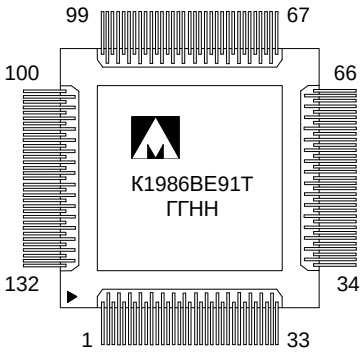
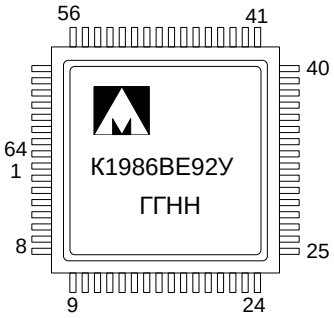
Микросхема 32-разрядного однокристального микро-ЭВМ с памятью Flash-типа 1986ВЕ9ху, К1986ВЕ9ху, К1986ВЕ9хуК К1986ВЕ92QI, К1986ВЕ92QC, 1986ВЕ91Н4, К1986ВЕ91Н4, 1986ВЕ94Н4, К1986ВЕ94Н4

1986ВЕ91Т, 1986ВЕ94Т



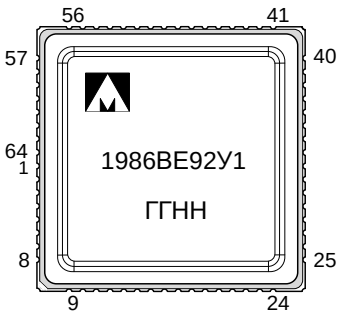
132-х выводной
металлокерамический корпус
4229.132-3

1986ВЕ92У



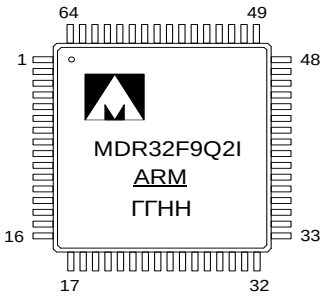
64-х выводной
металлокерамический корпус
Н18.64-1В

1986ВЕ92У1



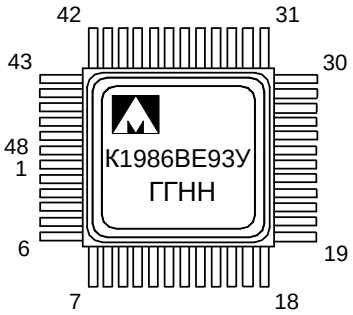
64-х выводной
металлокерамический корпус
МК 5153.64-2

К1986ВЕ92Q



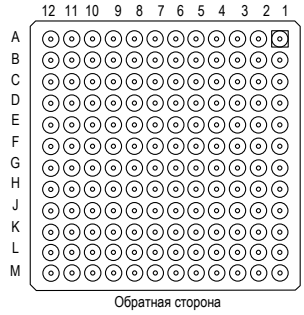
64-х выводной
пластиковый корпус LQFP64

1986ВЕ93У



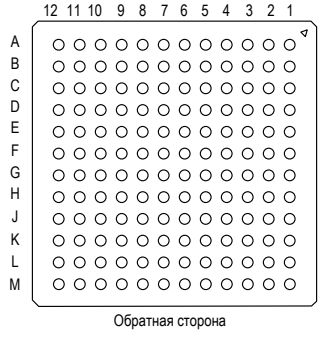
48-ми выводной
металлокерамический корпус
Н16.48-1В

1986ВЕ94Ф



144-х выводной
металлокерамический корпус
МК 6109.144-А

1986ВЕ94Я



144-х выводной
металлокерамический корпус
МК 8307.144-АН3

Обозначение	Температурный диапазон
1986ВЕ9ху	минус 60 ÷ 125 °С
К1986ВЕ9ху	минус 60 ÷ 125 °С
К1986ВЕ9хуК	0 ÷ 70 °С
К1986ВЕ9хГуК	0 ÷ 70 °С
К1986ВЕ9хДуК	0 ÷ 70 °С
К1986ВЕ92QC	0 ÷ 70 °С
К1986ВЕ92QI	минус 40 ÷ 85 °С

ГГ – год выпуска
НН – неделя выпуска

х = 1, 2, 3, 4
у = тип корпуса: Т, У, У1, Ф или Я

Микросхемы **K1986BE9xyK** являются полным функциональным аналогом микросхем K1986BE9xy с другим температурным диапазоном.

Микросхемы **K1986BE9xГyK** являются полным функциональным аналогом микросхем K1986BE9xyK, за исключением того, что обладают **немонотонным АЦП** с дифференциальной нелинейностью более |-1…+2| EMP.

Микросхемы **K1986BE9xДyK** являются полным аналогом микросхем K1986BE9xyK, за исключением того, что обладают **немонотонным ЦАП** с дифференциальной нелинейностью более |-1…+2| EMP.

Микросхемы **K1986BE92QC** и **K1986BE92QI** в пластиковом корпусе являются полным функциональным аналогом микросхемы 1986BE92У.

Микросхемы **1986BE91H4**, **K1986BE91H4**, **1986BE94H4**, **K1986BE94H4** поставляются в бескорпусном исполнении.

Основные характеристики микроконтроллеров серии 1986BE9x

Ядро:
- ARM 32-битное RISC-ядро Cortex™-M3 ревизии 2.0, тактовая частота до 80 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.

Память:
- встроенная энергонезависимая Flash-память программ размером 128 Кбайт;
- встроенное ОЗУ размером 32 Кбайт;
- контроллер внешней шины с поддержкой микросхем памяти СОЗУ, ПЗУ, NAND Flash.

Питание и тактовая частота:
- внешнее питание 2,2 ÷ 3,6 В;
- встроенный регулируемый стабилизатор напряжения на 1,8 В для питания ядра;
- встроенные схемы контроля питания;
- встроенный домен с батарейным питанием;
- встроенные подстраиваемые RC генераторы 8 МГц и 40 кГц;
- внешние кварцевые резонаторы на 2 ÷ 16 МГц и 32 кГц;
- встроенный умножитель тактовой частоты PLL для ядра;
- встроенный умножитель тактовой частоты PLL для USB.

Режим пониженного энергопотребления:
- режимы Sleep, Deep Sleep и Standby;
- батарейный домен с часами реального времени и регистрами аварийного сохранения.

Аналоговые модули:
- два 12-разрядных АЦП (до 16 каналов);
- температурный датчик;
- двухканальный 12-разрядный ЦАП;
- встроенный компаратор.

Периферия:
- контроллер DMA с функциями передачи Периферия-Память, Память-Память;
- два контроллера CAN интерфейса;
- контроллер USB интерфейса с функциями работы Device и Host;
- контроллеры интерфейсов UART, SPI, I2C;
- три 16-разрядных таймер-счетчика с функциями ШИМ и регистрации событий;
- до 96 пользовательских линий ввода-вывода.

Отладочные интерфейсы:
- последовательные интерфейсы SWD и JTAG.

Содержание

1 Введение 7
2 Основные характеристики 8
3 Структурная блок-схема микросхемы 9
4 Описание выводов 10
4.1 Диаграммы расположения выводов в корпусах с планарным расположением выводов 26
5 Указания по применению и эксплуатации 29
6 Система питания 31
6.1 Схема сброса при включении и выключении основного питания 33
7 Организация памяти 35
7.1 Секция CODE 37
7.2 Секция DATA 37
7.3 Секция PERIPHERAL 37
7.4 Секция EXTERNAL RAM 38
7.5 Секция SYSTEM 38
7.6 Блок BUS MATRIX 38
7.7 Память BOOT ROM 38
7.8 Память EEPROM 38
7.9 Память SRAM 39
7.10 Регионы памяти, типы и атрибуты 39
7.11 Последовательность обращений к памяти 40
7.12 Поведение обращений к памяти 40
7.13 Программное упорядочение обращений к памяти 41
7.14 Bit-band регионы 43
7.15 Примитивы синхронизации 44
7.16 Указания по программированию примитивов синхронизации 46
7.17 Базовые адреса процессора 47
8 Загрузочное ПЗУ и режимы работы микроконтроллера 49
8.1 UART загрузчик 51
9 Контроллер Flash-памяти программ MDR_EEPROM 56
9.1 Работа Flash-памяти программ в обычном режиме 56
9.2 Работа Flash-памяти программ в режиме программирования 57
9.3 Регистры управления контроллера Flash-памяти программ 61
10 Процессорное ядро ARM Cortex-M3 65
10.1 Программная модель 66
10.2 Стек 67
10.3 Регистры ядра 67
10.4 Исключения и прерывания 75
11 Система команд 76
11.1 Встроенные функции 80
11.2 Описание инструкций 81
11.3 Команды доступа к памяти 89
11.4 Инструкции обработки данных 105
11.5 Инструкции умножения и деления 119
11.6 Инструкции преобразования данных с насыщением 123
11.7 Команды работы с битовыми полями 125
11.8 Инструкции передачи управления 129
11.9 Прочие инструкции 137
12 Системный таймер SysTick 149

12.1 Описание регистров системного таймера SysTick ... 149
12.2 Советы и особенности при применении системного таймера ... 152
13 Модуль защиты памяти MPU ... 153
13.1 Описание регистров MPU ... 154
13.2 Советы и особенности применения MPU ... 163
14 Сигналы тактовой частоты MDR_RST_CLK ... 165
14.1 Описание регистров блока контроллера тактовой частоты ... 167
15 Батарейный домен и часы реального времени MDR_BKP ... 180
15.1 Часы реального времени ... 180
15.2 Регистры аварийного сохранения ... 181
15.3 Описание регистров блока батарейного домена ... 181
16 Порты ввода-вывода MDR_PORTx ... 190
16.1 Описание регистров портов ввода-вывода ... 193
17 Детектор напряжения питания MDR_POWER ... 197
18 Внешняя системная шина MDR_EBC (кроме 1986BE94T) ... 200
18.1 Работа с внешними статическими ОЗУ, ПЗУ и периферийными устройствами ... 200
18.2 Работа с внешней NAND Flash-памятью ... 203
18.3 Описание регистров блока контроллера внешней системной шины ... 206
19 Внешняя системная шина MDR_EBC (для 1986BE94T) ... 209
19.1 Работа с внешними статическими ОЗУ, ПЗУ и периферийными устройствами ... 209
19.2 Работа с внешней NAND Flash-памятью ... 212
19.3 Описание регистров блока контроллера внешней системной шины ... 215
20 Контроллер интерфейса MDR_USB ... 220
20.1 Инициализация контроллера при включении ... 220
20.2 Задание параметров шины USB и события подключения/отключения ... 220
20.3 Задание адреса и инициализация оконечных точек ... 221
20.4 Транзакция IN (USB Device) ... 222
20.5 Транзакция SETUP/OUT (USB Device) ... 224
20.6 Транзакция SETUP/OUT (USB Host) ... 225
20.7 Транзакция IN (USB Host) ... 227
20.8 Отправка SOF пакетов и отсчет времени (USB Host) ... 228
20.9 Описание регистров управление контроллером USB интерфейса ... 228
21 Контроллер интерфейса MDR_CAN ... 248
21.1 Режимы работы ... 249
21.2 Типы пакетов сообщений ... 250
21.3 Структура пакета данных (Data Frame) ... 251
21.4 Инициализация ... 255
21.5 Передача сообщений ... 256
21.6 Передача сообщений по Remote Transmit Request (RTR) ... 256
21.7 Прием сообщений ... 256
21.8 Автоматическая фильтрация принимаемых сообщений ... 257
21.9 Перезапись принятых сообщений ... 257
21.10 Задание скорости передачи и момента семплирования ... 257
21.11 Синхронизация ... 258
21.12 Обработка ошибок ... 259
21.13 Прерывания ... 262
21.14 Описание регистров контроллера CAN ... 263
22 Таймеры общего назначения MDR_TIMERx ... 278
22.1 Функционирование ... 278
22.2 Режимы счета ... 280
22.3 Источник событий для счета ... 282

22.4 Режим захвата ... 289
22.5 Режим ШИМ ... 291
22.6 Примеры ... 294
22.7 Описание регистров блока таймера ... 298
23 Контроллер MDR_ADC ... 314
23.1 Преобразование внешнего канала ... 315
23.2 Последовательное преобразование нескольких каналов ... 316
23.3 Преобразование с контролем границ ... 316
23.4 Внутренний источник опорного напряжения ... 316
23.5 Датчик температуры ... 317
23.6 Синхронный запуск двух АЦП ... 317
23.7 Время заряда внутренней емкости ... 318
23.8 Описание регистров блока контроллера АЦП ... 320
24 Контроллер MDR_DAC ... 328
24.1 Описание регистров блока контроллера ЦАП ... 328
25 Контроллер схемы компаратора MDR_COMP ... 331
25.1 Сравнение внешних сигналов ... 332
25.2 Сравнение сигнала с внутренним источником опорного напряжения ... 332
25.3 Сравнение внешних сигналов с внутренней шкалой напряжений ... 332
25.4 Формирование внутренней шкалы напряжений ... 332
25.5 Описание регистров блока контроллера компаратора ... 334
26 Контроллер интерфейса MDR_I2C ... 336
26.1 Конфигурация системы ... 336
26.2 Протокол I2C ... 336
26.3 Сигнал START ... 337
26.4 Передача адреса ... 337
26.5 Передача данных ... 337
26.6 Сигнал STOP ... 338
26.7 Описание регистров контроллера I2C ... 338
27 Контроллер MDR_SSP ... 342
27.1 Основные характеристики модуля SSP ... 342
27.2 Программируемые параметры ... 342
27.3 Характеристики интерфейса SPI ... 343
27.4 Характеристики интерфейса Microwire ... 343
27.5 Характеристики интерфейса SSI ... 344
27.6 Общий обзор модуля SSP ... 344
27.7 Программное управление модулем ... 362
27.8 Прерывания ... 368
28 Контроллер MDR_UART ... 370
28.1 Основные сведения ... 370
28.2 Функциональные возможности ... 371
28.3 Описание функционирования блока UART ... 373
28.4 Описание функционирования ИК кодека IrDA SIR ... 375
28.5 Описание работы UART ... 377
28.6 Линии управления модемом ... 382
28.7 Интерфейс прямого доступа к памяти ... 383
28.8 Прерывания ... 386
28.9 Программное управление модулем ... 388
29 Контроллер прямого доступа в память MDR_DMA ... 406
29.1 Основные свойства контроллера DMA ... 406
29.2 Термины и определения ... 406

29.3 Функциональное описание ... 408
29.4 Управление DMA ... 411
29.5 Структура управляющих данных канала ... 433
29.6 Описание регистров контроллера DMA ... 444
30 Прерывания и исключения ... 464
30.1 Типы исключений ... 464
30.2 Прерывания (IRQ) ... 465
30.3 Обработчики исключений ... 466
30.4 Таблица векторов ... 466
30.5 Приоритеты исключений ... 468
30.6 Вход в обработчик и выход из обработчика ... 468
30.7 Обработка отказов ... 471
30.8 Управление электропитанием ... 473
31 Контроллер прерываний NVIC ... 476
31.1 Упрощенный доступ к регистрам контроллера прерываний ... 477
31.2 Прерывания, срабатывающие по уровню сигнала ... 482
31.3 Аппаратное и программное управление прерываниями ... 482
31.4 Рекомендации по работе с контроллером прерываний ... 483
32 Блок управления системой ... 484
32.1 Упрощенный доступ к регистрам блока управления системой ... 485
33 Сторожевые таймеры ... 501
33.1 Описание регистров блока сторожевых таймеров ... 501
34 Предельно допустимые характеристики микросхемы ... 506
35 Электрические параметры микросхемы ... 509
36 Справочные данные ... 511
37 Типовые зависимости ... 515
38 Габаритный чертеж микросхемы ... 520
39 Информация для заказа ... 531
40 Лист регистрации изменений ... 532

1 Введение

Микроконтроллеры серии 1986BE9x, K1986BE9x и K1986BE92QI, K1986BE92QC (далее 1986BE9x), построенные на базе высокопроизводительного процессорного RISC ядра ARM Cortex-M3, содержат встроенную 128 Кбайт Flash-память программ и 32 Кбайт ОЗУ. Микроконтроллеры работают на тактовой частоте до 80 МГц. Периферия микроконтроллера включает контроллер USB интерфейса со встроенным аналоговым приемопередатчиком со скоростями передачи 12 Мбит/с (Full Speed) и 1,5 Мбит/с (Low Speed), стандартные интерфейсы UART, SPI и I2C, контроллер внешней системной шины, что позволяет работать с внешними микросхемами статического ОЗУ и ПЗУ, NAND Flash-памятью и другими внешними устройствами. Микроконтроллеры содержат три 16-разрядных таймера с 4-мя каналами схем захвата и ШИМ с функциями формирования «мертвой зоны» и аппаратной блокировки, а также системный 24-х разрядный таймер и два сторожевых таймера. Кроме того, в состав микроконтроллеров входят: два 12-разрядных высокоскоростных (до 0,5 Мвыборок/с) АЦП с возможностью оцифровки информации от 16 внешних каналов и от встроенных датчиков температуры и опорного напряжения; два 12-разрядных ЦАП; встроенный компаратор 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 две схемы умножения тактовой частоты PLL для ядра и USB интерфейса позволяют гибко настраивать скорость работы микроконтроллеров.

Архитектура системы памяти за счет матрицы системных шин позволяет минимизировать возможные конфликты при работе системы и повысить общую производительность. Контроллер DMA позволяет ускорить обмен информацией между ОЗУ и периферией без участия процессорного ядра.

Встроенный регулятор, предназначенный для формирования питания внутренней цифровой части, формирует напряжение 1,8 В и не требует дополнительных внешних элементов. Таким образом, для работы микроконтроллера достаточно одного внешнего напряжения питания в диапазоне от 2,2 до 3,6 В. Также в микроконтроллерах реализован батарейный домен, работающий от внешней батареи, который предназначен для обеспечения функций часов реального времени и сохранения некоторого объёма данных при отсутствии основного питания. Встроенные детекторы напряжения питания могут отслеживать уровень внешнего основного питания, уровень напряжения питания на батарее. Аппаратные схемы сброса при просадке питания позволяют исключить сбойную работу микросхемы при выходе уровня напряжения питания за допустимые пределы.

2 Основные характеристики

В зависимости от корпуса, в котором выпускается микросхема, изменяются функциональные возможности микроконтроллеров, но при этом объем памяти программ и ОЗУ остается одинаковым.

