов частоты PCLK периферийной шины APB

36.1.6 WWDG_CFR

Таблица 36-12 – Регистр конфигурации

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	1	1	1	1	1	1	1
	WDGTB0	**W6**	**W5**	**W4**	**W3**	**W2**	**W1**	**W0**

Номер	15	14	13	12	11	10	9	8
Доступ*	U	U	U	U	U	U	R/S	R/W
Сброс							0	0
	-	**-**	**-**	**-**	**-**	**-**	**EWI**	**WDGTB1**

Таблица 36-13 – Описание бит регистра конфигурации

№	**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 значении 40h. Прерывание запрещается только аппаратным сбросом.
8..7	WGTB[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.

36.1.7 WWDG_SR

Таблица 36-14 – Регистр статуса

Номер	7	6	5	4	3	2	1	0
Доступ*	U	U	U	U	U	U	U	R/C
Сброс								0
	-	**-**	**-**	**-**	**-**	**-**	**-**	**EWIF**

Таблица 36-15 – Описание бит регистра статуса

№	**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.

37 Предельные и предельно-допустимые режимы работы

Таблица 37-1 – 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 батарейного 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	U_{IL}	0	0,8	минус 0,3	–
на выводе: OSC_IN при: BYPASS=1		0	0,2 x U_{CC}	минус 0,3	–
Входное напряжение высокого уровня, В, на выводах: PD, PE (0-10), DN, DP	U_{IH}	2,0	U_{CC}	–	4,0
на выводах: PA, PB, PC, PE (11-15), PF, RESET, WAKEUP		2,0	5,25	–	5,3
на выводе: OSC_IN при: BYPASS=1		0,8 x 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	I_{OL}	минус 6	–	минус 10	–
на выводах: PE 6, 7		минус 3	–	минус 10	–
на выводах: DN, DP		минус 6	–	минус 40	–
Выходной ток высокого уровня, мА, на выводах: PA, PB, PC, PD, PE (1-5, 8-12), PF, STANDBY	I_{OH}	–	6	–	10
на выводах: PE 6, 7		–	3	–	10
на выводах: DN, DP		–	6	–	40
Частота следования импульсов тактовых сигналов RISC процессора, МГц	f_{C_RISC}	–	100	–	–
Частота следования импульсов тактовых сигналов модуля ПЦОС, МГц	f_{C_DSP}	–	100	–	–
Частота следования импульсов тактовых сигналов HSE, МГц при: BYPASS=0	f_{C_HSE}	2	16	–	–
при: BYPASS=1		0	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 LSE, кГц при: BYPASS=0	f_{C_LSE}	32	33	–	–
при: BYPASS=1		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+}	минус 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	–	–
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}	–	–	минус 0,3	U_{CCAU}+ 0,3
Частота следования импульсов тактовых сигналов аудиокодека, кГц	f_{C_AUC}	–	24	–	–
Резистивная нагрузка ЦАП, Ом	R_{LDAC}	600	–	–	–
Резистивная нагрузка BIAS, Ом	R_{LBIAS}	600	–	–	–

Наименование параметра, единица измерения	Буквенное обозначение параметра	Норма параметра			
		Предельно-допустимый режим		Предельный режим	
		не менее	не более	не менее	не более
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	t_{GS}	25	–	–	–
при: T=85 °C		10	–	–	–

* 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 $\pm$ 0,2 В.

** При использовании внутреннего опорного напряжения $U_{ADC1\_REF-} = AGND$, $U_{ADC0\_REF+} = U_{CCA}$.

Примечание – Не допускается одновременное воздействие двух и более предельных режимов.

38 Электрические параметры микросхемы

Таблица 38-1 – Электрические параметры микросхемы

Наименование параметра, единица измерения, режим измерения	Буквенное обозначение параметра	Норма параметра		Температура среды, °С
		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, минус 60
Выходное напряжение высокого уровня, В, на выводах: PA, PB, PC, PD, PE, PF, STANDBY, DN, DP	U_{OH}	2,4	–	25, 85, минус 60
Уровень напряжения питания для срабатывания схемы формирования общего сброса, В	U_{BOR}	1,8	2,1	25, 85, минус 60
Входной ток утечки высокого уровня, мкА, на выводах: PA, PB, PC, PD, PE, PF, DN, DP, RESET, WAKEUP	I_{ILH1}	минус 1	1	25, 85, минус 60
Входной ток утечки высокого уровня, мкА, на выводе: OSC_IN при: BYPASS=1	I_{ILH2}	минус 40	40	25, 85, минус 60
Входной ток утечки низкого уровня, мкА, на выводах: PA, PB, PC, PD, PE, PF, RESET, WAKEUP, DN, DP	I_{ILL1}	минус 1	1	25, 85, минус 60
Входной ток утечки низкого уровня, мкА, на выводе: OSC_IN при: BYPASS=1	I_{ILL2}	минус 1	1	25, 85, минус 60
Статический ток потребления в режиме покоя (регулятор напряжения выключен), мкА	I_{CCS}	–	10	25, минус 60
		–	20	85
Динамический ток потребления, мА, при: f_{C_DSP}= 0 МГц, f_{C_RISC}= 100 МГц	I_{OCC1}	–	180	25, 85, минус 60
Динамический ток потребления, мА, при: f_{C_DSP}= 100 МГц, f_{C_RISC}= 100 МГц	I_{OCC2}	–	300	25, 85, минус 60
Частота работы LSI RC-генератора, кГц	f_{O_LSI}	10	60	25, 85, минус 60
Частота работы HSI RC-генератора, МГц	f_{O_HSI}	6	10	25, 85, минус 60
Выходная частота PLL, МГц максимальная минимальная	f_{O_PLL}	 100 –	 – 6	25, 85, минус 60