Таблица 1 – Основные характеристики микроконтроллеров серии 1986BE9x

	1986BE91T **1986BE94T**	**1986BE94Ф** **1986BE94Я**	**K1986BE91H4**	**1986BE92У** **1986BE92У1** **K1986BE92QI** **K1986BE92QC**	**1986BE93У**
Корпус	132 вывода	144 вывода	бескорпусная	64 вывода	48 выводов
Ядро	ARM Cortex-M3				
ПЗУ	128 Кбайт Flash				
ОЗУ	32 Кбайт				
Питание	2,2…3,6 В				
Частота	80 МГц				
USER IO	96	96	96	43	30
USB	Device и Host FS (до 12 Мбит/с) встроенный PHY				
UART	2	2	2	2	2
CAN	2	2	2	2	2
SPI	2	2	2	2	1
I2C	1	1	1	1	-
2 x 12-разрядных АЦП	16 каналов	16	16 каналов	8 каналов	4 канала
ЦАП 12 разрядов	2	2	2	1	1
Компаратор	3 входа	3 входа	3 входа	2 входа	2 входа
Внешняя шина	32 разряда	32 разряда	32 разряда	8 разрядов	-

3 Структурная блок-схема микросхемы

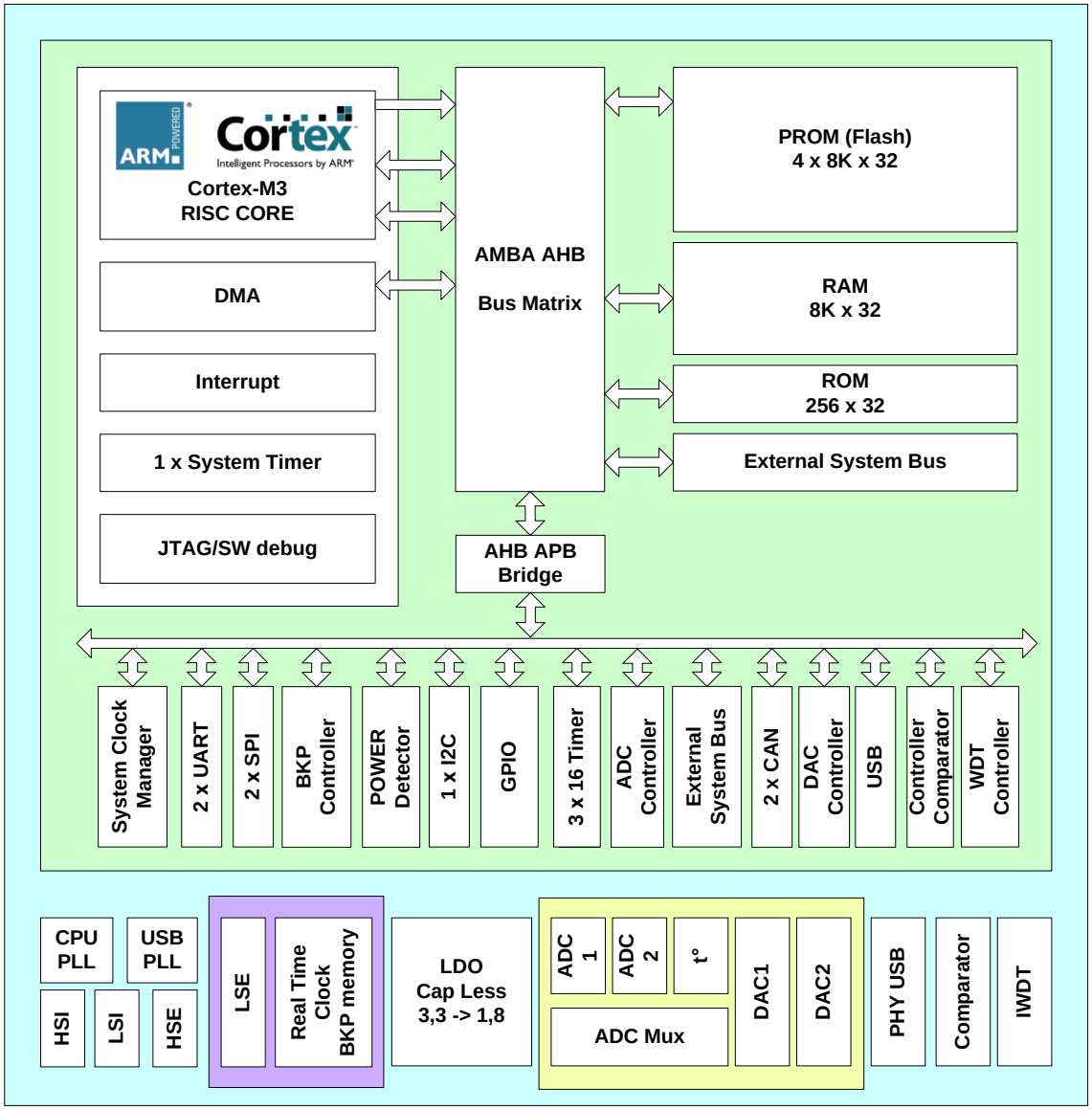


Рисунок 1. Структурная блок-схема микроконтроллера 1986BE9xy

4 Описание выводов

Таблица 2 – Описание выводов микроконтроллеров серии 1986BE9x в корпусах 4229.132-3, H18.64-1B, H16.48-1B, МК 5153.64-2, LQFP64

Вывод	Контактная площадка кристалла	Тип корпуса					Дополнительные функции вывода			
		4229.132-3	H18.64-1B	H16.48-1B	МК 5153.64-2	LQFP64	Аналог.	Основ.	Альтер.	Переопр.
Порт A										
PA0	137	130	55	41	55	63	-	DATA0	EXT_INT1	-
PA1	136	129	54	40	54	62	-	DATA1	TMR1_CH1	TMR2_CH1
PA2	135	128	53	39	53	61	-	DATA2	TMR1_CH1N	TMR2_CH1N
PA3	134	127	52	38	52	60	-	DATA3	TMR1_CH2	TMR2_CH2
PA4	133	126	51	37	51	59	-	DATA4	TMR1_CH2N	TMR2_CH2N
PA5	132	125	50	36	50	58	-	DATA5	TMR1_CH3	TMR2_CH3
PA6	131	124	49	35	49	57	-	DATA6	CAN1_TX	UART1_RXD
PA7	130	123	48	34	48	56	-	DATA7	CAN1_RX	UART1_TXD
PA8	129	122	-	-	-	-	-	DATA8	TMR1_CH3N	TMR2_CH3N
PA9	128	121	-	-	-	-	-	DATA9	TMR1_CH4	TMR2_CH4
PA10	125	119	-	-	-	-	-	DATA10	nUART1DTR	TMR2_CH4N
PA11	124	118	-	-	-	-	-	DATA11	nUART1RTS	TMR2_BLK
PA12	123	117	-	-	-	-	-	DATA12	nUART1RI	TMR2_ETR
PA13	122	115	-	-	-	-	-	DATA13	nUART1DCD	TMR1_CH4N
PA14	121	114	-	-	-	-	-	DATA14	nUART1DSR	TMR1_BLK
PA15	120	113	-	-	-	-	-	DATA15	nUART1CTS	TMR1_ETR
Порт B										
PB0 JA_TDO	97	92	35	25	35	43	-	DATA16	TMR3_CH1	UART1_TXD
PB1 JA_TMS	98	93	36	26	36	44	-	DATA17	TMR3_CH1N	UART2_RXD
PB2 JA_TCK	99	94	37	27	37	45	-	DATA18	TMR3_CH2	CAN1_TX
PB3 JA_TDI	100	95	38	28	38	46	-	DATA19	TMR3_CH2N	CAN1_RX
PB4 JA_TRST	101	96	39	29	39	47	-	DATA20	TMR3_BLK	TMR3_ETR
PB5	107	102	42	32	42	50	-	DATA21	UART1_TXD	TMR3_CH3
PB6	108	103	43	33	43	51	-	DATA22	UART1_RXD	TMR3_CH3N
PB7	109	104	44	-	44	52	-	DATA23	nSIROUT1	TMR3_CH4
PB8	110	105	45	-	45	53	-	DATA24	COMP_OUT	TMR3_CH4N
PB9	111	106	46	-	46	54	-	DATA25	nSIRIN1	EXT_INT4
PB10	112	107	47	-	47	55	-	DATA26	EXT_INT2	nSIROUT1
PB11	113	108	-	-	-	-	-	DATA27	EXT_INT1	COMP_OUT
PB12	114	109	-	-	-	-	-	DATA28	SSP1_FSS	SSP2_FSS
PB13	115	110	-	-	-	-	-	DATA29	SSP1_CLK	SSP2_CLK
PB14	116	111	-	-	-	-	-	DATA30	SSP1_RXD	SSP2_RXD
PB15	119	112	-	-	-	-	-	DATA31	SSP1_TXD	SSP2_TXD
Порт C										
PC0	96	91	34	23	34	42	-	READY*	SCL1	SSP2_FSS
PC1	95	90	33	-	33	41	-	OE	SDA1	SSP2_CLK
PC2	94	89	31	-	31	40	-	WE	TMR3_CH1	SSP2_RXD
PC3	93	88	-	-	-	-	-	BE0	TMR3_CH1N	SSP2_TXD
PC4	92	87	-	-	-	-	-	BE1	TMR3_CH2	TMR1_CH1
PC5	91	86	-	-	-	-	-	BE2	TMR3_CH2N	TMR1_CH1N
PC6	90	85	-	-	-	-	-	BE3	TMR3_CH3	TMR1_CH2
PC7	89	84	-	-	-	-	-	CLOCK	TMR3_CH3N	TMR1_CH2N
PC8	88	83	-	-	-	-	-	CAN1_TX	TMR3_CH4	TMR1_CH3
PC9	87	82	-	-	-	-	-	CAN1_RX	TMR3_CH4N	TMR1_CH3N

Вывод	Контактная площадка кристалла	Тип корпуса					Дополнительные функции вывода			
		4229.132-3	H18.64-1B	H16.48-1B	МК 5153.64-2	LQFP64	Аналог.	Основ.	Альтер.	Переопр.
PC10	86	81	-	-	-	-	-	-	TMR3_ETR	TMR1_CH4
PC11	85	80	-	-	-	-	-	-	TMR3_BLK	TMR1_CH4N
PC12	84	79	-	-	-	-	-	-	EXT_INT2	TMR1_ETR
PC13	83	78	-	-	-	-	-	-	EXT_INT4	TMR1_BLK
PC14	82	77	-	-	-	-	-	-	SSP2_FSS	CAN2_RX
PC15	81	76	-	-	-	-	-	-	SSP2_RXD	CAN2_TX
Порт D										
PD0 JB_TMS	70	65	23	17	23	31	ADC0_REF+	TMR1_CH1N	UART2_RXD	TMR3_CH1
PD1 JB_TCK	71	66	24	18	24	32	ADC1_REF-	TMR1_CH1	UART2_TXD	TMR3_CH1N
PD2 JB_TRST	72	67	25	19	25	33	ADC2	BUSY1	SSP2_RXD	TMR3_CH2
PD3 JB_TDI	73	68	26	20	26	34	ADC3	-	SSP2_FSS	TMR3_CH2N
PD4 JB_TDO	69	64	22	-	22	30	ADC4	TMR1_ETR	nSIROUT2	TMR3_BLK
PD5	74	69	27	-	27	35	ADC5	CLE	SSP2_CLK	TMR2_ETR
PD6	75	70	28	-	28	36	ADC6	ALE	SSP2_TXD	TMR2_BLK
PD7	68	63	21	-	21	29	ADC7	TMR1_BLK	nSIRIN2	UART1_RXD
PD8	67	62	-	-	-	-	ADC8	TMR1_CH4N	TMR2_CH1	UART1_TXD
PD9	76	71	-	-	-	-	ADC9	CAN2_TX	TMR2_CH1N	SSP1_FSS
PD10	66	61	-	-	-	-	ADC10	TMR1_CH2	TMR2_CH2	SSP1_CLK
PD11	65	60	-	-	-	-	ADC11	TMR1_CH2N	TMR2_CH2N	SSP1_RXD
PD12	64	59	-	-	-	-	ADC12	TMR1_CH3	TMR2_CH3	SSP1_TXD
PD13	63	58	-	-	-	-	ADC13	TMR1_CH3N	TMR2_CH3N	CAN1_TX
PD14	62	57	-	-	-	-	ADC14	TMR1_CH4	TMR2_CH4	CAN1_RX
PD15	61	56	-	-	-	-	ADC15	CAN2_RX	BUSY2	EXT_INT3
Порт E										
PE0	56	53	18	14	18	26	DAC2_OUT	ADDR16	TMR2_CH1	CAN1_RX
PE1	55	52	17	-	17	25	DAC2_REF	ADDR17	TMR2_CH1N	CAN1_TX
PE2	48	45	14	11	14	22	COMP_IN1	ADDR18	TMR2_CH3	TMR3_CH1
PE3	47	44	13	10	13	21	COMP_IN2	ADDR19	TMR2_CH3N	TMR3_CH1N
PE4	45	42	-	-	-	-	COMP_REF+	ADDR20	TMR2_CH4N	TMR3_CH2
PE5	44	41	-	-	-	-	COMP_REF-	ADDR21	TMR2_BLK	TMR3_CH2N
PE6	36	33	8	6	8	16	OSC_IN32	ADDR22	CAN2_RX	TMR3_CH3
PE7	35	32	7	-	7	15	OSC_OUT32	ADDR23	CAN2_TX	TMR3_CH3N
PE8	46	43	-	-	-	-	COMP_IN3	ADDR24	TMR2_CH4	TMR3_CH4
PE9	54	51	-	-	-	-	DAC1_OUT	ADDR25	TMR2_CH2	TMR3_CH4N
PE10	53	50	-	-	-	-	DAC1_REF	ADDR26	TMR2_CH2N	TMR3_ETR
PE11	26	23	-	-	-	-	-	ADDR27	nSIRIN1	TMR3_BLK
PE12	21	20	-	-	-	-	-	ADDR28	SSP1_RXD	UART1_RXD

Вывод	Контактная площадка кристалла	Тип корпуса					Дополнительные функции вывода			
		4229.132-3	H18.64-1B	H16.48-1B	МК 5153.64-2	LQFP64	Аналог.	Основ.	Альтер.	Переопр.
PE13	20	19	-	-	-	-	-	ADDR29	SSP1_FSS	UART1_TXD
PE14	43	40	-	-	-	-	-	ADDR30	TMR2_ETR	SCL1
PE15	19	18	-	-	-	-	-	ADDR31	EXT_INT3	SDA1
Порт F										
PF0	3	2	58	44	58	2	-	ADDR0	SSP1_TXD	UART2_RXD
PF1	4	3	59	45	59	3	-	ADDR1	SSP1_CLK	UART2_TXD
PF2	5	4	60	46	60	4	-	ADDR2	SSP1_FSS	CAN2_RX
PF3	6	5	61	47	61	5	-	ADDR3	SSP1_RXD	CAN2_TX
PF4 MODE[0]	7	6	62	48	62	6	-	ADDR4	-	-
PF5 MODE[1]	8	7	63	1	63	7	-	ADDR5	-	-
PF6 MODE[2]	9	8	64	-	64	8	-	ADDR6	TMR1_CH1	-
PF7	10	9	-	-	-	-	-	ADDR7	TMR1_CH1N	TMR3_CH1
PF8	11	10	-	-	-	-	-	ADDR8	TMR1_CH2	TMR3_CH1N
PF9	12	11	-	-	-	-	-	ADDR9	TMR1_CH2N	TMR3_CH2
PF10	13	12	-	-	-	-	-	ADDR10	TMR1_CH3	TMR3_CH2N
PF11	14	13	-	-	-	-	-	ADDR11	TMR1_CH3N	TMR3_ETR
PF12	15	14	-	-	-	-	-	ADDR12	TMR1_CH4	SSP2_FSS
PF13	16	15	-	-	-	-	-	ADDR13	TMR1_CH4N	SSP2_CLK
PF14	17	16	-	-	-	-	-	ADDR14	TMR1_ETR	SSP2_RXD
PF15	18	17	-	-	-	-	-	ADDR15	TMR1_BLK	SSP2_TXD
Системное управление										
RESET	40	37	10	7	10	18	Сигнал внешнего сброса 0 – сброс 1 – разрешена работа			
WAKEUP	38	35	9	-	9	17	Сигнал внешнего выхода из режима Standby 0 – выход из режима StandBy 1 – нет сигнала выхода			
STANDBY	34	31	6	-	6	14	Флаг режима Standby 0 – микросхема не в режиме StandBy 1 – микросхема в режиме StandBy			
OSC_IN	41	38	11	8	11	19	Вход генератора HSE			
OSC_OUT	42	39	12	9	12	20	Выход генератора HSE			
USB интерфейс										
DP	22	21	1	2	1	9	Шина USB D+			
DN	25	22	2	3	2	10	Шина USB D-			
Питание										
U_{CC}	1,2,31,32, 77,78,103, 104	1,28,29, 72,73, 98,99	4,29,40, 57	5,21,30,4 3	4, 29, 40, 57	1, 12, 38, 48	Основное питание 2,2…3,6 В			
AU_{CC}	59,60	55	20	16	20	28	Аналоговое питание АЦП, ЦАП и Comparator 2,4…3,6 В			
AU_{CC1}	51,52	48,49	16	13	16	24	Аналоговое питание PLL 2,2…3,6 В			
BU_{CC}	33	30	5	-	5	13	Батарейное питание 1.8…3,6 В			
GND	29,30,79, 105,139	26,27, 74,100, 132	3,30,41, 56	4,22,31,4 2	3, 30, 41, 56	11, 39, 49, 64	Общий			
AGND	57,58	54	19	15	19	27	Общий			
AGND1	49,50	46,47	15	12	15	23	Общий			
DU_{CC}	28,80,106, 138	25,75,10 1,131	-	-	-	-	Рекомендуется не подсоединять			
Выводы для тестирования и исследования										
BDU_{CC}	37	34	-	-	-	-	Рекомендуется не подсоединять			

Вывод	Контактная площадка кристалла	Тип корпуса					Дополнительные функции вывода			
		4229.132-3	H18.64-1B	H16.48-1B	МК 5153.64-2	LQFP64	Аналог.	Основ.	Альтер.	Переопр.
EXT_POR	39	36	-	-	-	-	Рекомендуется не подсоединять или подключить к шине «Общий»			
SHDN	27	24	-	-	-	-	Тестовый вход включения/отключения внутреннего регулятора LDO. Рекомендуется не подсоединять или подключить к шине «Общий»			
JTAG_EN	102	97	-	-	-	-	Рекомендуется не подсоединять или подключить к шине «Общий»			
Не используются										
NC	23,24, 117,118, 126,127	116,120	32	24	32	37	Рекомендуется не подсоединять или подключить к шине «Общий»			

* – реализовано только в микроконтроллере 1986BE94x.

Таблица 3 – Описание выводов микроконтроллеров серии 1986BE9x в корпусах МК 6109.144-А, МК 8307.144-АН3

Номер вывода	Обозначение вывода	Назначение и функции вывода		
		Тип функции вывода	Обозначение функции вывода	Функциональное назначение вывода
Порт А				
B2	PA0	Вход/выход 0 порта А		
		Аналоговая	–	–
		Основная	DATA0	Разряд 0 шины данных DATA[31:0] EXT_BUS
		Альтернативная	EXT_INT1	Вход внешнего прерывания
		Переопределенная	–	–
A3	PA1	Вход/выход 1 порта А		
		Аналоговая	–	–
		Основная	DATA1	Разряд 1 шины данных DATA[31:0] EXT_BUS
		Альтернативная	TMR1_CH1	Вход/выход канала 1 таймера 1
		Переопределенная	TMR2_CH1	Вход/выход канала 1 таймера 2
B3	PA2	Вход/выход 2 порта А		
		Аналоговая	–	–
		Основная	DATA2	Разряд 2 шины данных DATA[31:0] EXT_BUS
		Альтернативная	TMR1_CH1N	Инверсный выход канала 1 таймера 1
		Переопределенная	TMR2_CH1N	Инверсный выход канала 1 таймера 2
C3	PA3	Вход/выход 3 порта А		
		Аналоговая	–	–
		Основная	DATA3	Разряд 3 шины данных DATA[31:0] EXT_BUS
		Альтернативная	TMR1_CH2	Вход/выход канала 2 таймера 1
		Переопределенная	TMR2_CH2	Вход/выход канала 2 таймера 2
D4	PA4	Вход/выход 4 порта А		
		Аналоговая	–	–
		Основная	DATA4	Разряд 4 шины данных DATA[31:0] EXT_BUS
		Альтернативная	TMR1_CH2N	Инверсный выход канала 2 таймера 1
		Переопределенная	TMR2_CH2N	Инверсный выход канала 2 таймера 2

Номер вывода	Обозначение вывода	Назначение и функции вывода		
		Тип функции вывода	**Обозначение функции вывода**	**Функциональное назначение вывода**
A4	PA5	Вход/выход 5 порта A		
		Аналоговая	–	–
		Основная	DATA5	Разряд 5 шины данных DATA[31:0] EXT_BUS
		Альтернативная	TMR1_CH3	Вход/выход канала 3 таймера 1
		Переопределенная	TMR2_CH3	Вход/выход канала 3 таймера 2
B4	PA6	Вход/выход 6 порта A		
		Аналоговая	–	–
		Основная	DATA6	Разряд 6 шины данных DATA[31:0] EXT_BUS
		Альтернативная	CAN1_TX	Выходные данные интерфейса CAN1
		Переопределенная	UART1_RXD	Входные данные UART1
C4	PA7	Вход/выход 7 порта A		
		Аналоговая	–	
		Основная	DATA7	Разряд 7 шины данных DATA[31:0] EXT_BUS
		Альтернативная	CAN1_RX	Входные данные интерфейса CAN1
		Переопределенная	UART1_TXD	Выходные данные UART1
C5	PA8	Вход/выход 8 порта A		
		Аналоговая	–	–
		Основная	DATA8	Разряд 8 шины данных DATA[31:0] EXT_BUS
		Альтернативная	TMR1_CH3N	Инверсный выход канала 3 таймера 1
		Переопределенная	TMR2_CH3N	Инверсный выход канала 3 таймера 2
A5	PA9	Вход/выход 9 порта A		
		Аналоговая	–	–
		Основная функция	DATA9	Разряд 9 шины данных DATA[31:0] EXT_BUS
		Альтернативная	TMR1_CH4	Вход/выход канала 4 таймера 1
		Переопределенная	TMR2_CH4	Вход/выход канала 4 таймера 2
B5	PA10	Вход/выход 10 порта A		
		Аналоговая	–	–
		Основная	DATA10	Разряд 10 шины данных DATA[31:0] EXT_BUS
		Альтернативная ф	nUART1DTR	Выход готовности для передачи данных UART1
		Переопределенная	TMR2_CH4N	Инверсный выход канала 4 таймера 2
D5	PA11	Вход/выход 11 порта A		
		Аналоговая	–	–
		Основная функция	DATA11	Разряд 11 шины данных DATA[31:0] EXT_BUS
		Альтернативная	nUART1RTS	Выход запроса на передачу данных UART1
		Переопределенная	TMR2_BLK	Вход внешнего события таймера 2
D6	PA12	Вход/выход 12 порта A		
		Аналоговая	–	–
		Основная	DATA12	Разряд 12 шины данных DATA[31:0] EXT_BUS
		Альтернативная	nUART1RI	Вход сигнала вызова от источника
		Переопределенная	TMR2_ETR	Вход внешнего события таймера 2

Номер вывода	Обозначение вывода	Назначение и функции вывода		
		Тип функции вывода	**Обозначение функции вывода**	**Функциональное назначение вывода**
A6	PA13	Вход/выход 13 порта A		
		Аналоговая	–	–
		Основная	DATA13	Разряд 13 шины данных DATA[31:0] EXT_BUS
		Альтернативная	nUART1DCD	Вход информационного сигнала от источника
		Переопределенная	TMR1_CH4N	Инверсный выход канала 4 таймера 1
B6	PA14	Вход/выход 14 порта A		
		Аналоговая	–	–
		Основная функция	DATA14	Разряд 14 шины данных DATA[31:0] EXT_BUS
		Альтернативная	nUART1DSR	Вход готовности для передачи данных источником
		Переопределенная	TMR1_BLK	Вход внешнего события таймера 1
C6	PA15	Вход/выход 15 порта A		
		Аналоговая	–	–
		Основная	DATA15	Разряд 15 шины данных DATA[31:0] EXT_BUS
		Альтернативная	nUART1CTS	Вход готовности для приема данных источником
		Переопределенная	TMR1_ETR	Вход внешнего события таймера 1
Порт B				
D11	PB0/ JA_TDO	Вход/выход 0 порта B/ Выход тестовых данных JTAG-интерфейса		
		Аналоговая	–	–
		Основная	DATA16	Разряд 16 шины данных DATA[31:0] EXT_BUS
		Альтернативная	TMR3_CH1	Вход/выход канала 1 таймера 3
		Переопределенная	UART1_TXD	Выходные данные UART1
D12	PB1/ JA_TMS	Вход/выход 1 порта B/ Выбор режима тестирования JTAG интерфейса		
		Аналоговая	–	–
		Основная	DATA17	Разряд 17 шины данных DATA[31:0] EXT_BUS
		Альтернативная	TMR3_CH1N	Инверсный выход канала 1 таймера 3
		Переопределенная	UART2_RXD	Входные данные UART2
D9	PB2/ JA_TCK	Вход/выход 2 порта B/ Тестовое тактирование JTAG интерфейса		
		Аналоговая	–	–
		Основная	DATA18	Разряд 18 шины данных DATA[31:0] EXT_BUS
		Альтернативная	TMR3_CH2	Вход/выход канала 2 таймера 3
		Переопределенная	CAN1_TX	Выходные данные интерфейса CAN1
C10	PB3/ JA_TDI	Вход/выход 3 порта B/ Вход тестовых данных JTAG интерфейса		
		Аналоговая	–	–
		Основная	DATA19	Разряд 19 шины данных DATA[31:0] EXT_BUS
		Альтернативная	TMR3_CH2N	Инверсный выход канала 2 таймера 3
		Переопределенная	CAN1_RX	Входные данные интерфейса CAN1
C11	PB4/ JA_TRST	Вход/выход 4 порта B/ Тестовый сброс JTAG интерфейса		
		Аналоговая	–	–
		Основная	DATA20	Разряд 20 шины данных DATA[31:0] EXT_BUS
		Альтернативная	TMR3_BRK	Асинхронный вход внешнего события таймера 3
		Переопределенная	TMR3_ETR	Синхронный вход внешнего события таймера 3

Номер вывода	Обозначение вывода	Назначение и функции вывода		
		Тип функции вывода	**Обозначение функции вывода**	**Функциональное назначение вывода**
A10	PB5	Вход/выход 5 порта B		
		Аналоговая	–	–
		Основная	DATA21	Разряд 21 шины данных DATA[31:0] EXT_BUS
		Альтернативная	UART1_TXD	Выходные данные UART1
		Переопределенная	TMR3_CH3	Вход/выход канала 3 таймера 3
B10	PB6	Вход/выход 6 порта B		
		Аналоговая	–	–
		Основная	DATA22	Разряд 22 шины данных DATA[31:0] EXT_BUS
		Альтернативная	UART1_RXD	Входные данные UART1
		Переопределенная	TMR3_CH3N	Инверсный выход канала 3 таймера 3
B9	PB7	Вход/выход 7 порта B		
		Аналоговая	–	–
		Основная	DATA23	Разряд 23 шины данных DATA[31:0] EXT_BUS
		Альтернативная	nSIROUT1	Выходные данные IRDA SIR ART1
		Переопределенная	TMR3_CH4	Вход/выход канала 4 таймера 3
A9	PB8	Вход/выход 8 порта B		
		Аналоговая	–	–
		Основная	DATA24	Разряд 24 шины данных DATA[31:0] EXT_BUS
		Альтернативная	COMP_OUT	Выходные данные компаратора
		Переопределенная	TMR3_CH4N	Инверсный выход канала 4 таймера 3
D8	PB9	Вход/выход 9 порта B		
		Аналоговая	–	–
		Основная	DATA25	Разряд 25 шины данных DATA[31:0] EXT_BUS
		Альтернативная	nSIRIN1	Входные данные IRDA SIR UART1
		Переопределенная	EXT_INT4	Вход внешнего прерывания
C8	PB10	Вход/выход 10 порта B		
		Аналоговая	–	–
		Основная	DATA26	Разряд 26 шины данных DATA[31:0] EXT_BUS
		Альтернативная	EXT_INT2	Вход внешнего прерывания
		Переопределенная	nSIROUT1	Выходные данные IRDA SIR UART1
B8	PB11	Вход/выход 11 порта B		
		Аналоговая	–	–
		Основная	DATA27	Разряд 27 шины данных DATA[31:0] EXT_BUS
		Альтернативная	EXT_INT1	Вход внешнего прерывания
		Переопределенная	COMP_OUT	Выходные данные компаратора
A8	PB12	Вход/выход 12 порта B		
		Аналоговая	–	–
		Основная	DATA28	Разряд 28 шины данных DATA[31:0] EXT_BUS
		Альтернативная	SSP1_FSS	Вход/выход разрешения SPI1
		Переопределенная	SSP2_FSS	Вход/выход разрешения SPI2

Номер вывода	Обозначение вывода	Назначение и функции вывода		
		Тип функции вывода	**Обозначение функции вывода**	**Функциональное назначение вывода**
B7	PB13	Вход/выход 13 порта B		
		Аналоговая	–	–
		Основная	DATA29	Разряд 29 шины данных DATA[31:0] EXT_BUS
		Альтернативная	SSP1_CLK	Вход/выход синхросигнала SPI1
		Переопределенная	SSP2_CLK	Вход/выход синхросигнала SPI2
A7	PB14	Вход/выход 14 порта B		
		Аналоговая	–	–
		Основная	DATA30	Разряд 30 шины данных DATA[31:0] EXT_BUS
		Альтернативная	SSP1_RXD	Входные данные SPI1
		Переопределенная	SSP2_RXD	Входные данные SPI2
C7	PB15	Вход/выход 15 порта B		
		Аналоговая	–	–
		Основная	DATA31	Разряд 31 шины данных DATA[31:0] EXT_BUS
		Альтернативная	SSP1_TXD	Выходные данные SPI1
		Переопределенная	SSP2_TXD	Выходные данные SPI2
Порт C				
D10	PC0	Вход/выход 0 порта C		
		Аналоговая	–	–
		Основная я	–	
		Альтернативная	SCL1	Выход синхросигнала I2C
		Переопределенная	SSP2_FSS	Вход/выход разрешения SPI 2
E10	PC1	Вход/выход 1 порта C		
		Аналоговая	–	–
		Основная	OE	Выход сигнала разрешения выдачи данных EXT_BUS
		Альтернативная	SDA1	Вход/выход данных I2C
		Переопределенная	SSP2_CLK	Вход/выход синхросигнала SPI2
E12	PC2	Вход/выход 2 порта C		
		Аналоговая	–	–
		Основная	WE	Выход сигнала разрешения записи данных EXT_BUS
		Альтернативная	TMR3_CH1	Вход/выход канала 1 таймера 3
		Переопределенная	SSP2_RXD	Входные данные SPI2
E11	PC3	Вход/выход 3 порта C		
		Аналоговая	–	–
		Основная	BE0	Выход разрешения байта 0 в 32 разрядном слове шины данных EXT_BUS
		Альтернативная	TMR3_CH1N	Инверсный выход канала 1 таймера 3
		Переопределенная	SSP2_TXD	Выходные данные SPI2
E9	PC4	Вход/выход 4 порта C		
		Аналоговая	–	–
		Основная	BE1	Выход разрешения байта 1 в 32 разрядном слове шины данных EXT_BUS
		Альтернативная	TMR3_CH2	Вход/выход канала 2 таймера 3
		Переопределенная	TMR1_CH1	Вход/выход канала 1 таймера 1

Номер вывода	Обозначение вывода	Назначение и функции вывода		
		Тип функции вывода	**Обозначение функции вывода**	**Функциональное назначение вывода**
F9	PC5	Вход/выход 5 порта C		
		Аналоговая	–	
		Основная	BE2	Выход разрешения байта 2 в 32 разрядном слове шины данных EXT_BUS
		Альтернативная	TMR3_CH2N	Инверсный выход канала 2 таймера 3
		Переопределенная	TMR1_CH1N	Инверсный выход канала 1 таймера 1
F12	PC6	Вход/выход 6 порта C		
		Аналоговая	–	–
		Основная	BE3	Выход разрешения байта 3 в 32 разрядном слове шины данных EXT_BUS
		Альтернативная	TMR3_CH3	Вход/выход канала 3 таймера 3
		Переопределенная	TMR1_CH2	Вход/выход канала 2 таймера 1
F11	PC7	Вход/выход 7 порта C		
		Аналоговая	–	–
		Основная	CLOCK	Выход синхросигнала EXT_BUS
		Альтернативная	TMR3_CH3N	Инверсный выход канала 3 таймера 3
		Переопределенная	TMR1_CH2N	Инверсный выход канала 2 таймера 1
F10	PC8	Вход/выход 8 порта C		
		Аналоговая	–	–
		Основная	CAN1_TX	Выходные данные интерфейса CAN1
		Альтернативная	TMR3_CH4	Вход/выход канала 4 таймера 3
		Переопределенная	TMR1_CH3	Вход/выход канала 3 таймера 1
G10	PC9	Вход/выход 9 порта C		
		Аналоговая	–	–
		Основная	CAN1_RX	Входные данные интерфейса CAN1
		Альтернативная	TMR3_CH4N	Инверсный выход канала 4 таймера 3
		Переопределенная	TMR1_CH3N	Инверсный выход канала 3 таймера 1
G12	PC10	Вход/выход 10 порта C		
		Аналоговая	–	–
		Основная	–	–
		Альтернативная	TMR3_ETR	Вход внешнего события таймера 3
		Переопределенная	TMR1_CH4	Вход/выход канала 4 таймера 1
G11	PC11	Вход/выход 11 порта C		
		Аналоговая	–	–
		Основная	–	–
		Альтернативная	TMR3_BLK	Вход внешнего события таймера 3
		Переопределенная	TMR1_CH4N	Инверсный выход канала 4 таймера 1
G9	PC12	Вход/выход 12 порта C		
		Аналоговая	–	–
		Основная	–	–
		Альтернативная	EXT_INT2	Вход внешнего прерывания
		Переопределенная	TMR1_ETR	Вход внешнего события таймера 1

Номер вывода	Обозначение вывода	Назначение и функции вывода		
		Тип функции вывода	**Обозначение функции вывода**	**Функциональное назначение вывода**
H12	PC13	Вход/выход 13 порта C		
		Аналоговая	–	–
		Основная	–	–
		Альтернативная	EXT_INT4	Вход внешнего прерывания
		Переопределенная	TMR1_BLK	Вход внешнего события таймера 1
H11	PC14	Вход/выход 14 порта C		
		Аналоговая	–	–
		Основная функция	–	–
		Альтернативная	SSP2_FSS	Вход/выход разрешения SPI2
		Переопределенная	CAN2_RX	Входные данные интерфейса CAN2
H10	PC15	Вход/выход 15 порта C		
		Аналоговая	–	–
		Основная	–	–
		Альтернативная	SSP2_RXD	Входные данные SPI2
		Переопределенная	CAN2_TX	Выходные данные интерфейса CAN2
Порт D				
L11	PD0/ JB_TMS	Вход/выход 0 порта D/ Выбор режима тестирования JTAG интерфейса		
		Аналоговая	ADC0_REF+	Положительное опорное напряжение /Входной канал 0 АЦП
		Основная	TMR1_CH1N	Инверсный выход канала 1 таймера 1
		Альтернативная	UART2_RXD	Входные данные UART2
		Переопределенная	TMR3_CH1	Вход/выход канала 1 таймера 3
M11	PD1/ JB_TCK	Вход/выход 1 порта D/ Тестовое тактирование JTAG интерфейса		
		Аналоговая	ADC1_REF-	Отрицательное опорное напряжение / Входной канал 1 АЦП
		Основная	TMR1_CH1	Вход/выход канала 1 таймера 1
		Альтернативная	UART2_TXD	Выходные данные UART2
		Переопределенная	TMR3_CH1N	Инверсный выход канала 1 таймера 3
L12	PD2/ JB_TRST	Вход/выход 2 порта D/ Тестовый сброс JTAG интерфейса		
		Аналоговая	ADC2	Входной канал 2 АЦП
		Основная	BUSY1	Вход сигнала занятости внешнего устройства EXT_BUS
		Альтернативная	SSP2_RXD	Входные данные SPI2
		Переопределенная	TMR3_CH2	Вход/выход канала 2 таймера 3
M12	PD3/ JB_TDI	Вход/выход 3 порта D/ Вход тестовых данных JTAG интерфейса		
		Аналоговая	ADC3	Входной канал 3 АЦП
		Основная	–	–
		Альтернативная	SSP2_FSS	Вход/выход разрешения SPI2
		Переопределенная	TMR3_CH2N	Инверсный выход канала 3 таймера 2
M10	PD4/ JB_TDO	Вход/выход 4 порта D/ Выход тестовых данных JTAG-интерфейса		
		Аналоговая	ADC4	Входной канал 4 АЦП
		Основная	TMR1_ETR	Вход внешнего события таймера 1
		Альтернативная	nSIROUT2	Выходные данные IRDA SIR UART2
		Переопределенная	TMR3_BLK	Вход внешнего события таймера 3