Наименование параметра, единица измерения, режим измерения	Буквенное обозначение параметра	Норма параметра		Температура среды, °С
		не менее	не более	
Время выхода из режима sleep, мкс	t_{ON}	–	100	25, 85, минус 60
Параметры АЦП				
Разрядность АЦП	E_{NADC}	12	–	25, 85, минус 60
Дифференциальная нелинейность АЦП, единица младшего разряда	E_{DLADC}	минус 1	2	25, 85, минус 60
Интегральная нелинейность АЦП, единица младшего разряда	E_{ILADC}	минус 3	3	25, 85 минус 60
Ошибка смещения АЦП, единица младшего разряда	E_{OFFADC}	минус 6	6	25, 85, минус 60
Параметры ЦАП				
Разрядность ЦАП	E_{NDAC}	12	–	25, 85, минус 60
Дифференциальная нелинейность ЦАП, единица младшего разряда	E_{DLDAC}	минус 1	2	25, 85, минус 60
Интегральная нелинейность ЦАП, единица младшего разряда	E_{ILDAC}	минус 6	6	25, 85, минус 60
Ошибка смещения ЦАП, мВ	E_{OFFDAC}	минус 40	40	25, 85, минус 60
Минимальное выходное напряжение ЦАП, В при R_L=10 кОм	$U_{O\_DAC\ min}$	–	0,08	25, 85, минус 60
Максимальное выходное напряжение ЦАП, В при R_L= 10 кОм	$U_{O\_DAC\ max}$	$U_{REF(DAC)}$-0,08	–	25, 85, минус 60
Параметры компаратора				
Время включения компаратора, мкс	t_{ON_C}	–	100	25, 85, минус 60
Время задержки переключения компаратора, нс	t_{d_C}	–	400	25, 85, минус 60
Параметры аудиокодека				
Коэффициент нелинейных искажений + шум ЦАП, дБ, при: Синус, частота 1 кГц, U_I*= 0 дБ,	T_{DAC_THD+N}	68	–	25, 85, минус 60

Наименование параметра, единица измерения, режим измерения	Буквенное обозначение параметра	Норма параметра		Температура среды, °С
		не менее	не более	
DAGAIN= минус 1 дБ				
Коэффициент 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	–	25, 85, минус 60
U_I**= минус 6 дБ ADGAIN= 0 дБ, INBG= 6 дБ		78	–	
U_I**= минус 12 дБ ADGAIN= 0 дБ, INBG= 2 дБ		74	–	
U_I**= минус 24 дБ ADGAIN= 0 дБ, INBG= 24 дБ		64	–	
Ошибка коэффициента входного предусилителя АЦП, дБ	$GAIN_{ERRADC}$	минус 0,5	0,5	25, 85, минус 60
Максимальная амплитуда выходного дифференциального сигнала ЦАП на выходах OUTP1 и OUTM1, В, DAGAINT= минус 1 дБ U_I*= 0 дБ	U_{OAUDAC}	1,8	2,1	25, 85, минус 60
Напряжение нуля выходного сигнала ЦАП, на выходах OUTP1 и OUTM1, В, GAINT=0, MUTE1= 1, ODRCT1= 1	$U_{OAUDAC0}$	1,3	1,6	25, 85, минус 60
Выходное напряжение BIAS (на выходе BIAS), В	U_{OBIAS}	1,3	1,6	25, 85, минус 60

* – 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= минус 6 дБ соответствует сигналу с максимальной амплитудой U_{IAUADC}/2 (на входах INP1/INM1 и INP2/INM2).

- Полная шкала сигнала для АЦП U_I= минус 12 дБ соответствует сигналу с максимальной амплитудой U_{IAUADC}/4 (на входах INP1/INM1 и INP2/INM2).