Номер вывода	Обозначение вывода	Назначение и функции вывода		
		Тип функции вывода	**Обозначение функции вывода**	**Функциональное назначение вывода**
K12	PD5	Вход/выход 5 порта D		
		Аналоговая	ADC5	Входной канал 5 АЦП
		Основная	CLE	Выход разрешения защёлкивания команды EXT_BUS
		Альтернативная	SSP2_CLK	Вход/выход синхросигнала SPI2
		Переопределенная	TMR2_ETR	Вход внешнего события таймера 2
K11	PD6	Вход/выход 6 порта D		
		Аналоговая	ADC6	Входной канал 6 АЦП
		Основная	ALE	Выход разрешения защёлкивания адреса EXT_BUS
		Альтернативная	SSP2_TXD	Выходные данные SPI2
		Переопределенная	TMR2_BLK	Вход внешнего события таймера 2
L10	PD7	Вход/выход 7 порта D		
		Аналоговая	ADC7	Входной канал 7 АЦП
		Основная	TMR1_BLK	Вход внешнего события таймера 1
		Альтернативная	nSIRIN2	Входные данные IRDA SIR UART2
		Переопределенная	UART1_RXD	Входные данные UART1
K10	PD8	Вход/выход 8 порта D		
		Аналоговая	ADC8	Входной канал 8 АЦП
		Основная	TMR1_CH4N	Инверсный выход канала 4 таймера 1
		Альтернативная	TMR2_CH1	Вход/выход канала 1 таймера 2
		Переопределенная	UART1_TXD	Выходные данные UART1
J10	PD9	Вход/выход 9 порта D		
		Аналоговая	ADC9	Входной канал 9 АЦП
		Основная	CAN2_TX	Выходные данные интерфейса CAN2
		Альтернативная	TMR2_CH1N	Инверсный выход канала 1 таймера 2
		Переопределенная	SSP1_FSS	Вход/выход разрешения SPI1
J9	PD10	Вход/выход 10 порта D		
		Аналоговая	ADC10	Входной канал 10 АЦП
		Основная	TMR1_CH2	Вход/выход канала 2 таймера 1
		Альтернативная	TMR2_CH2	Вход/выход канала 2 таймера 2
		Переопределенная	SSP1_CLK	Вход/выход синхросигнала SPI1
M9	PD11	Вход/выход 11 порта D		
		Аналоговая	ADC11	Входной канал 11 АЦП
		Основная	TMR1_CH2N	Инверсный выход канала 2 таймера 1
		Альтернативная	TMR2_CH2N	Инверсный выход канала 2 таймера 2
		Переопределенная	SSP1_RXD	Входные данные SPI1
L9	PD12	Вход/выход 12 порта D		
		Аналоговая	ADC12	Входной канал 12 АЦП
		Основная	TMR1_CH3	Вход/выход канала 3 таймера 1
		Альтернативная	TMR2_CH3	Вход/выход канала 3 таймера 2
		Переопределенная	SSP1_TXD	Выходные данные SPI1

Номер вывода	Обозначение вывода	Назначение и функции вывода		
		Тип функции вывода	**Обозначение функции вывода**	**Функциональное назначение вывода**
K9	PD13	Вход/выход 13 порта D		
		Аналоговая	ADC13	Входной канал 13 АЦП
		Основная	TMR1_CH3N	Инверсный выход канала 3 таймера 1
		Альтернативная	TMR2_CH3N	Инверсный выход канала 3 таймера 2
		Переопределенная	CAN1_TX	Выходные данные интерфейса CAN1
K8	PD14	Вход/выход 14 порта D		
		Аналоговая	ADC14	Входной канал 14 АЦП
		Основная	TMR1_CH4	Вход/выход канала 4 таймера 1
		Альтернативная	TMR2_CH4	Вход/выход канала 4 таймера 2
		Переопределенная	CAN1_RX	Входные данные интерфейса CAN1
M8	PD15	Вход/выход 15 порта D		
		Аналоговая	ADC15	Входной канал 15 АЦП
		Основная	CAN2_RX	Входные данные интерфейса CAN2
		Альтернативная	BUSY2	Вход сигнала занятости внешнего устройства EXT_BUS
		Переопределенная	EXT_INT3	Вход внешнего прерывания
Порт E				
L7	PE0	Вход/выход 0 порта E		
		Аналоговая	DAC2_OUT	Аналоговый выход ЦАП2
		Основная	ADDR16	Разряд 16 шины адреса
		Альтернативная	TMR2_CH1	Вход/выход канала 1 таймера 2
		Переопределенная	CAN1_RX	Входные данные интерфейса CAN1
K7	PE1	Вход/выход 1 порта E		
		Аналоговая	DAC2_REF	Опорное напряжение ЦАП 2
		Основная	ADDR17	Разряд 17 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	TMR2_CH1N	Инверсный выход канала 1 таймера 2
		Переопределенная	CAN1_TX	Выходные данные интерфейса CAN1
L5	PE2	Вход/выход 2 порта E		
		Аналоговая	COMP_IN1	Вход 1 компаратора
		Основная	ADDR18	Разряд 18 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	TMR2_CH3	Вход/выход канала 3 таймера 2
		Переопределенная	TMR3_CH1	Вход/выход канала 1 таймера 3
K5	PE3	Вход/выход 3 порта E		
		Аналоговая	COMP_IN2	Вход 2 компаратора
		Основная	ADDR19	Разряд 19 шины адреса
		Альтернативная	TMR2_CH3N	Инверсный выход канала 3 таймера 2
		Переопределенная	TMR3_CH1N	Инверсный выход канала 1 таймера 3
M4	PE4	Вход/выход 4 порта E		
		Аналоговая	COMP_REF+	Положительное опорное напряжение компаратора
		Основная	ADDR20	Разряд 20 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	TMR2_CH4N	Инверсный выход канала 4 таймера 2
		Переопределенная	TMR3_CH2	Вход/выход канала 2 таймера 3

Номер вывода	Обозначение вывода	Назначение и функции вывода		
		Тип функции вывода	**Обозначение функции вывода**	**Функциональное назначение вывода**
L4	PE5	Вход/выход 5 порта E		
		Аналоговая	COMP_REF-	Отрицательное опорное напряжение компаратора
		Основная	ADDR21	Разряд 21 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	TMR2_BLK	Вход внешнего события таймера 2
		Переопределенная	TMR3_CH2N	Инверсный выход канала 2 таймера 3
L1	PE6	Вход/выход 6 порта E		
		Аналоговая	OSC_IN32	Вход осциллятора 32 кГц
		Основная	ADDR22	Разряд 22 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	CAN2_RX	Входные данные интерфейса CAN2
		Переопределенная	TMR3_CH3	Вход/выход канала 3 таймера 3
L2	PE7	Вход/выход 7 порта E		
		Аналоговая	OSC_OUT32	Выход осциллятора 32 кГц
		Основная	ADDR23	Разряд 23 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	CAN2_TX	Выходные данные интерфейса CAN2
		Переопределенная	TMR3_CH3N	Инверсный выход канала 3 таймера 3
J5	PE8	Вход/выход 8 порта E		
		Аналоговая	COMP_IN3	Вход 3 компаратора
		Основная	ADDR24	Разряд 24 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	TMR2_CH4	Вход/выход канала 4 таймера 2
		Переопределенная	TMR3_CH4	Вход/выход канала 4 таймера 3
K6	PE9	Вход/выход 9 порта E		
		Аналоговая	DAC1_OUT	Аналоговый выход ЦАП1
		Основная	ADDR25	Разряд 25 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	TMR2_CH2	Вход/выход канала 2 таймера 2
		Переопределенная	TMR3_CH4N	Инверсный выход канала 4 таймера 3
M6	PE10	Вход/выход 10 порта E		
		Аналоговая	DAC1_REF	Опорное напряжение ЦАП1
		Основная	ADDR26	Разряд 26 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	TMR2_CH2N	Инверсный выход канала 2 таймера 2
		Переопределенная	TMR3_ETR	Синхронный вход внешнего события таймера 3
H3	PE11	Вход/выход 11 порта E		
		Аналоговая	–	–
		Основная	ADDR27	Разряд 27 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	nSIRIN1	Входные данные IRDA SIR UART1
		Переопределенная	TMR3_BRK	Асинхронный вход внешнего события таймера 3
H4	PE12	Вход/выход 12 порта E		
		Аналоговая	–	–
		Основная	ADDR28	Разряд 28 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	SSP1_RXD	Входные данные SPI1
		Переопределенная	UART1_RXD	Входные данные UART1

Номер вывода	Обозначение вывода	Назначение и функции вывода		
		Тип функции вывода	**Обозначение функции вывода**	**Функциональное назначение вывода**
G4	PE13	Вход/выход 13 порта E		
		Аналоговая	–	–
		Основная	ADDR29	Разряд 29 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	SSP1_FSS	Вход/выход разрешения SPI1
		Переопределенная	UART1_TXD	Выходные данные UART1
K4	PE14	Вход/выход 14 порта E		
		Аналоговая	–	–
		Основная	ADDR30	Разряд 30 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	TMR2_ETR	Вход внешнего события таймера 2
		Переопределенная	SCL1	Выход синхросигнала I2C
G1	PE15	Вход/выход 15 порта E		
		Аналоговая	–	–
		Основная	ADDR31	Разряд 31 шины адреса
		Альтернативная	EXT_INT3	Вход внешнего прерывания
		Переопределенная	SDA1	Вход/выход данных I2C
Порт F				
C1	PF0	Вход/выход 0 порта F		
		Аналоговая	–	–
		Основная	ADDR0	Разряд 0 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	SSP1_TXD	Выходные данные SPI1
		Переопределенная	UART2_RXD	Входные данные UART2
C2	PF1	Вход/выход 1 порта F		
		Аналоговая	–	–
		Основная	ADDR1	Разряд 1 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	SSP1_CLK	Вход/выход синхросигнала SPI1
		Переопределенная	UART2_TXD	Выходные данные UART2
D3	PF2	Вход/выход 2 порта F		
		Аналоговая	–	–
		Основная	ADDR2	Разряд 2 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	SSP1_FSS	Вход/выход разрешения SPI1
		Переопределенная	CAN2_RX	Входные данные интерфейса CAN2
D2	PF3	Вход/выход 3 порта F		
		Аналоговая	–	
		Основная	ADDR3	Разряд 3 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	SSP1_RXD	Входные данные SPI1
		Переопределенная	CAN2_TX	Выходные данные интерфейса CAN2
D1	PF4/ MODE[0]	Вход/выход 4 порта F/ Режим работы микроконтроллера		
		Аналоговая	–	–
		Основная	ADDR4	Разряд 4 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	–	–
		Переопределенная	–	–

Номер вывода	Обозначение вывода	Назначение и функции вывода		
		Тип функции вывода	**Обозначение функции вывода**	**Функциональное назначение вывода**
E4	PF5/ MODE[1]	Вход/выход 5 порта F/ Режим работы микроконтроллера		
		Аналоговая	–	–
		Основная	ADDR5	Разряд 5 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	–	–
		Переопределенная	–	–
E3	PF6/ MODE[2]	Вход/выход 6 порта F/ Режим работы микроконтроллера		
		Аналоговая	–	–
		Основная	ADDR6	Разряд 6 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	TMR1_CH1	Вход/выход канала 1 таймера 1
		Переопределенная	–	–
E2	PF7	Вход/выход 7 порта F		
		Аналоговая	–	–
		Основная	ADDR7	Разряд 7 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	TMR1_CH1N	Инверсный выход канала 1 таймера 1
		Переопределенная	TMR3_CH1	Вход/выход канала 1 таймера 3
E1	PF8	Вход/выход 8 порта F		
		Аналоговая	–	–
		Основная	ADDR8	Разряд 8 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	TMR1_CH2	Вход/выход канала 2 таймера 1
		Переопределенная	TMR3_CH1N	Инверсный выход канала 1 таймера 3
E5	PF9	Вход/выход 9 порта F		
		Аналоговая	–	–
		Основная	ADDR9	Разряд 9 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	TMR1_CH2N	Инверсный выход канала 2 таймера 1
		Переопределенная	TMR3_CH2	Вход/выход канала 2 таймера 3
F4	PF10	Вход/выход 10 порта F		
		Аналоговая	–	–
		Основная	ADDR10	Разряд 10 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	TMR1_CH3	Вход/выход канала 3 таймера 1
		Переопределенная	TMR3_CH2N	Инверсный выход канала 2 таймера 3
F2	PF11	Вход/выход 11 порта F		
		Аналоговая	–	–
		Основная	ADDR11	Разряд 11 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	TMR1_CH3N	Инверсный выход канала 3 таймера 1
		Переопределенная	TMR3_ETR	Вход внешнего события таймера 3
F1	PF12	Вход/выход 12 порта F		
		Аналоговая	–	–
		Основная	ADDR12	Разряд 12 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	TMR1_CH4	Вход/выход канала 4 таймера 1
		Переопределенная	SSP2_FSS	Вход/выход разрешения SPI2

Номер вывода	Обозначение вывода	Назначение и функции вывода		
		Тип функции вывода	**Обозначение функции вывода**	**Функциональное назначение вывода**
F3	PF13	Вход/выход 13 порта F		
		Аналоговая	–	–
		Основная	ADDR13	Разряд 13 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	TMR1_CH4N	Инверсный выход канала 4 таймера 1
		Переопределенная	SSP2_CLK	Вход/выход синхросигнала SPI2
G3	PF14	Вход/выход 14 порта F		
		Аналоговая	–	–
		Основная	ADDR14	Разряд 14 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	TMR1_ETR	Вход внешнего события таймера 1
		Переопределенная	SSP2_RXD	Входные данные SPI2
G2	PF15	Вход/выход 15 порта F		
		Аналоговая	–	–
		Основная	ADDR15	Разряд 15 шины адреса ADDR [31:0] EXT_BUS
		Альтернативная	TMR1_BLK	Вход внешнего события таймера 1
		Переопределенная	SSP2_TXD	Выходные данные SPI2
Системное управление				
M1	RESET	Сигнал внешнего сброса		
M2	WAKEUP	Сигнал внешнего выхода из режима Standby		
K1	STANDBY	Флаг режима Standby		
M3	OSC_IN	Вход генератора HSE		
L3	OSC_OUT	Выход генератора HSE		
USB интерфейс				
H2	DP	Шина USB D+		
H1	DN	Шина USB D-		
Питание				
A1, B1, B11, B12, J4, J11, J12, K3	Ucc	Питание (2,2 – 3,6) В		
J8, L8	AUcc	Аналоговое питание (должно совпадать с UCC)		
J6, L6	AUcc1	Аналоговое питание (должно совпадать с UCC)		
K2	BUcc	Батарейное питание (1,8 – 3,6) В		
J1	GND	Общий		
J7, M7	AGND	Общий		
H5, M5	AGND1	Общий		
E6, E7, F5-F8, G5-G8, H6, H7, крышка корпуса	GND_EXP	Общий		
Выводы для тестирования				
A2, A11, H9, J2	DUcc	Тестовый выход внутреннего питания1,8 В. Рекомендуется не подсоединять		
J3	SHDN	Тестовый вход включения/отключения внутреннего регулятора LDO. Рекомендуется не подсоединять или подключить к шине «Общий»		

Номер вывода	Обозначение вывода	Назначение и функции вывода		
		Тип функции вывода	Обозначение функции вывода	Функциональное назначение вывода
C12	JTAG_EN	Тестовый вход включения/отключения TAP-контроллера. Рекомендуется не подсоединять или подключить к шине «Общий»		
Не используются				
A12, C9, D7, E8, H8	NC	Не используются. Рекомендуется не подсоединять или подключить к шине «Общий»		

4.1 Диаграммы расположения выводов в корпусах с планарным расположением выводов

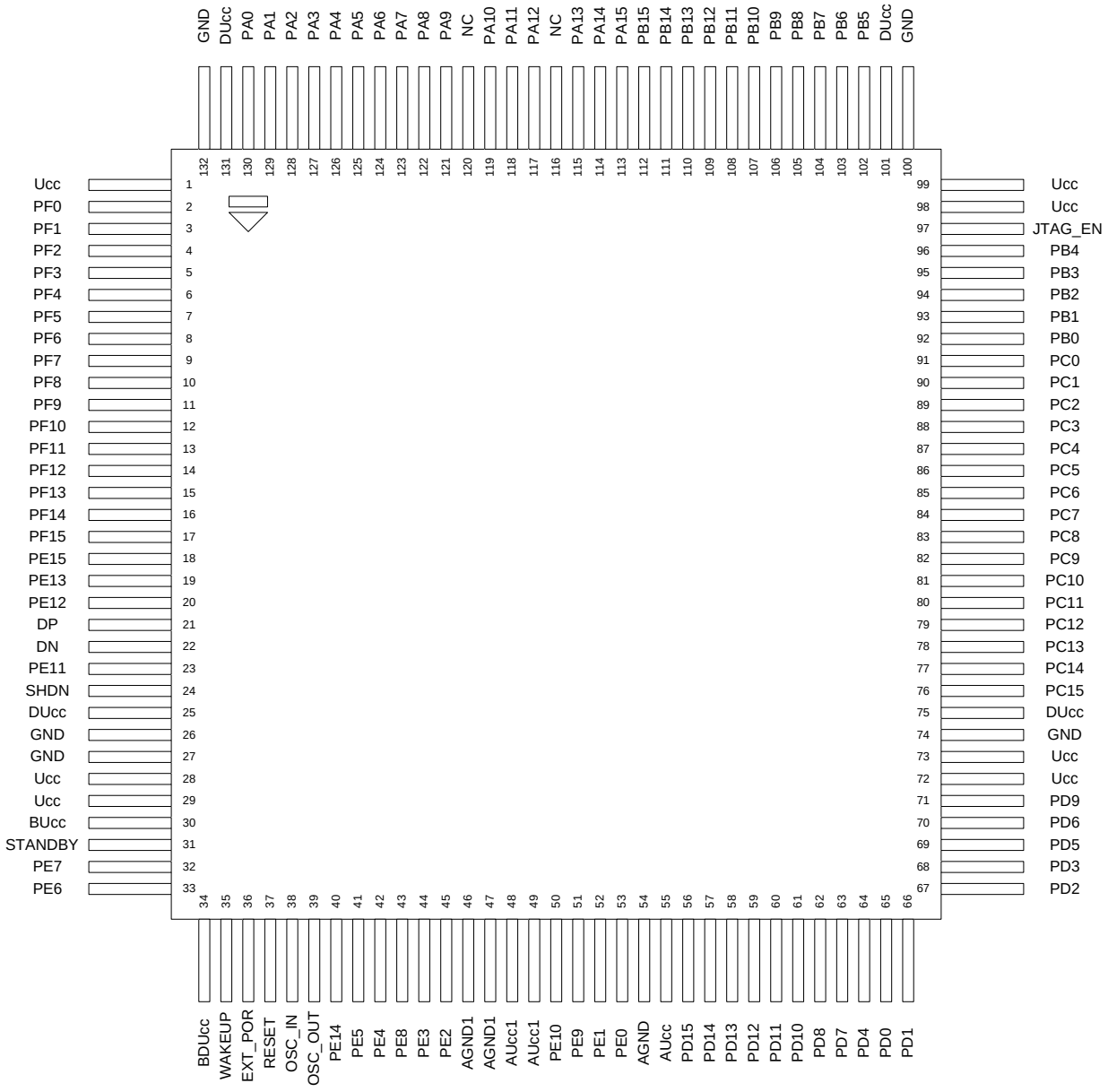


Рисунок 2. Расположение выводов в 132-х выводном корпусе 4229.132-3

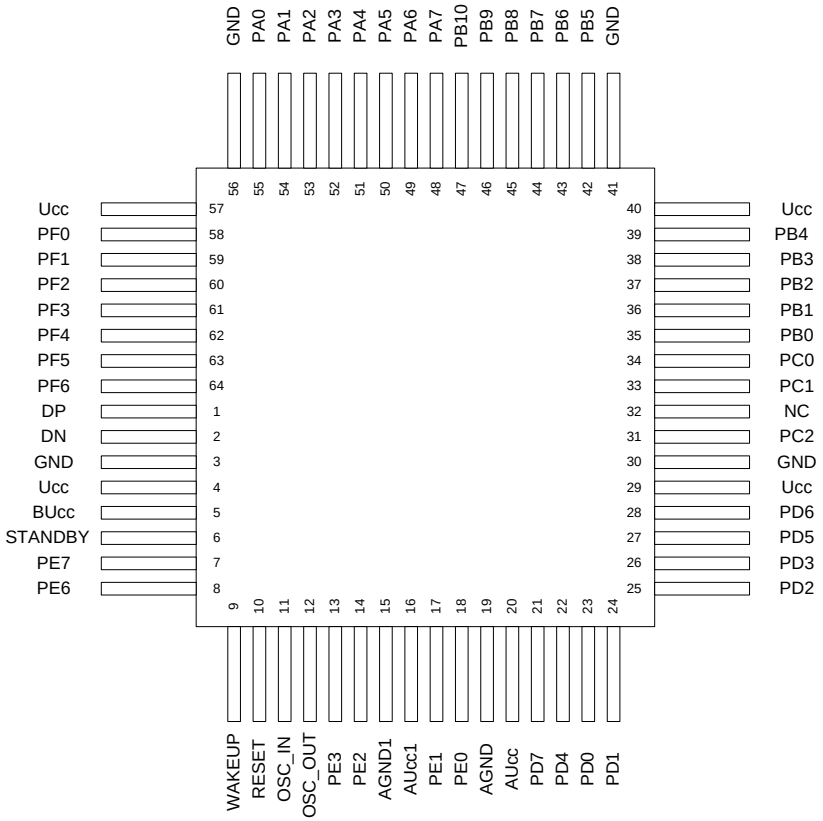


Рисунок 3. Расположение выводов в 64-х выводном корпусе H18.64-1B

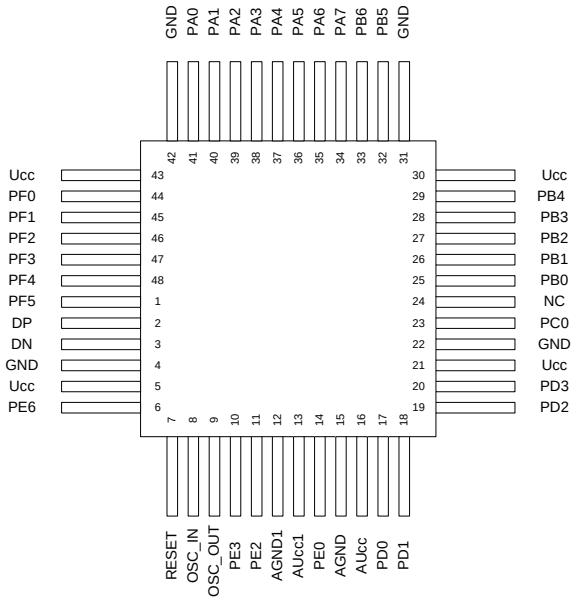


Рисунок 4. Расположение выводов в 48-ми выводном корпусе H16.48-1B

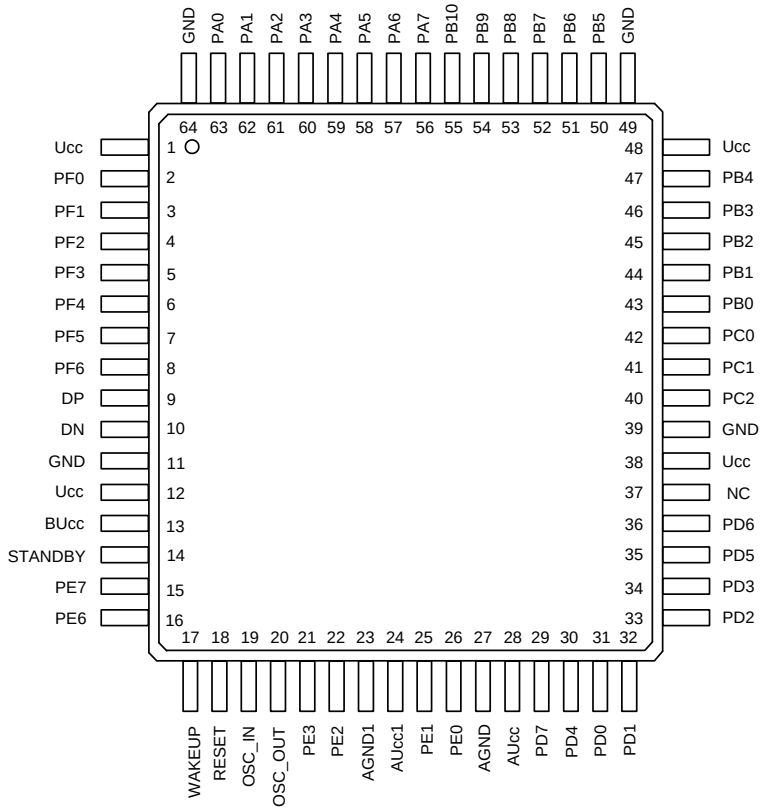


Рисунок 5. Расположение выводов в 64-х выводном пластиковом корпусе LQFP64

5 Указания по применению и эксплуатации

Между выводами «Питание» и «Общий» устанавливается фильтрующая емкость не менее 0,1 мкФ.

Для задания режимов MODE при запуске микросхем должны использоваться резисторы (номинал не более 10 кОм) доопределения до «0» или питания, подключаемые к следующим выводам:

- 6, 7, 8 микросхем 1986BE91T, 1986BE94T;
- 62, 63, 64 микросхем 1986BE92У, 1986BE92У;
- 48,1 микросхем 1986BE93У;
- D1, E3, E4 микросхем 1986BE94Ф, 1986BE94Я.

При эксплуатации микросхем 1986BE93У не допускается:

- запись единицы в 6-й бит регистров MDR_PORTF->OE, MDR_PORTF->ANALOG, MDR_PORTF->GFEN и MDR_ PORTF->RXTX;
- запись единицы в 12-й и/или 13-й биты регистров MDR_ PORTF->FUNC и MDR_ PORTF->PWR;
- запись единицы в 6-й и/или 22-й биты регистров MDR_ PORTF->PULL и MDR_ PORTF->PD.

Данные биты всегда должны иметь нулевое значение.

При использовании выводов порта B и порта D, задействованных для отладочных интерфейсов, а также при ошибочном отключении сигналов тактовой синхронизации возможно нарушение работы отладочного интерфейса. Это может привести к невозможности последующего стирания и перепрограммирования микросхемы.

Микросхемы 1986BE94Я пригодны для монтажа в аппаратуре методом групповой пайки при условии соблюдения требуемого температурного профиля паяльной пасты и равномерном прогреве места монтажа микросхемы по всей его площади. Скорость нагрева и охлаждения компонентов монтажа не должна превышать 4 °C/c. Параметры профиля пайки оловянно-свинцовой паяльной пастой приведены в таблице 4.

Таблица 4 – Параметры профиля пайки оловянно-свинцовой паяльной пастой

Параметр	**Оловянно-свинцовый припой**
Температура солидуса припоя, °C	183
Температурный диапазон пайки припоем, °C	210–220
Минимальная пиковая температура пайки (в самой холодной точке платы), °C	205
Скорость нагрева компонента, °C/c	1-4
Скорость охлаждения компонента, °C/c	2-4
Температура предварительного нагрева, °C	100-180
Длительность предварительного нагрева, с	60-120
Время пребывания в расплавленном состоянии, с	60-90
Максимальная выдержка при пиковой температуре, с	20

Микросхемы 1986ВЕ94Я пригодны для монтажа методом ручной пайки непосредственным присоединением с оплавлением шариков. Материал шариков: Sn63 / Pb37.

Прогнозируемая зависимость показателей надежности микросхем от температуры кристалла приведена на рисунке 4.

Крышка микросхем электрически соединена:

– с выводом 116 микросхем 1986ВЕ91Т, 1986ВЕ94Т;

– с выводом 32 микросхем 1986ВЕ92У;

– с выводом 24 микросхем 1986ВЕ93У;

– с выводами E6, E7, F5, F6, F7, G5, G6, G7, G8, H6, H7 микросхем 1986ВЕ94Ф, 1986ВЕ94Я.

Не допускается подведение каких-либо сигналов, кроме сигнала «Общий», к указанным выводам микросхем.

Металлизация обратной стороны микросхем 1986ВЕ92У1 должна быть соединена с шиной «Общий».

Не допускается подведение каких-либо сигналов, кроме сигнала «Общий», к выводам:

– 36, 24, 97 микросхем 1986ВЕ91Т, 1986ВЕ94Т;

– J3, C12 микросхем 1986ВЕ94Ф, 1986ВЕ94Я.

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

Свободными рекомендуется оставлять выводы:

– 25, 34, 75, 101, 131 микросхем 1986ВЕ91Т, 1986ВЕ94Т;

– A2, A11, H9, J2 микросхем 1986ВЕ94Ф, 1986ВЕ94Я.

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

Запрещается подведение каких–либо электрических сигналов (в том числе шин «Питание», «Общий») к выводам микросхем, не используемым согласно схеме электрической.

6 Система питания

Микроконтроллеры серии 1986BE9x имеют несколько типов выводов питания.

U_{CC} выводы: Основное питание микросхемы, включает питание пользовательских выводов, встроенного регулятора напряжения, USB PHY и генераторов. Входное напряжение должно быть в пределах от 2,2 до 3,6 В. Если используется интерфейс USB, то входное напряжение должно быть в пределах от 3,0 до 3,6 В. Если используется АЦП или ЦАП, то входное напряжение должно быть в пределах от 2,4 до 3,6 В.

DU_{CC} выводы: Питание внутренней цифровой части, памяти ОЗУ и Flash-памяти. Это питание формируется внутренним регулятором напряжения из U_{CC}. В нормальном режиме работы этот вывод должен остаться неподсоединенным. В некоторых корпусах данные выводы отсутствуют. Напряжение на выводе DU_{CC} должно быть в пределах от 1,62 до 1,98 В.

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

BDU_{CC} вывод: Результирующие напряжения после выбора между BU_{CC} и U_{CC} при питании батарейного домена. В нормальном режиме этот вывод должен остаться неподсоединенным. В некоторых корпусах данные выводы отсутствуют.

AU_{CC} выводы: Питание аналоговых блоков АЦП, ЦАП и Компаратора выведено на отдельные выводы для уменьшения помех, создаваемых работой других блоков. На данные выводы должно подаваться напряжение из того же источника, что и U_{CC}, но при этом на печатной плате должны быть применены меры по снижению помех. Для корректной работы 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.

$AU_{CC}1$ выводы: Питание аналоговых блоков и схем PLL выведено на отдельные выводы для уменьшения помех, создаваемых работой других блоков. На данные выводы должно подаваться напряжение из того же источника, что и U_{CC}, но при этом на печатной плате должны быть применены меры по снижению помех.

GND выводы: Основные выводы «Общий» питания.

AGND выводы: Выводы «Общий» аналогового питания AU_{CC}. Данные выводы должны соединяться с GND, но при этом на печатной плате должны быть применены меры по снижению помех.

AGND1 выводы: Выводы «Общий» аналогового питания AU_{CC1}. Данные выводы должны соединяться с GND, но при этом на печатной плате должны быть применены меры по снижению помех.

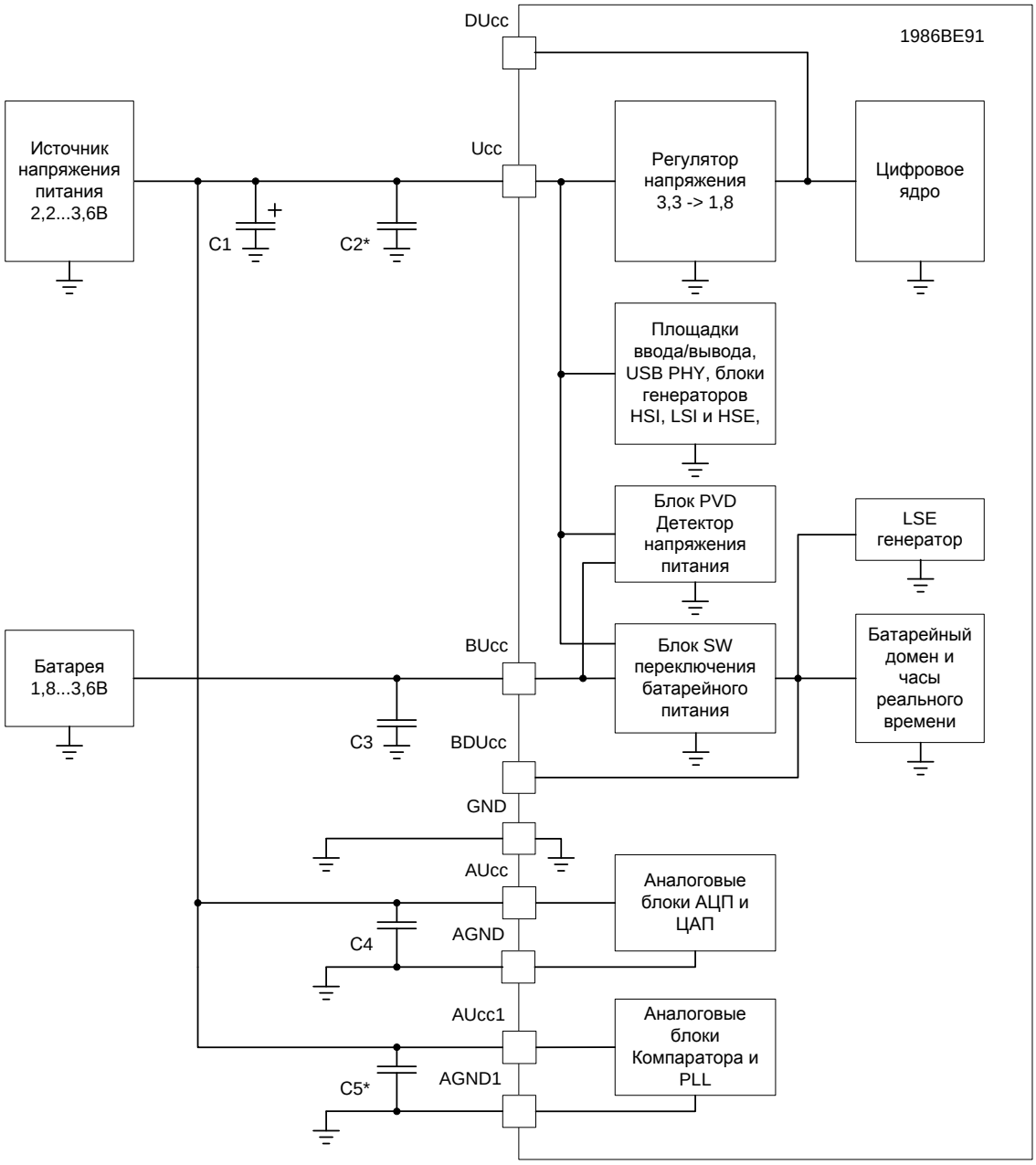


Рисунок 6. Структурная блок-схема подачи питания

Примечания:

1. Конденсаторы должны быть установлены у каждого вывода питания;
2. Конденсатор C1 = 22 мкФ, C2 = C3 = C4 = C5 = 0,1 мкФ;
3. Если не используется батарейное питание, то вывод BUcc должен быть объединен с U_{CC};
4. Если используется интерфейс USB, то напряжение питания U_{CC} должно быть в пределах от 3,0 до 3,6 В;
5. Если используется АЦП или ЦАП, то напряжение питания U_{CC} (AU_{CC} и AU_{CC1}) должно быть в пределах от 2,4 до 3,6 В.

Микроконтроллер имеет несколько режимов энергопотребления, подробнее см. разделы «Сигналы тактовой частоты» и «Управление электропитанием». Микроконтроллер имеет встроенный детектор напряжения питания, подробнее см. раздел «Детектор напряжения питания».

6.1 Схема сброса при включении и выключении основного питания

При включении питания вырабатывается внутренний сигнал сброса POR для цифровой части, питание U_{CC} нарастает и, пока оно не превысило уровень 2,0 В, сигнал сброса POR удерживается; после превышения данного уровня сигнал POR выдается еще на протяжении ~ 4 мс для того, чтобы гарантировано установилось напряжение питания, после чего сигнал POR снимается, и схема может начать работать.

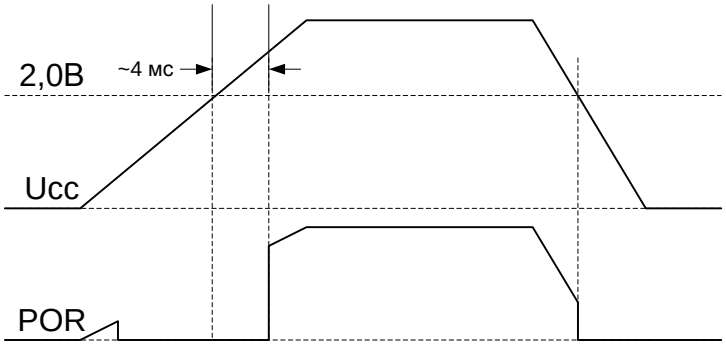


Рисунок 7. Сигнал сброса при включении и выключении основного напряжения питания

При снижении напряжения питания U_{CC} ниже уровня 2,0 В сигнал POR вырабатывается без задержки.

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

При включении основного напряжения питания U_{CC} автоматически включается встроенный регулятор напряжения для формирования напряжения DU_{CC} питания цифрового ядра. В ходе работы микроконтроллера встроенный регулятор может быть отключен. Подробнее см. в разделе «Система синхронизации и энергопотребление».

Микроконтроллер также может быть установлен в начальное состояние 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.