- Полная шкала сигнала для АЦП U_I= минус 24 дБ соответствует сигналу с максимальной амплитудой U_{IAUADC}/16 (на входах INP1/INM1 и INP2/INM2)

39 Справочные данные

Таблица 38-2 – Справочные данные

Наименование параметра, единица измерения, режим измерения	Буквенное обозначение параметра	Норма параметра		Температура среды, °С
		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, минус 60
Выходное напряжение регулятора LDO, В при: U_{CC}= 3,6 В, I_{OL}= 200 мА	U_{O_LDO}	1,62	1,98	
Гистерезис портов ввода/вывода, мВ, на выводах: PA-PF при: ModeRX = 0 ModeRX = 1	$\Delta U_{TH(PA-PF)}$	100 200	400 500	
Длительность фронта переключения выходных сигналов, нс, на выводе: STANDBY, C_l= 30 пФ	$t_{W(STANDBY)}$	–	10	
на выводах: PA – PF при: PowerTX=00, C_l= 50 пФ PowerTX=01, C_l= 50 пФ PowerTX=10, C_l= 50 пФ PowerTX=11, C_l= 50 пФ PowerTX=11, C_l= 30 пФ	$t_{W(PA-PF)}$	– – – – –	10 100 20 10 5	
на выводах: DN, DP при: Full Speed, C_l= 50 пФ Low Speed, C_l= 600 пФ		– –	15 300	
Компаратор				
Опорное напряжение компаратора при: CVRSS = 1 на выводах: PE4_COMP_REF+ и PE5_COMP_REF-	$U_{REFCOMP}$	2,4	Ucc	
Напряжение смещения компаратора, мВ, при: U_{CC} = 3,6 В	U_{IO_C}	минус 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				
Время установления сигнала HSIRDY относительно HSION, мкс	$t_{SU(HSI)}$	–	1	
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	$t_{SU(LSE)}$	–	$4096/f_{C\_LSE}$	

Наименование параметра, единица измерения, режим измерения	Буквенное обозначение параметра	Норма параметра		Температура среды, °С
		не менее	не более	
Время установления сигнала PLLRDY относительно PLLON, мкс	$t_{SU(PLL)}$	–	100	25, 85, минус 60
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}	–	$4 \bullet f_{C\_ADC}$	
Время преобразования АЦП, нс	t_{AO_ADC}	–	$28 \bullet f_{C\_ADC}$	
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-}	минус 50	–	
Ток потребления по питанию АЦП, мА при: U_{CCA}=3,6В, Fadc=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	
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	
USB				
Сопротивление резистора между линиями DN, DP и питанием, кОм при: D-PULLUP=1	R_{DN-UCC}	1	2	
D+PULLUP=1	R_{DP-UCC}	1	2	
Сопротивление резистора между линиями DN, DP и шиной «Общей», 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	

Наименование параметра, единица измерения, режим измерения	**Буквенное обозначение параметра**	**Норма параметра**		**Температура среды, °С**
		не менее	**не более**	
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, минус 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}	минус 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	

40 Типовые зависимости

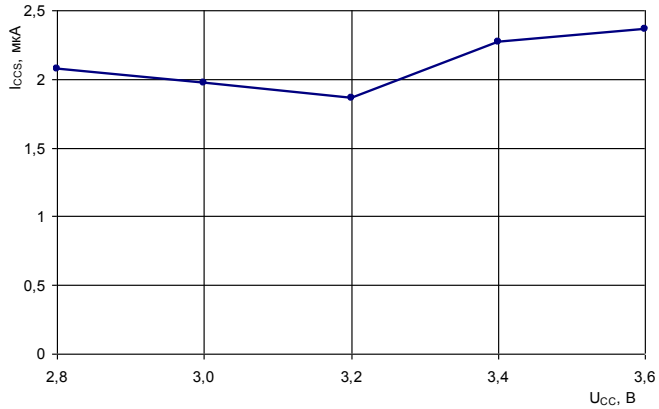


Рисунок 40–1 – Зависимость статического тока потребления в режиме покоя (регулятор напряжения выключен) от напряжения источника питания, при Т=25 ºС

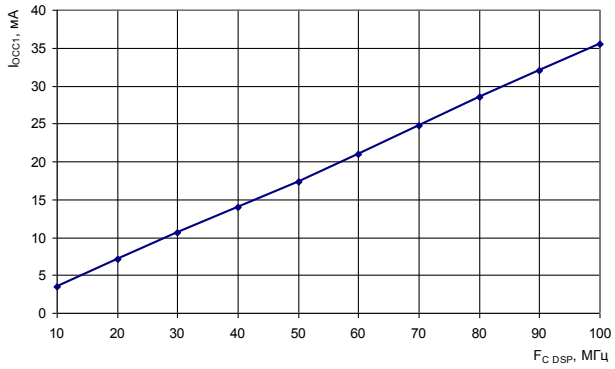


Рисунок 40–2 – Зависимость динамического тока потребления модуля ПЦОС от частоты следования импульсов тактовых сигналов, при Т=25 ºС, UCC =3,6 В, fC RISK =70 МГц

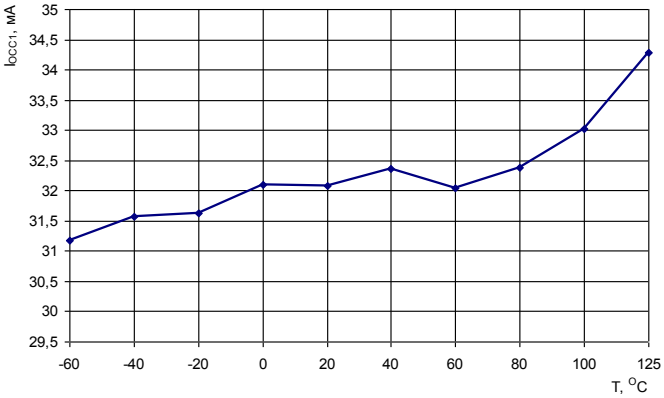


Рисунок 40–3 – Зависимость динамического тока потребления модуля ПЦОС от температуры, при U_{CC} =3,6 В, $f_{C\ DSP}$=100 МГц, $f_{C\ RISK}$=0 МГц

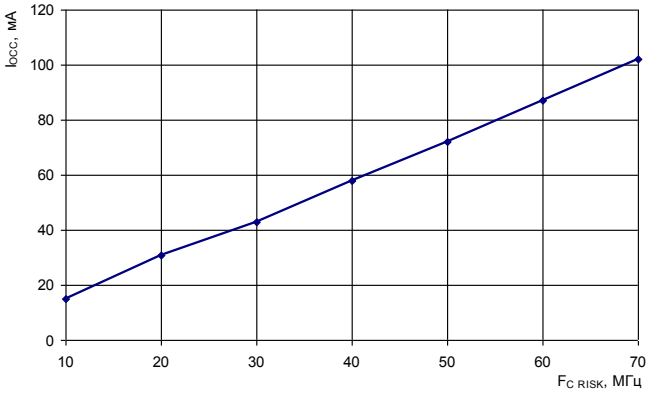


Рисунок 40–4 – Зависимость динамического тока потребления RISC процессора от частоты следования импульсов тактовых сигналов, при Т=25 ºС, U_{CC} =3,6 В, $f_{C\ DSP}$=100 МГц

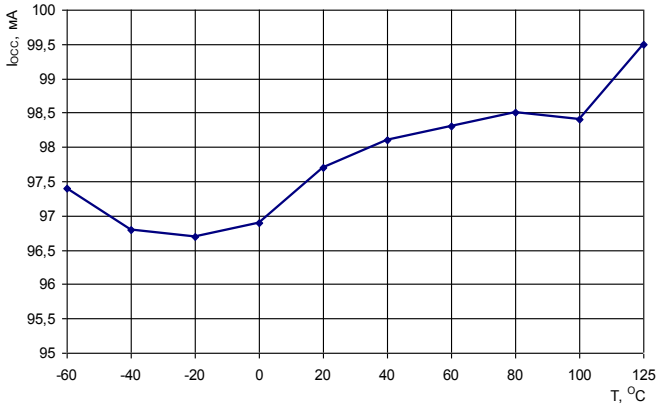


Рисунок 40–5 – Зависимость динамического тока потребления от температуры, при U_{CC} =3,6 В, $f_{C\ DSP}$ =0 МГц, $f_{C\ RISK}$ =70 МГц

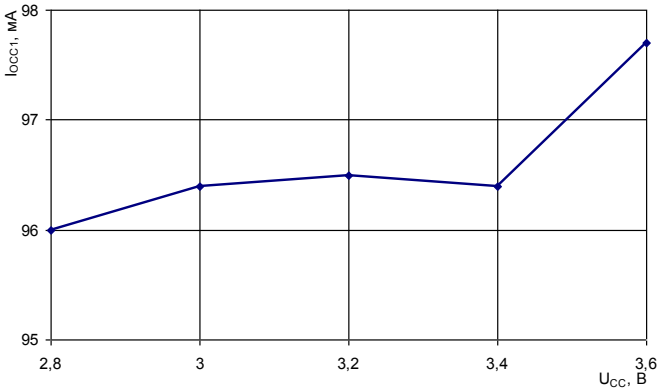


Рисунок 40–6 – Зависимость динамического тока потребления модуля ПЦОС от напряжения источника питания, при Т=25 ºС, $f_{C\ DSP}$ =100 МГц, $f_{C\ RISK}$ =0 МГц