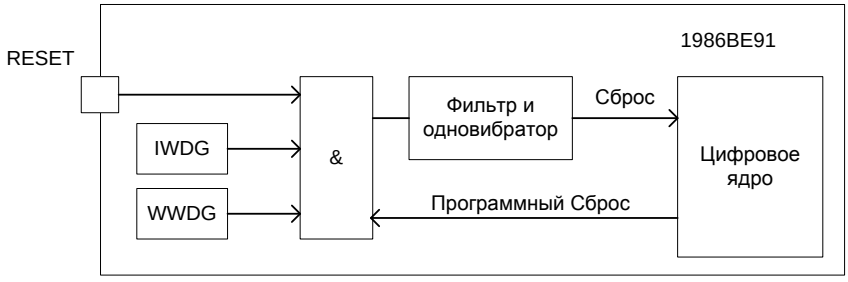


Рисунок 8. Структурная блок-схема сброса

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

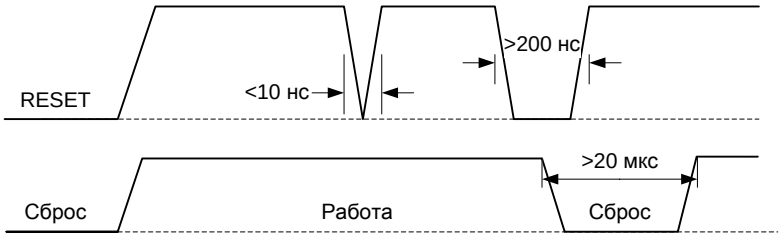


Рисунок 9. Формирование сигнала сброса

7 Организация памяти

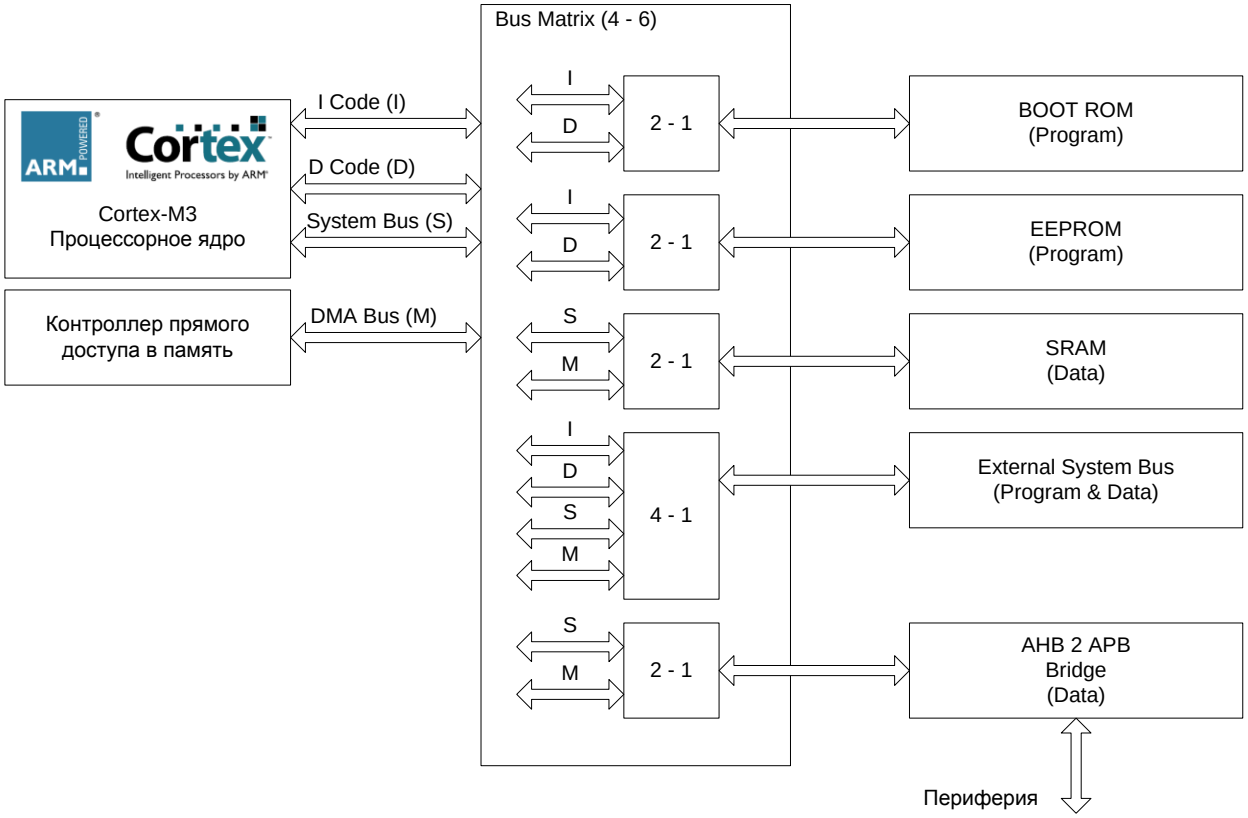


Рисунок 10. Структурная схема организации памяти

Процессорное ядро имеет три системных шины:

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

Также в микроконтроллере реализован контроллер прямого доступа в память (DMA), который осуществляет выборку через шину DMA Bus.

Все адресное пространство микроконтроллера едино и имеет максимальный объем 4 Гбайт. В данное адресное пространство отображаются различные модули памяти и периферии.

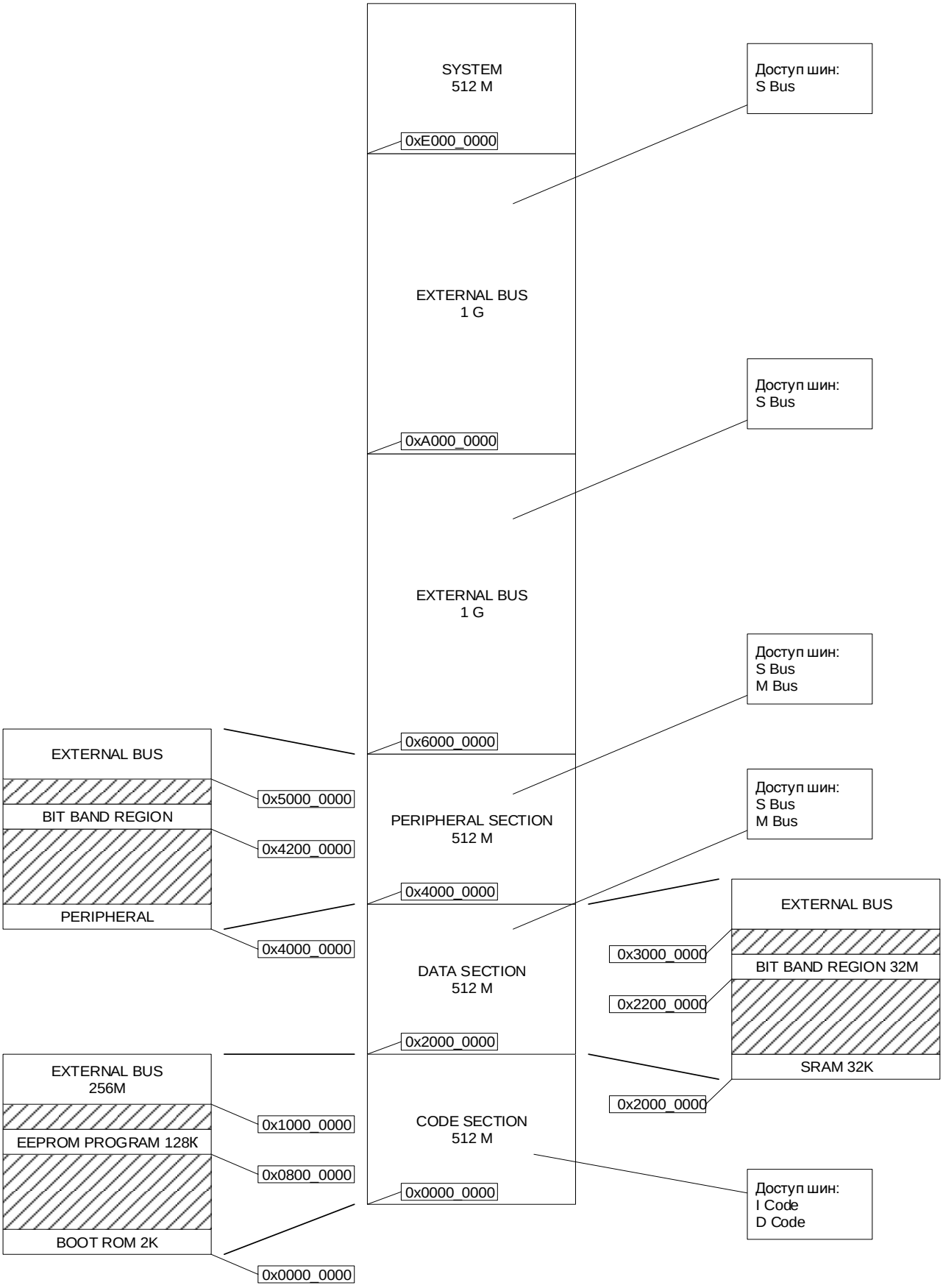


Рисунок 11. Карта распределения основных областей памяти

7.1 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 и предназначена для хранения кода программ во внешних микросхемах памяти, подсоединенных к внешней системной шине.

7.2 Секция DATA

Область Internal SRAM (Data):

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

Область BIT BAND REGION TO SRAM (Data):

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

Запись слова в область alias-региона имеет тот же эффект, что и операция чтение-модификация-запись для целевого бита в области bit-band региона. Бит [0] записываемого значения в область alias-региона определяет значение, которое будет записано в целевой бит в области bit-band региона. Запись значения с установленным битом [0] записывает 1 в целевой бит bit-band региона, а запись значения с очищенным битом [0] записывает 0 в целевой бит bit-band региона. Биты [31:1] записываемого слова в область alias-региона не влияют на целевой бит bit-band региона. Запись 0x01 имеет тот же эффект, что и запись 0xFF. Запись 0x00 имеет тот же эффект, что и запись 0xFE. При чтении слова из области alias-региона будет возвращено значение 0x01 или 0x00. Биты [31:1] равны нулю. Значение 0x01 указывает, что целевой бит в области bit-band региона установлен. Значение 0x00 указывает, что целевой бит в области bit-band региона сброшен.

Область EXTERNAL BUS:

Область отображения внешней системной шины в адресное пространство области данных и предназначена для хранения данных во внешних микросхемах памяти, подсоединенных к внешней системной шине.

7.3 Секция PERIPHERAL

Область PERIPHERAL (Data):

Область отображения регистров периферии в общее адресное пространство памяти.

Область BIT BAND REGION TO PERIPHERAL (Data):

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

Область EXTERNAL BUS:

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

7.4 Секция EXTERNAL RAM

Область EXTERNAL BUS:

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

7.5 Секция SYSTEM

Предназначена для отображения системных регистров ядра и системной периферии.

7.6 Блок BUS MATRIX

Блок BUS MATRIX предназначен для переключения системных шин I Code, D Code, System Bus и DMA Bus между различными областями памяти. 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 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. Запрос с меньшим приоритетом задерживается до окончания выполнения 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.

7.7 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.

7.8 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 Flash-памяти программ».

7.9 Память SRAM

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

7.10 Регионы памяти, типы и атрибуты

Карта памяти и модуль защиты памяти (MPU) разбивают всё 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 см. раздел «Модуль защиты памяти».

По отношению к порядку выполнения обращений к памяти различаются следующие типы памяти:

- Normal
- Device
- Strongly-ordered («Строго упорядоченная»)

Normal

Процессор может переопределить последовательность обращений для большей эффективности или чтобы выполнить опережающее чтение.

Device

Процессор сохраняет последовательность обращений по отношению к другим обращениям к памяти типов Device или Strongly-ordered.

Strongly-ordered

Процессор сохраняет последовательность обращений по отношению ко всем другим обращениям.

Различие в требованиях к памяти типов Device и Strongly-Ordered состоит в том, что система памяти может буферизировать запись в память типа Device, но буферизация записи в память типа Strongly-ordered не допускается.

Дополнительные атрибуты памяти:

- Shareable («Допускающая совместное использование»)
- Execute Never или XN («Не выполнять»)

Shareable

Для регионов с атрибутом Shareable система памяти обеспечивает синхронизацию между различными устройствами управления передачей данных по шине при наличии нескольких таких устройств, например, процессор и контроллер DMA.

Память типа Strongly-ordered всегда имеет атрибут Shareable.

Если несколько устройств управления передачей данных по шине могут обращаться к региону, не обладающему атрибутом Shareable, непротиворечивость данных между такими устройствами должна быть гарантирована программным обеспечением.

Execute Never (XN)

Атрибут указывает на запрет обращения к командам (инструкциям). Любая попытка извлечь инструкцию из XN региона приведет к исключению типа “Memory Management Fault”.

7.11 Последовательность обращений к памяти

Для большинства обращений к памяти, инициируемых явно командами обращения к памяти, если только это не затрагивает поведения командной последовательности, система памяти не гарантирует, что порядок, в котором выполняются эти обращения, совпадает с порядком, в котором соответствующие команды следуют в программе.

Обычно, если правильное выполнение программы требует, чтобы два обращения память произошли в порядке, заданном программой, то между командами обращения к памяти в программе должна быть вставлена инструкция барьерной синхронизации (memory barrier instruction), см. раздел «Программное упорядочение обращений к памяти».

Впрочем, система памяти гарантирует некоторый порядок доступа в регионы памяти Device и Strongly-ordered. Для двух команд обращения к памяти, A1 и A2, если A1 следует перед A2 в коде программы, последовательность обращений к памяти, вызванных этими двумя командами, будет такой, как показывает Таблица 5.

Таблица 5 – Последовательность обращений инструкций к памяти

A1 \ A2	Normal	Device		Strongly-ordered
		“non-shareable”	shareable	
Normal	-	-	-	-
Device, “non-shareable”	-	<	-	<
Device, shareable	-	-	<	<
Strongly-ordered	-	<	<	<

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

Типы памяти Normal, Device, Strongly-ordered и атрибут Shareable см. «Регионы памяти, типы и атрибуты»; “non-shareable” обозначает регион, не обладающий атрибутом Shareable.

7.12 Поведение обращений к памяти

Особенности доступа к разным областям памяти показывает Таблица 6.

Таблица 6 – Поведение обращений к памяти

Адресный диапазон	Секция памяти	Тип памяти	XN	Описание
0x00000000–0x1FFFFFFF	Code	Normal	-	Область памяти для кода программы, данные также могут храниться здесь.
0x20000000–0x3FFFFFFF	SRAM	Normal	-	Область памяти для данных. Код программы также может располагаться здесь. Эта секция содержит области bit-band доступа
0x40000000–0x5FFFFFFF	Peripheral	Device	XN	Эта секция содержит области bit-band доступа
0x60000000–0x9FFFFFFF	External RAM	Normal	-	Область памяти для данных и кода
0xA0000000–0xDFFFFFFF	External Device	Device	XN	Область памяти для внешних устройств
0xE0000000–0xE00FFFFF	Private Peripheral	Strongly-ordered	XN	Эта секция содержит регистры NVIC, системный таймер и регистры блока

	Bus			управления ядра
0xE0100000– 0xFFFFFFFF	Резерв	Device	XN	Зарезервировано

Секции Code, SRAM и External RAM могут содержать код программы. Однако рекомендуется для программы использовать секцию Code, так как процессор имеет отдельные шины доступа к этой секции, что позволяет одновременно выполнять выборку инструкций и данных.

Блок MPU может изменить описанное здесь стандартное поведение при доступе в память. Подробности см. раздел «Модуль защиты памяти».

7.12.1 Дополнительные условия доступа к совместно используемой памяти

Если система содержит совместно используемую память, следует учитывать для некоторых регионов дополнительные условия доступа, и для некоторых регионов их собственное разбиение, как это показывает Таблица 7.

Таблица 7 – Дополнительные условия совместного использования памяти

Адресный диапазон	Секция памяти	Тип памяти	Возможность совместного использования	
0x00000000– 0x1FFFFFFF	Code	Normal	-	
0x20000000– 0x3FFFFFFF	SRAM	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	-	

Типы памяти Normal, Device, Strongly-ordered и атрибут Shareable см. «Регионы памяти, типы и атрибуты»; “non-shareable” обозначает регион, не обладающий атрибутом Shareable.

7.13 Программное упорядочение обращений к памяти

Последовательность расположения инструкций (команд) в потоке программы не всегда гарантирует последовательность соответствующих обращений к памяти; это происходит потому, что:

- процессор может переменить порядок следования некоторых обращений для увеличения производительности при условии, что при этом не изменяется общее поведение программы;
- процессор имеет несколько интерфейсов для обращений к памяти;
- 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 к памяти. Если порядок обращений к памяти критичен, программный код должен содержать инструкции барьерной синхронизации, чтобы обеспечить нужный порядок. Процессор предлагает следующие инструкции (команды) барьерной синхронизации:

DMB

Инструкция Data Memory Barrier (DMB) позволяет быть уверенным, что все выполняемые обращения к памяти будут завершены до следующего обращения к памяти. Смотрите описание инструкции DMB.

DSB

Инструкция Data Synchronization Barrier (DSB) позволяет быть уверенным, что все выполняемые обращения к памяти будут завершены до начала выполнения следующей команды (инструкции). Смотрите 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.

7.14 Bit-band регионы

Механизм bit-band отображает каждый бит bit-band региона в слово в bit-band alias регионе. Bit-band регион занимает младший 1 Мбайт области SRAM и области периферийных устройств (Peripheral). Им соответствуют области по 32 Мбайта bit-band alias, как это показывает Таблица 8.

Таблица 8 – Описание bit-band регионов

Адресный диапазон	Регион памяти	Доступ к инструкциям и данным
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.

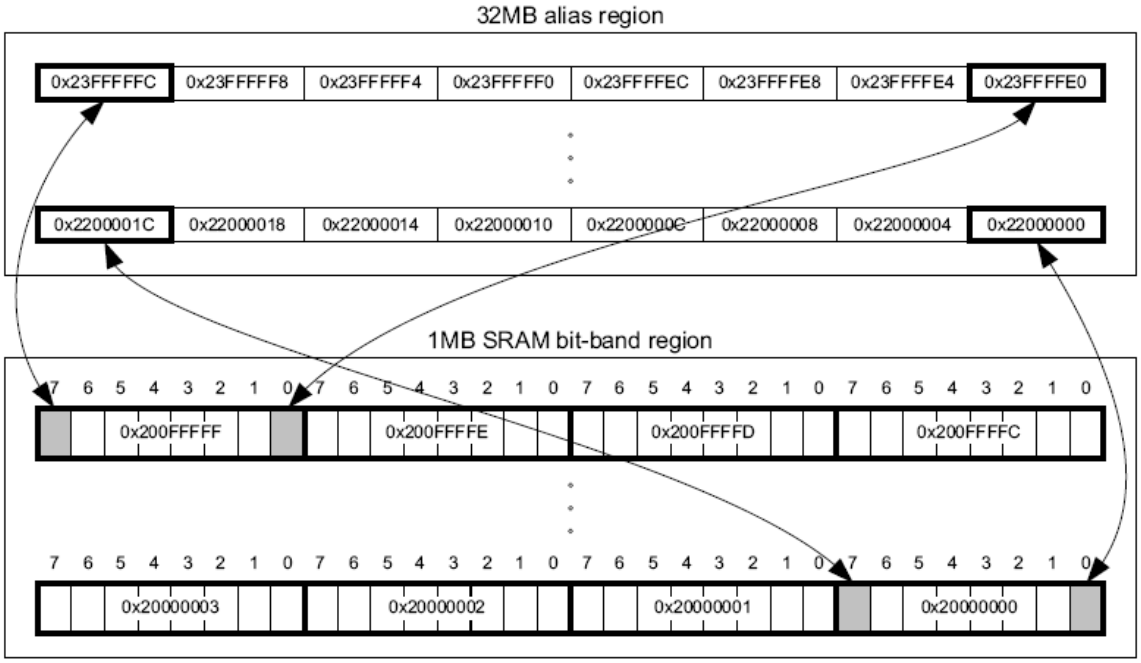


Рисунок 12. Схема отображения региона bit-band alias в регионе bit-band region

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 хранится по меньшему адресу, например:

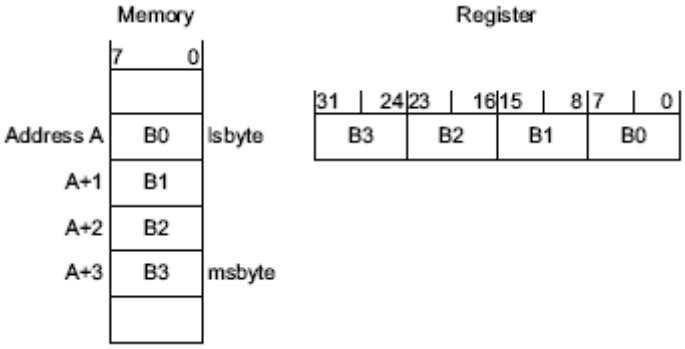


Рисунок 13. Организация расположения байтов в 32-х битной памяти

7.15 Примитивы синхронизации

Система команд Cortex-M3 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.