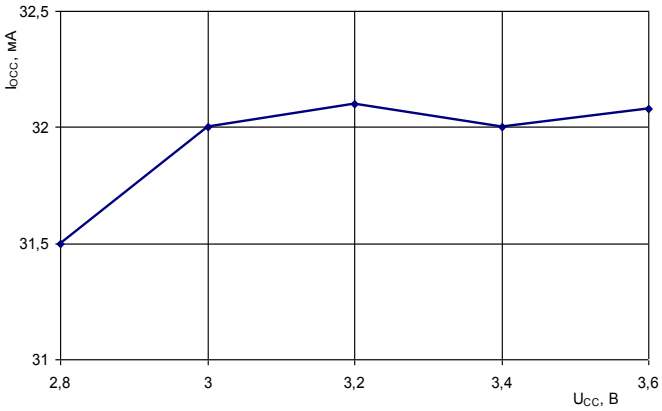


Рисунок 40–7 – Зависимость динамического тока потребления от напряжения источника питания, при T=25 ºС, $f_{C\ DSP}$ =0 МГц, $f_{C\ RISK}$ =70 МГц

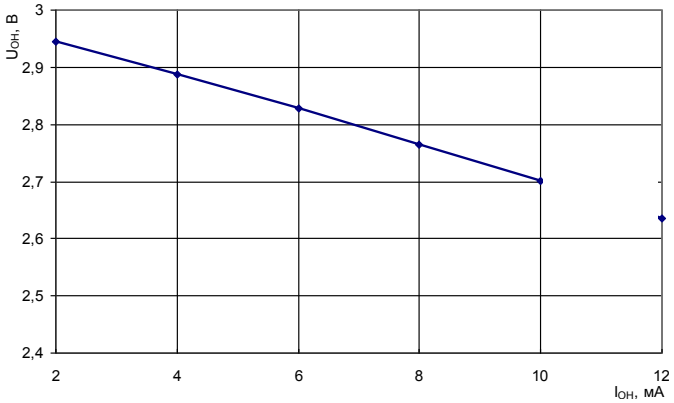


Рисунок 40–8 – Зависимость выходного напряжения высокого уровня от выходного тока нагрузки, при T=25 ºС, U_{CC} =3,0 В

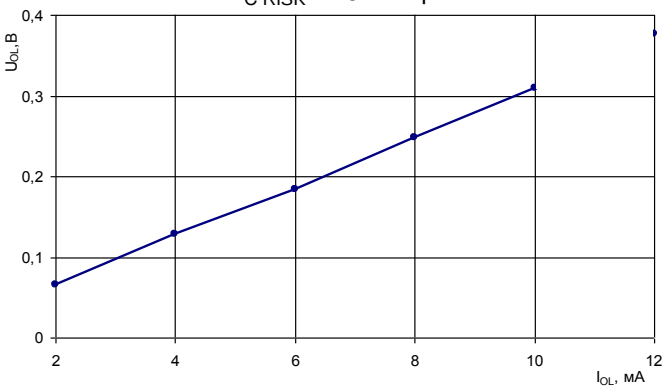


Рисунок 40–9 – Зависимость выходного напряжения низкого уровня от выходного тока нагрузки, при T=25 ºС, U_{CC} =3,0 В