7.15.1 Команда Load-Exclusive

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

7.15.2 Команда Store-Exclusive

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

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

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

Команды Load Exclusive и Store-Exclusive образуют следующие пары:

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

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

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

- инструкцией Load-Exclusive считать значение из памяти;
- изменить значение;
- инструкцией Store-Exclusive попытаться записать новое значение обратно в память; проверить возвращаемый статусный бит.
  Если этот бит - 0, то процедура «чтение-модификация-запись» выполнена успешно.
  Если этот бит - 1, то запись не была выполнена. Это означает, что значение, считанное первоначально, возможно устарело и программа должна повторить цикл «чтение-модификация-запись».

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

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

Ядро Cortex-M3 имеет монитор эксклюзивных доступов, который отмечает, что процессор выполнил команду Load-Exclusive. Если процессор является частью многопроцессорной системы, то система также отмечает на глобальном уровне адреса памяти, для которых имел место эксклюзивный доступ со стороны каждого процессора.

Процессор удаляет свою отметку об эксклюзивном доступе в тех случаях, когда:

- процессор выполняет команду CLREX;

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

В многопроцессорной реализации:

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

Подробнее см. описание инструкций LDREX, STREX и CLREX.

7.16 Указания по программированию примитивов синхронизации

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

Таблица 9 – Встроенные функции для создания инструкций эксклюзивного доступа

Инструкции	Функции
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 зависят от типа данных указателя, передаваемого встроенной функции. Например, следующий код создаст инструкцию LDREXB:

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

7.17 Базовые адреса процессора

Таблица 10 – Базовые адреса процессора

Адрес	Размер	Блок		Примечание
Память программ				
0x0000_0000	1 Кбайт	BOOT ROM		Загрузочная программа
0x0800_0000	128 Кбайт	EEPROM		Область Flash-памяти программ с пользовательской программой
0x1000_0000	256 Мбайт	EXTERNAL BUS		Область доступа к внешней системной шине
Память данных				
0x2000_0000	32Кбайт	SYSTEM RAM		Область внутреннего ОЗУ
0x2200_0000	16 Мбайт	SYSTEM RAM Bit Band Region		Область битового доступа внутреннего ОЗУ
0x3000_0000	256 Мбайт	EXTERNAL BUS		Область доступа к внешней системной шине
Периферия				
0x4000_0000	1536 байт	0	CAN1	Регистры контроллера интерфейса CAN1
0x4000_8000	1536 байт	1	CAN2	Регистры контроллера интерфейса CAN2
0x4001_0000	904 байт	2	USB	Регистры контроллера интерфейса USB
0x4001_8000	20 байт	3	EEPROM_CNTRL	Регистры контроллера Flash-памяти программ
0x4002_0000	48 байт	4	RST_CLK	Регистры контроллера сигналов тактовой частоты
0x4002_8000	80 байт	5	DMA	Регистры контроллера прямого доступа в память
0x4003_0000	72 байт	6	UART1	Регистры контроллера интерфейса UART1
0x4003_8000	72 байт	7	UART2	Регистры контроллера интерфейса UART2
0x4004_0000	36 байт	8	SPI1	Регистры контроллера интерфейса SSP1
0x4004_8000	-	9	-	
0x4005_0000	28 байт	10	I2C1	Регистры контроллера интерфейса I2C1
0x4005_8000	4 байт	11	POWER	Регистры детектора напряжения питания
0x4006_0000	12 байт	12	WWDT	Регистры контроллера сторожевого таймера WWDT
0x4006_8000	16 байт	13	IWDT	Регистры контроллера сторожевого таймера IWDT
0x4007_0000	128 байт	14	TIMER1	Регистры управления Таймер 1
0x4007_8000	128 байт	15	TIMER2	Регистры управления Таймер 2
0x4008_0000	128 байт	16	TIMER3	Регистры управления Таймер 3
0x4008_8000	48 байт	17	ADC	Регистры управления АЦП
0x4009_0000	12 байт	18	DAC	Регистры управления ЦАП
0x4009_8000	12 байт	19	COMP	Регистры управления Компаратора
0x400A_0000	36 байт	20	SPI2	Регистры контроллера интерфейса SSP2
0x400A_8000	32 байт	21	PORTA	Регистры управления порта A
0x400B_0000	32 байт	22	PORTB	Регистры управления порта B
0x400B_8000	32 байт	23	PORTC	Регистры управления порта C
0x400C_0000	32 байт	24	PORTD	Регистры управления порта D
0x400C_8000	32 байт	25	PORTE	Регистры управления порта E
0x400D_0000	-	26	-	
0x400D_8000	84 байт	27	BKP	Регистры доступа и управления батарейным

				доменом
0x400E_0000	-	28	-	
0x400E_8000	32 байт	29	PORTF	Регистры управления порта F
0x400F_0000	88 байт	30	EXT_BUS_CNTRL	Область доступа к внешней системной шине
0x400F_8000	-	31	-	
0x4200_0000	16 Мбайт	PERIPHERAL Bit Band Region		Область битового доступа к регистрам периферии
0x5000_0000	256 Мбайт	EXTERNAL BUS		Область доступа к внешней системной шине
Внешняя системная шина				
0x6000_0000	1 Гбайт	EXTERNAL BUS		Область доступа к внешней системной шине
0xA000_0000	1 Гбайт	EXTERNAL BUS		Область доступа к внешней системной шине
SYSTEM REGION				
0xE000_0000	256 Мбайт			Системные регистры процессора ARM Cortex-M3

8 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 шине «Общий» (встроенные резисторы подтяжки к шине «Общий» 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 11 – Режимы первоначального запуска микроконтроллера

MODE[2:0]	Режим	Стартовый адрес / таблица векторов прерываний	Описание
000	Микроконтроллер в режиме отладки	0x0800_0000	Процессор начинает выполнять программу из внутренней Flash-памяти программ. При этом установлен отладочный интерфейс JTAG_B
001	Микроконтроллер в режиме отладки	0x0800_0000	Процессор начинает выполнять 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 выполнять 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 выполнять программу из внешней памяти, установленной на внешней системной шине. При этом отладочный интерфейс JTAG/SW заблокирован
100	Зарезервировано	–	-
101	UART загрузчик	Определяется пользователем	Микроконтроллер через интерфейс UART2 на выводах PD[1:0] получает код программы в ОЗУ для исполнения
110	UART загрузчик	Определяется пользователем	Микроконтроллер через интерфейс UART2 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 10К с учетом, чтобы эти подтяжки не влияли на работу системы.

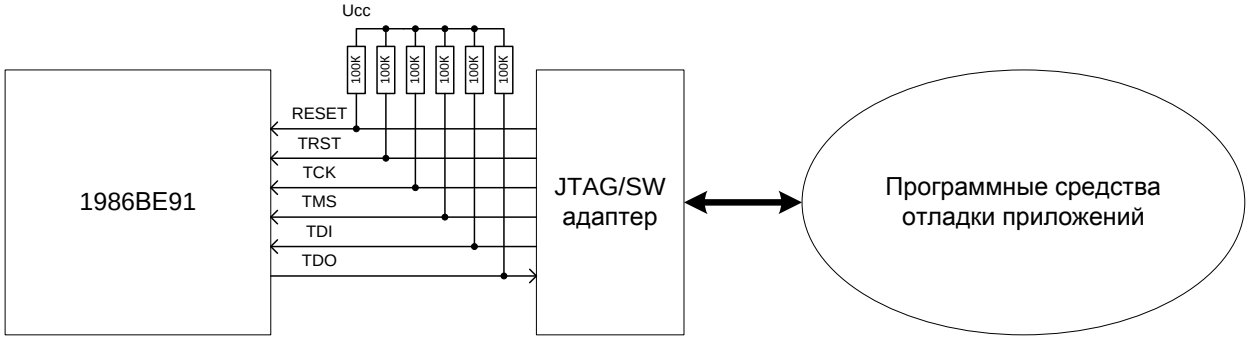


Рисунок 14. Схема работы в режиме отладки

В отладочном режиме можно:

- стирать, записывать, считывать внутреннюю Flash-память программ;
- считывать и записывать содержимое ОЗУ, 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, как это показывает Таблица 12.

Таблица 12 – Переопределение выводов интерфейса JTAG/SW

Вывод JTAG/SW	Вывод микроконтроллера	Описание
JTAG_A		
TRST	PB4/JA_TRST	В качестве выводов интерфейса используются выводы порта B, совмещенные с выводами данных внешней системной шины, выводами таймера 3, выводами UART1, UART2 и CAN1, использование которых при отладке запрещено
TCK	PB2/JA_TCK	
TMS	PB1/JA_TMS	
TDI	PB3/JA_TDI	
TDO	PB0/JA_TDO	
JTAG_B		
TRST	PD2/JB_TRST	В качестве выводов интерфейса используются выводы порта D, совмещенные с каналами АЦП, выводами каналов Таймера 1 и 3, UART2 и SSP2, использование которых при отладке запрещено
TCK	PD1/JB_TCK	
TMS	PD0/JB_TMS	
TDI	PD3/JB_TDI	
TDO	PD4/JB_TDO	

8.1 UART загрузчик

В режиме UART загрузчика используют один и тот же периферийный модуль UART2, один и тот же протокол обмена, но различные порты ввода/вывода, как это показывает Таблица 13.

Таблица 13 – Используемые порты ввода/вывода UART загрузчиком

Режим MODE[2:0]	TX	RX
101	PD[1]	PD[0]
110	PF[1]	PF[0]

Данные режимы предоставляют 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. Помимо доступа к ОЗУ, может быть осуществлен доступ и к другим адресным диапазонам (EEPROM, ROM, Периферия).

В качестве источника тактовой частоты UART2 используется внутренний RC-генератор HSI с частотой 8 МГц. Так как имеется разброс значений частоты HSI, то требуется этап подбора значения делителя частоты UART2 для синхронизации с внешним устройством.

8.1.1 Параметры связи по UART

Для связи по UART выбраны следующие параметры канала связи:

- начальная скорость – 9600 Бод;
- количество бит данных – 8;
- четность – нет;
- количество Stop бит – 1;
- загрузчик не использует FIFO UART2;
- загрузчик всегда выступает в качестве Slave, а внешнее устройство, подающее команды, – в качестве Master;
- данные передаются младшим битом вперед.

8.1.2 Протокол обмена по UART

После синхронизации с внешним устройством, подающим команды (Master), загрузчик переходит в диспетчер команд.

Таким образом, внешнему устройству доступны следующие команды:

Таблица 14 – 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

8.1.3 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, чтобы минимизировать ошибки обмена. Как только Загрузчик настроил скорость, он переходит в диспетчер команд и выдает приглашение (3 байта 0x0D (перевод строки), 0x0A (возврат каретки), 0x3E ('>'),) Master-у.

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

8.1.4 Команда CMD_SYNC

Пустая команда.

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

Таблица 15 – 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

8.1.5 Команда CMD_CR

Выдача приглашения Master-у

Таблица 16 – 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 '>')

8.1.6 Команда CMD_BAUD

Установка скорости обмена

Таблица 17 – Команда CMD_BAUD

	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

8.1.7 Команда CMD_LOAD

Загрузка массива байт в память микроконтроллера

Таблица 18 – 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')

8.1.8 Команда CMD_VFY

Выдача массива байт из памяти микроконтроллера

Таблица 19 – 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')

8.1.9 Команда CMD_RUN

Запуск программы на выполнение.

Таблица 20 – 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

8.1.10 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.

8.1.11 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.

9 Контроллер Flash-памяти программ MDR_EEPROM

Микроконтроллер содержит встроенную Flash-память программ с 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.

9.1 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 чтобы процессорное ядро могло получать новые инструкции на больших частотах, в микроконтроллере реализуется Flash-память с физической организацией 8К на 128 разрядов (4 сектора 8К на 32 разряда). Данные из различных секторов чередуются в линейном адресном пространстве. Таким образом, за 40 нс из Flash-памяти извлекается 16 байт, в которых может быть закодировано от 4 до 8 последовательных инструкций процессора. И пока ядро выполняет эти инструкции, из памяти извлекается следующая порция данных. Таким образом, тактовая частота может превышать частоту 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, чтобы данные успели считаться из Flash. Число тактов паузы зависит от тактовой частоты процессора; так при работе с частотой ниже 25 МГц пауза не требуется, поскольку 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]. Таблица 21 показывает характеристики необходимой паузы для работы Flash-памяти программ.

Таблица 21 – 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 80 МГц не гарантируется
0x04	4	До 125 МГц	Установлено по умолчанию после сброса
0x05	5	До 150 МГц	
0x06	6	До 175 МГц	
0x07	7	До 200 МГц	

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

9.2 Работа Flash-памяти программ в режиме программирования

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

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

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

Page 31 256 x 128 4K x 8	0x0801_FFFC … 0x0801_F00C	0x0801_FFF8 … 0x0801_F008	0x0801_FFF4 … 0x0801_F004	0x0801_FFF0 … 0x0801_F000
	…	…	…	…
Page 1 256 x 128 4K x 8	0x0800_1FFC … 0x0800_100C	0x0800_1FF8 … 0x0800_1008	0x0800_1FF4 … 0x0800_1004	0x0800_1FF0 … 0x0800_1000
Page 0 256 x 128 4K 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 256 x 32 1K x 8	Sector_C 256 x 32 1K x 8	Sector_B 256 x 32 1K x 8	Sector_A 256 x 32 1K x 8

Основная память (IFREN=0)

Page 0 256 x 128 4K 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 256 x 32 1K x 8	Sector_C 256 x 32 1K x 8	Sector_B 256 x 32 1K x 8	Sector_A 256 x 32 1K x 8

Информационная память (IFREN=1)

Рисунок 15. Структура памяти Flash

9.2.1 Стирание всей памяти

Стирание всей памяти выполняется в 4 этапа:

1 этап - стирание Sector_A для всей памяти;
2 этап - стирание Sector_B для всей памяти;

3 этап - стирание Sector_C для всей памяти;
4 этап - стирание Sector_D для всей памяти.

Стирание одного сектора памяти возможно только в режиме программирования. Для стирания одного сектора памяти надо установить необходимое значение в бит IFREN (1 – для основной и информационной памяти и 0 – для основной памяти), и номер сектора EEPROM_ADR[3:2] (00 – Sector_A, 01 – Sector_B, 10 – Sector_C и 11 – Sector_D) затем установить биты XE, MAS1 и ERASE в 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 памятью можно выполнять спустя время trcv = 1 мкс. Временная диаграмма стирания памяти представлена далее (см. Рисунок 16). При стирании информационной области, автоматически стирается и основная. Для стирания всей памяти необходимо осуществить стирание всех секторов памяти.

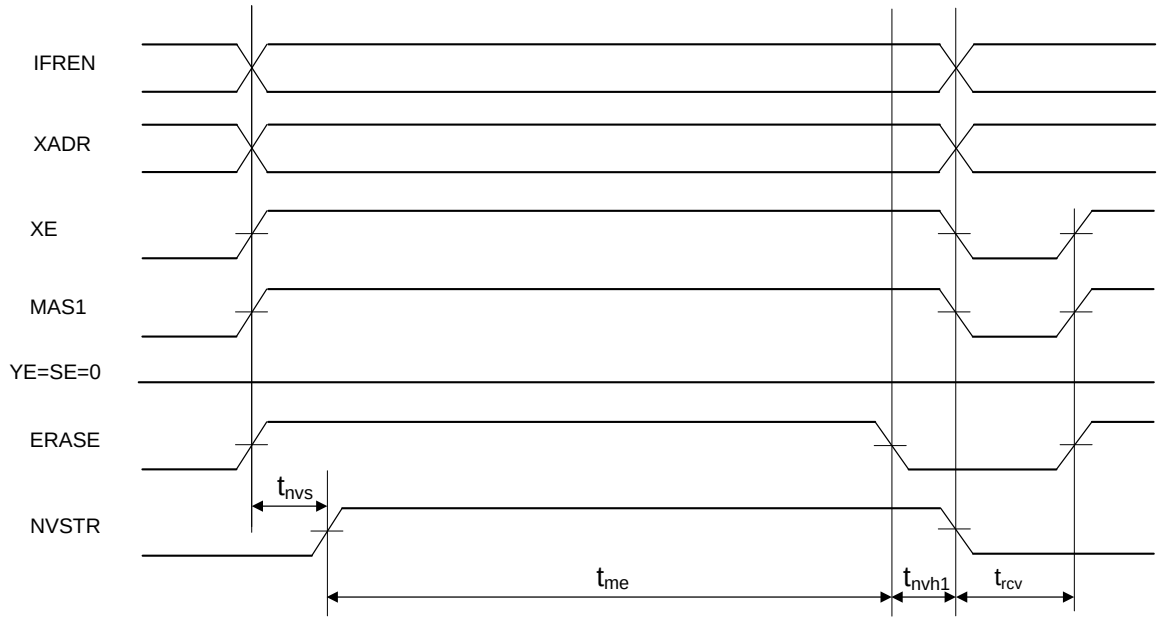


Рисунок 16. Временная диаграмма стирания памяти

9.2.2 Стирание банка памяти одной страницы размером 4 Кбайт

Стирание одной страницы выполняется в 4 этапа:

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

Стирание одного сектора страницы памяти возможно только в режиме программирования. Для стирания страницы памяти надо установить необходимое значение в бит IFREN (1 - для информационной памяти и 0 - для основной памяти), затем установить адрес стираемой страницы в регистре EEPROM_ADR[16:12] и номер сектора EEPROM_ADR[3:2] (00 – Sector_A, 01 – Sector_B, 10 – Sector_C и 11 – Sector_D) и установить биты XE и ERASE в 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 далее (см. Рисунок 17). Для стирания всей страницы необходимо осуществить стирание всех секторов страницы.