41 Габаритный чертеж микросхемы

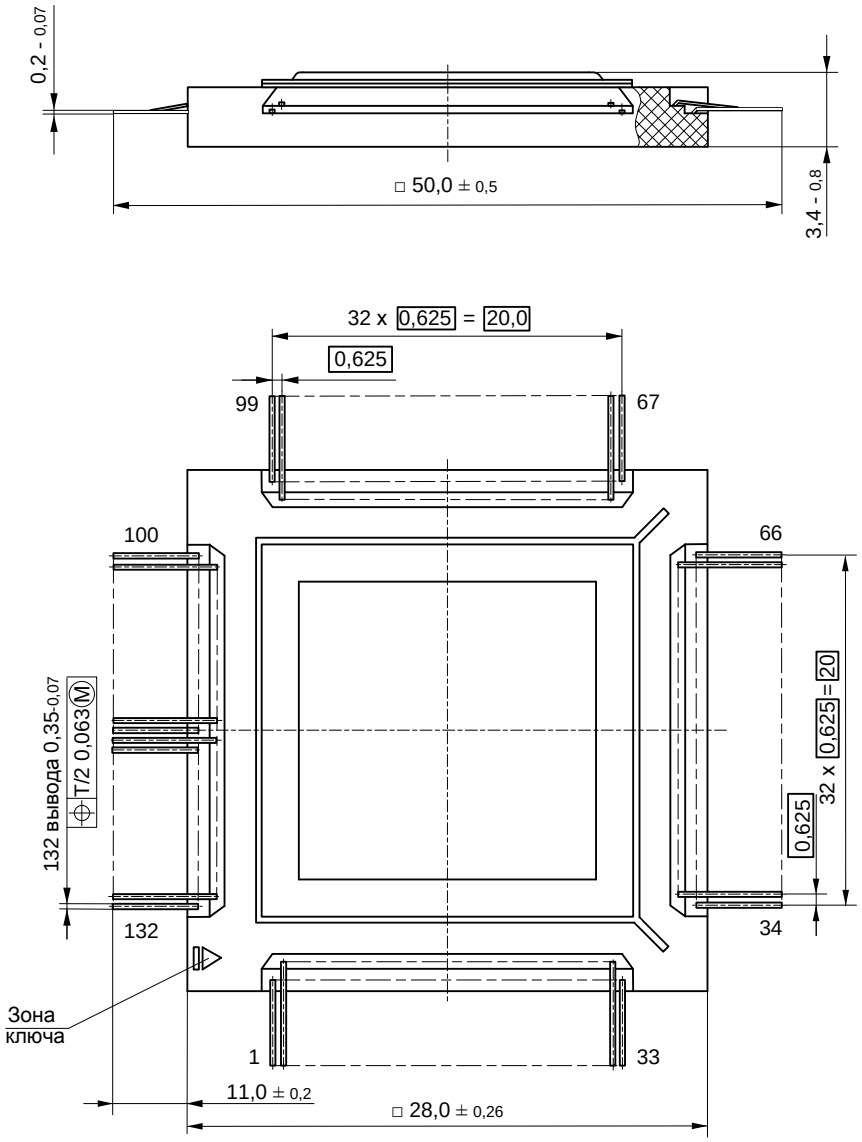
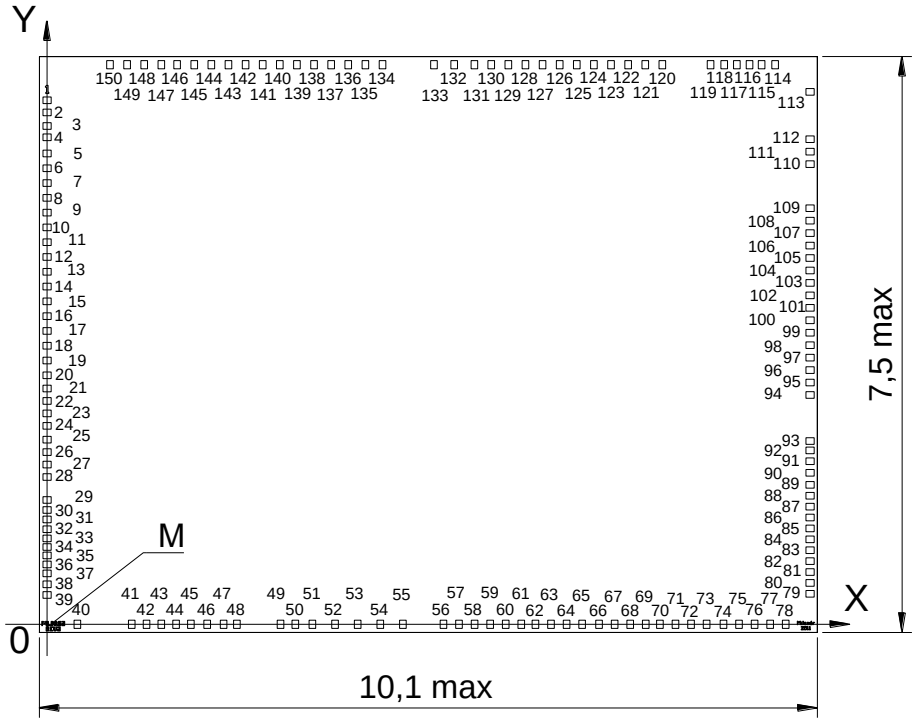


Рисунок 41–1 – Микросхема в корпусе 4229.132-3



1 Размер контактных площадок (КП) кристалла (85 х 105) мкм. Материал КП - AlCu.
2 Толщина кристалла (0,460±0,015)мм.
3 Номера КП кристалла, кроме первой, присвоены условно. Расположение КП соответствует топологическому чертежу.
4 Координаты КП приведены в таблице ниже.
5 М - Маркировка кристалла MLDR52.

Рисунок 41–2 – Кристалл (бескорпусное исполнение)

Таблица 41-1 – Координаты КП кристалла

№ КП	Обозначение КП	Координаты КП		№ КП	Обозначение КП	Координаты КП	
		X	Y			X	Y
1	U_{CCD}	0,000	6732,400	48	BIAS	2439,750	0,000
2	U_{CC}	0,000	6576,600	49	U_{CCAU}	3000,600	0,000
3	U_{CC}	0,000	6401,950	50	GND_{AU}	3190,600	0,000
4	GND	0,000	6256,950	51	INP1	3410,600	0,000
5	PF0	0,000	6049,050	52	INM1	3701,400	0,000
6	PF1	0,000	5859,050	53	INP2	3992,250	0,000
7	PF2	0,000	5669,050	54	INM2	4283,100	0,000
8	PF3	0,000	5479,050	55	MICIN	4573,900	0,000
9	PF4	0,000	5289,050	56	PE14	5095,000	0,000
10	PF5	0,000	5099,050	57	PE8	5295,000	0,000
11	PF6	0,000	4909,050	58	PE3	5495,000	0,000
12	PF7	0,000	4719,050	59	PE2	5695,000	0,000
13	PF8	0,000	4529,050	60	PE5	5895,000	0,000
14	PF9	0,000	4339,050	61	PE4	6095,000	0,000
15	PF10	0,000	4149,050	62	U_{CCA}	6277,250	0,000
16	PF11	0,000	3959,050	63	U_{CCA}	6477,250	0,000
17	PF12	0,000	3769,050	64	GND	6677,250	0,000
18	PF13	0,000	3579,050	65	GND	6877,250	0,000
19	PF14	0,000	3389,050	66	PE10	7095,100	0,000
20	PF15	0,000	3199,050	67	PE9	7295,100	0,000
21	PE15	0,000	3029,400	68	PE1	7495,100	0,000
22	PE13	0,000	2869,400	69	PE0	7695,100	0,000
23	PE12	0,000	2709,400	70	GND	7877,250	0,000
24	SHDN	0,000	2549,400	71	GND	8077,250	0,000
25	PE11	0,000	2372,150	72	U_{CCA}	8277,250	0,000
26	N/C	0,000	2212,150	73	U_{CCA}	8477,250	0,000
27	U_{CCB}	0,000	2052,150	74	PD15	8694,100	0,000
28	RESET	0,000	1892,150	75	PD14	8894,100	0,000
29	DN	0,000	1597,200	76	PD13	9094,100	0,000
30	N/C	0,000	1468,500	77	PD12	9294,100	0,000
31	N/C	0,000	1347,200	78	PD11	9494,100	0,000
32	DP	0,000	1221,050	79	PD0	9807,100	391,450
33	GND	0,000	1102,950	80	PD10	9807,100	531,450
34	GND	0,000	992,650	81	PD1	9807,100	671,450
35	U_{CCD}	0,000	884,450	82	PD7	9807,100	811,450
36	U_{CCD}	0,000	768,950	83	PD8	9807,100	951,450
37	OSC_IN	0,000	653,650	84	PD2	9807,100	1091,450
38	U_{CC}	0,000	513,650	85	PD4	9807,100	1231,450
39	U_{CC}	0,000	376,850	86	PD3	9807,100	1371,450
40	OSC_OUT	389,200	0,000	87	PD5	9807,100	1511,450
41	N/C	1088,200	0,000	88	PD6	9807,100	1651,450
42	PE7	1278,200	0,000	89	PD9	9807,100	1791,450
43	PE6	1468,200	0,000	90	U_{CC}	9807,100	1946,300
44	WAKEUP	1658,200	0,000	91	U_{CC}	9807,100	2094,800
45	StandBy	1848,200	0,000	92	GND	9807,100	2234,250
46	OUTP	2058,100	0,000	93	U_{CCD}	9807,100	2354,100
47	OUTM	2268,950	0,000	94	PC15	9807,100	2945,850

№ КП	Обозначение КП	Координаты КП		№ КП	Обозначение КП	Координаты КП	
		X	Y			X	Y
95	PC14	9807,100	3105,850	123	PB7	7248,700	7189,700
96	PC13	9807,100	3265,850	124	PB6	7028,700	7189,700
97	PC12	9807,100	3425,850	125	PB5	6808,700	7189,700
98	PC10	9807,100	3585,850	126	PB4	6588,700	7189,700
99	PC11	9807,100	3745,850	127	PB3	6368,700	7189,700
100	PC8	9807,100	3905,850	128	PB2	6148,700	7189,700
101	PC9	9807,100	4065,850	129	PB1	5928,700	7189,700
102	PC6	9807,100	4225,850	130	PB0	5708,700	7189,700
103	PC7	9807,100	4385,850	131	N/C	5492,850	7189,700
104	PC4	9807,100	4545,850	132	N/C	5232,850	7189,700
105	PC5	9807,100	4705,850	133	N/C	4972,850	7189,700
106	PC2	9807,100	4865,850	134	PA15	4311,300	7189,700
107	PC0	9807,100	5025,850	135	PA14	4091,300	7189,700
108	PC3	9807,100	5185,850	136	PA13	3871,300	7189,700
109	PC1	9807,100	5345,850	137	PA12	3651,300	7189,700
110	PB14	9807,100	5912,250	138	PA11	3431,300	7189,700
111	PB15	9807,100	6072,250	139	PA10	3211,500	7189,700
112	PB13	9807,100	6232,250	140	PA9	2991,500	7189,700
113	PB12	9807,100	6840,350	141	PA8	2771,500	7189,700
114	PB11	9357,850	7189,700	142	PA7	2551,500	7189,700
115	U_{CC}	9184,200	7189,700	143	PA6	2331,450	7189,700
116	U_{CC}	9029,200	7189,700	144	PA5	2111,450	7189,700
117	GND	8866,000	7189,700	145	PA4	1891,450	7189,700
118	GND	8703,100	7189,700	146	VPP	1671,450	7189,700
119	U_{CCD}	8528,300	7189,700	147	PA3	1468,450	7189,700
120	PB10	7908,700	7189,700	148	PA2	1248,450	7189,700
121	PB9	7688,700	7189,700	149	PA1	1028,450	7189,700
122	PB8	7468,700	7189,700	150	PA0	808,450	7189,700

42 Информация для заказа

Обозначение	Маркировка	Тип корпуса	Температурный диапазон
1901ВЦ1Т	1901ВЦ1Т	4229.132-3	минус 60 – 85 °С
К1901ВЦ1Т	К1901ВЦ1Т	4229.132-3	минус 60 – 85 °С
К1901ВЦ1ТК	К1901ВЦ1Т●	4229.132-3	0 – 70 °С

Микросхемы с приемкой «ВП» маркируются ромбом.
Микросхемы с приемкой «ОТК» маркируются буквой «К».

Примечание – Микросхемы в бескорпусном исполнении поставляются в виде отдельных кристаллов, получаемых разделением пластины. Микросхемы поставляются в таре (кейсах) без потери ориентации. Маркировка микросхемы в бескорпусном исполнении – К1901ВЦ1Н4 – наносится на тару.

Лист регистрации изменений

№ п/п	Дата	Версия	Краткое содержание изменения	№№ изменяемых листов
1	23.05.2011	1.0		
2	05.07.2011	2.0	Уточнения по результатам сдачи ОКР	По тексту
3	07.02.2013	2.1.0	Дополнено описание периферийных блоков	По тексту
4	13.02.2013	2.1.1	Исправления оформления	По тексту
5	17.09.2013	2.2.0	Исправлено описание UART - загрузчика. Испралено: -Порты загрузки через UART. -Номер блока UART, используемый для загрузки. -Таблица режимов работы микроконтроллера.	По тексту
6	11.02.2014	2.3.0	- Исправления оформления - Дополнение таблица 361: информация о смещении векторов прерываний относительно базового адреса таблицы векторов прерываний. - Исправление таблицы 34: смещения адресов регистров SSP_CLOCK2 и DSP_CONTROL_STATUS относительно базового адреса регистров блока управления синхросигналами микроконтроллера. - Исправление таблицы 50: исправлена и дополнена информация о соответствии бит регистра PER_CLOCK - Исправление таблицы 54: Дополнена информация об упрапвление синхросигналами периферийного модуля UART3. - Исправлена информация о смещение регистров CRPT_SYNR и CRPT_CR блока шифрования. - Таблица 1 дополнена описанием выводов DUcc. - Дополнены таблица 1, таблица 120, раздел организация управления DSP подсистемой описанием возможности вывода сигнала XF из DSP части напрямую на порт PA[15] микроконтроллера. - В описании регистра прерываний от DSP к RISC добавлен бит DMAIRQ.	По тексту
7	14.04.2014	2.3.1	Исправлена нумерация рисунков и таблиц	По тексту
8	22.09.2014	2.3.2	Приведение в соответствие с КД и ТУ. Выполнено форматирование. Исправлена структура заголовков, нумерация рисунков и таблиц.	По тексту
9	03.03.2015	2.4.0	– Исправлено значение тактовой частоты; – Исправлены таблица 4-8, таблица 4-19, таблица 9-13, таблица 14-3, таблица 21-2, таблица 22-4, таблица 22-7, таблица 33-5; – Исправлен рисунок 9-1	20 50, 58, 155, 193, 194, 642, 643, 651, 653, 843 148

10	13.03.2015	2.5.0	– Заменен рисунок 15-3; – Обозначение счетчика PCR исправлено на PSC; – Исправлено описание п. «Сигма-Дельта АЦП»; – Исправлено описание поля OVERCUR в таблице 22-8	252 640 645 653
11	20.03.2015	2.6.0	– Исправлено описание вывобов SHDN и NC в таблице 2-1; – Исправления в таблице 16-32	26 340
12	02.04.2015	2.6.1	Удален вывод 24 в разделе Системное управлени таблицы 2-1	25
13	12.11.2015	2.7.0	Введение бескорпусной микросхемы К1901ВЦ1Н4 Внесены исправления в таблицы 22-6, 22-7, 22-8	По тексту 256, 257