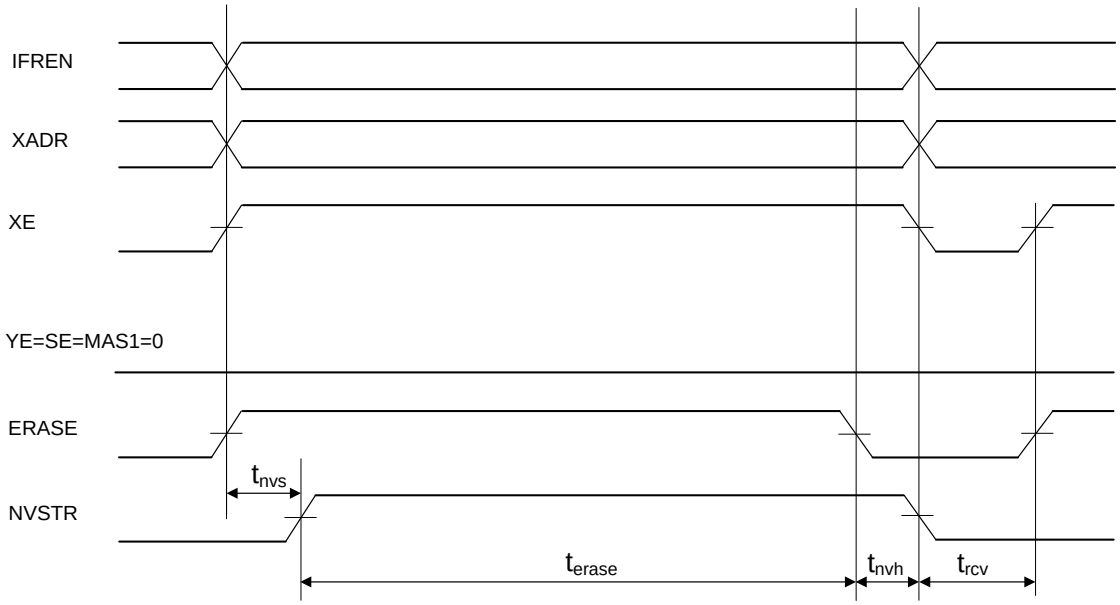


Рисунок 17. Временная диаграмма стирания банка памяти

9.2.3 Запись 32-х битного слова в память

Запись в память возможна 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 поместить 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 длится время tprog = 40 мкс. Спустя это время необходимо очистить бит YE, и через tadh = 20 нс установить новый адрес (в пределах того же сектора и той же страницы) и значение для записи в другую ячейку памяти; затем через tads = 20 нс установить YE в единицу и записать следующее слово.

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

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 памятью можно выполнять спустя время trcv = 1 мкс.

Временная диаграмма записи памяти представлена ниже (см. Рисунок 18).

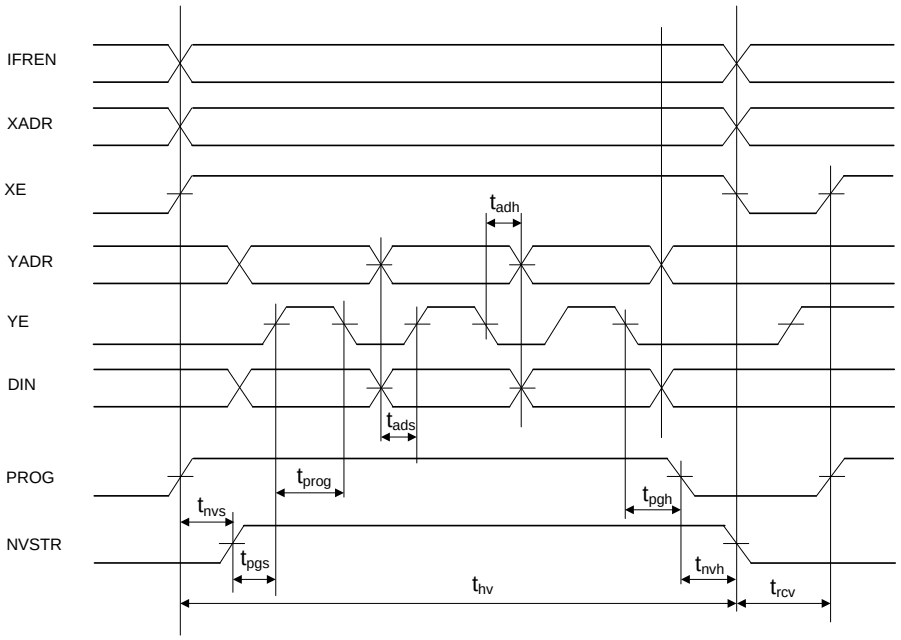


Рисунок 18. Временная диаграмма записи памяти

9.2.4 Чтение 32-х битного слова из памяти

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

В режиме программирования для чтения доступна и основная, и информационная память. Для чтения из памяти надо установить необходимое значение в бит IFREN (1 – для информационной памяти и 0 – для основной памяти), затем установить адрес, из которого необходимо считать данные, в регистре EEPROM_ADR и установить биты XE, YE и SE в единицу, и спустя время txa = 30 нс из регистра EEPROM_DO можно считать данные. Если необходимо считать следующее слово, то в регистр EEPROM_ADR следует записать новый адрес и, спустя время 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 в режиме программирования представлена далее (см. Рисунок 19).

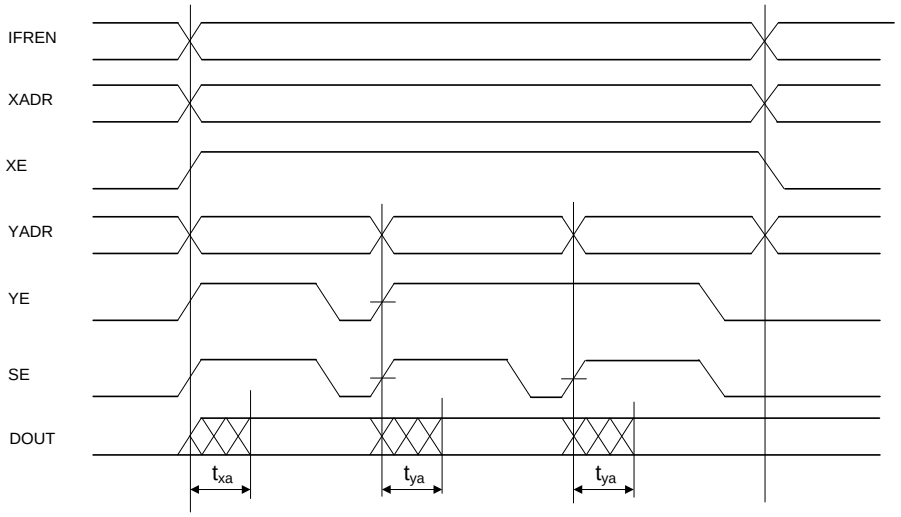


Рисунок 19. Временная диаграмма чтения памяти

Flash-память программ поддерживает до 10 000 циклов перезаписи. Нельзя повторять циклы стирания – записи и стирания – стирания одной ячейки памяти с периодом менее 4 мс.

9.3 Регистры управления контроллера Flash-памяти программ

Таблица 22 предоставляет перечень регистров управления контроллера Flash-памяти программ.

Таблица 22 – 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

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.

9.3.1 MDR_EEPROM->CMD

Таблица 23 – Регистр команды EEPROM_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	U	R/W
Сброс	0	0	0	0	100	0	0
	IFREN	**SE**	**YE**	**XE**	**Delay[2:0]**		**CON**

Таблица 24 – 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 EEPROM_DI: 0 – нет записи; 1 – есть запись
11	MAS1	При ERASE =1: 0 – стирается страница с адресом ADR[16:12] в секторе с адресом ADR[3:2]; 1 – стирается весь сектор с адресом ADR[3:2]
10	ERASE	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, 1	-	Зарезервировано
0	CON	Переключение контроллера памяти EEPROM на регистровое управление, не может производиться при исполнении программы из области EEPROM: 0 – управление EEPROM от ядра, рабочий режим; 1 – управление от регистров, режим программирования

9.3.2 MDR_EEPROM->ADR

Таблица 25 – Регистр адреса EEPROM_ADR

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

Таблица 26 – 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

9.3.3 MDR_EEPROM->DI

Таблица 27 – Регистр записываемых данных EEPROM_DI

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

Таблица 28 – Описание бит регистра записываемых данных EEPROM_DI

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

9.3.4 MDR_EEPROM->DO

Таблица 29 – Регистр считываемых данных EEPROM_DO

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

Таблица 30 – Описание бит регистра считываемых данных EEPROM_DO

№	**Функциональное**	**Расшифровка функционального имени бита, краткое**

бита	имя бита	описание назначения и принимаемых значений
31...0	DATA[31:0]	Данные, считанные из EEPROM

9.3.5 MDR_EEPROM->KEY

Таблица 31 – Регистр ключа EEPROM_KEY

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

Таблица 32 – 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 EEPROM_KEY записать комбинацию `0x8AAA5551`

10 Процессорное ядро ARM Cortex-M3

Описание процессора Cortex-M3 и периферии ядра:

- процессор Cortex-M3 – высокопроизводительный 32-х разрядный процессор, разработанный для микроконтроллерных систем;
- высокая производительность скомбинирована с быстрой обработкой прерываний;
- расширенная система отладки с точками останова и трассировкой;
- эффективное процессорное ядро для работы системы и памяти;
- сверхнизкое потребление с встроенными режимами Sleep;
- защищенная система с интегрированными блоком защиты памяти MPU.

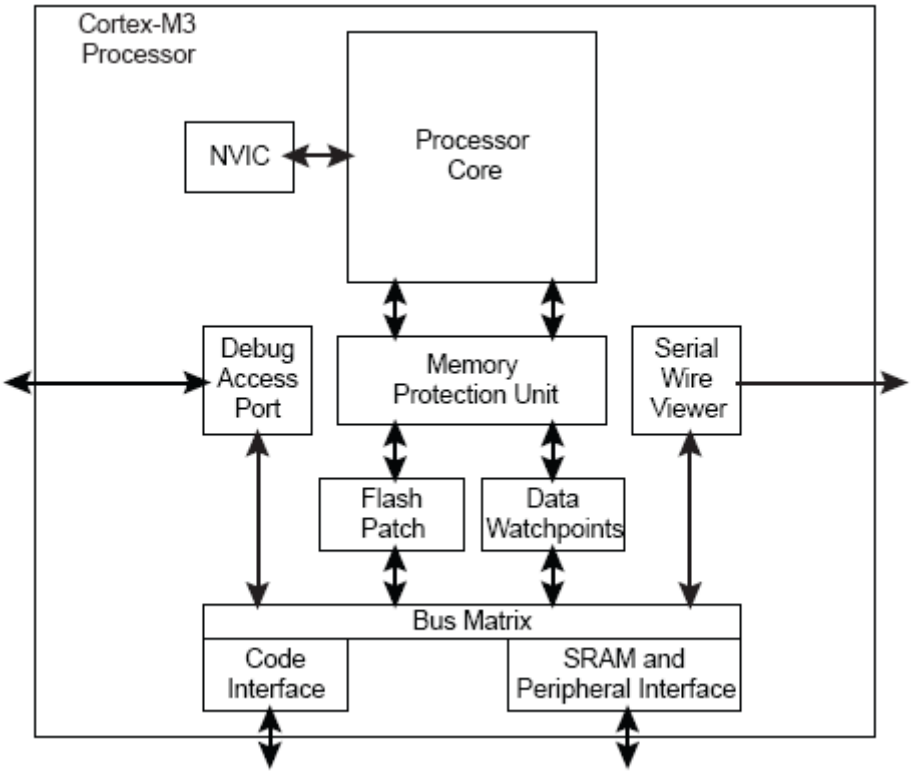


Рисунок 20. Структурная блок-схема процессорного ядра Cortex-M3

Процессор Cortex-M3 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 Cortex-M3 обеспечивает производительность 32-х битных архитектур с размером кода, сравнимым с 8-ми и 16-битными микроконтроллерами.

Процессор содержит контроллер прерываний NVIC, обеспечивающий высокоскоростную обработку прерываний. NVIC обеспечивает до 8-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 Cortex-M3 обеспечивает большую скорость и низкую задержку обращения в память. Также поддерживаются невыровненные обращения и битовые манипуляции с областью ОЗУ и регистрами периферии.

Ядро Cortex-M3 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 Cortex-M3 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.

10.1 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.

10.2 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 33 – 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

10.3 Регистры ядра

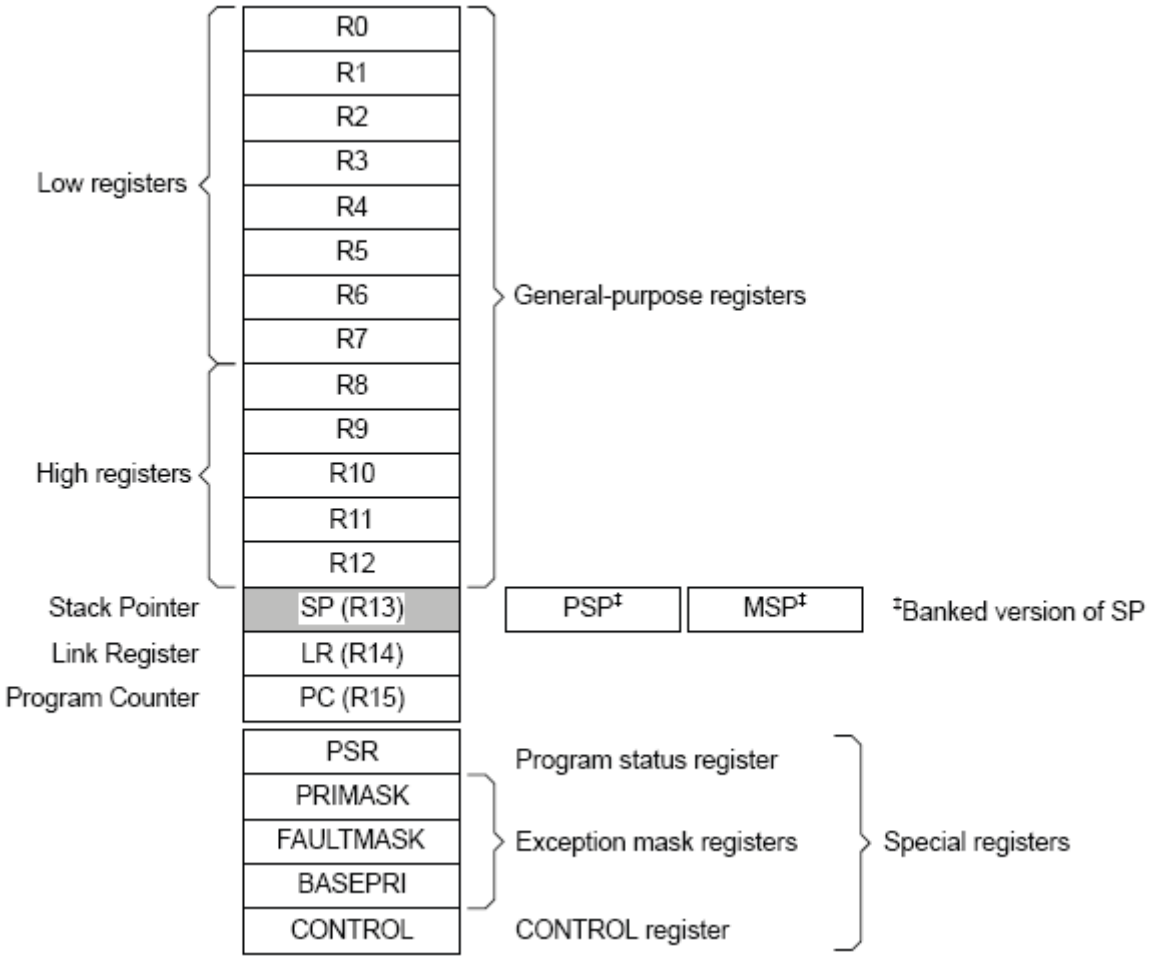


Рисунок 21. Регистры ядра

Таблица 34 – 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

10.3.1 Регистры общего назначения R0-R12

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

10.3.2 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.

10.3.3 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.

10.3.4 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.

10.3.5 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 35 – 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

3. Игнорируется запись в IPSR биты
4. При чтении EPSR бит читаются нули, и запись в них игнорируется.

Подробнее в описании инструкции MRS и MSR.

10.3.6 Программный регистр состояния приложения APSR

Регистр APSR содержит текущие флаги состояния выполнения предыдущей инструкции.

Таблица 36 – 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 37 – 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

10.3.7 Программный регистр состояния прерываний IPSR

Регистр IPSR содержит номер типа исключения для текущего обработчика прерывания.

Таблица 38 – Регистр IPSR

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

Таблица 39 – 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»

10.3.8 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 40 – 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 41 – 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.

10.3.8.1 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.

10.3.8.2 If-Then блок инструкций

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

10.3.9 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.

10.3.10 Регистр маски приоритетов Priority Mask

Регистр PRIMASK предотвращает активацию всех исключений с конфигурируемым приоритетом.

Таблица 42 – Регистр PRIMASK

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

Таблица 43 – 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

10.3.11 Регистр маски сбоев Fault Mask

Регистр FAULTMASK предотвращает активацию всех исключений.

Таблица 44 – Регистр FAULTMASK

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

Таблица 45 – 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.

10.3.12 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 46 – Регистр BASEPRI

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

Таблица 47 – 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 [7:5] этого поля, биты [4:0] читаются как нули и игнорируются при записи. Для более полной информации смотрите Таблица 447 – Регистры приоритета прерываний. Помните, что большее значение в поле приоритета соответствует меньшему приоритету обработчика.

10.3.13 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 48 – Регистр CONTROL

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

Таблица 49 – 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 Privilege Level	0 – привилегирован 1 – непривилегирован

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.

10.4 Исключения и прерывания

Процессорное ядро ARM Cortex-M3 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 «Контроллер прерываний NVIC» для большей 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. Смотрите «Регионы памяти, типы и атрибуты» для большей информации.

11 Система команд

В процессоре реализована версия системы команд Thumb. Поддерживаемые команды представлены ниже (Таблица 50).

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 50 – Система команд процессора Cortex-M3

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

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

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

Мнемокод команды	Операнды	Краткое описание	Флаги	Прим.
STREXB	Rd, Rt, [Rn]	Эксклюзивная запись регистра, байт	-	
STREXH	Rd, Rt, [Rn]	Эксклюзивная запись регистра, полуслово	-	
STRH, STRHT	Rt, [Rn, #offset]	Сохранение регистра, полуслово	-	
STRT	Rt, [Rn, #offset]	Сохранение регистра, слово	-	
SUB, SUBS	{Rd,} Rn, Op2	Вычитание	N,Z,C,V	
SUB, SUBW	{Rd,} Rn, #imm12	Вычитание	N,Z,C,V	
SVC	#imm	Вызов супервизора	-	
SXTB	{Rd,}Rm{,ROR#n}	Преобразовать байт со знаком в слово	-	
SXTH	{Rd,}Rm{,ROR#n}	Преобразовать полуслово со знаком в слово	-	
TBB	[Rn, Rm]	Табличный переход по индексу, смещения - байты	-	
TBH	[Rn, Rm, LSL #1]	Табличный переход по индексу, смещения - полуслова	-	
TEQ	Rn, Op2	Проверка равенства	N,Z,C	
TST	Rn, Op2	Проверка значения бит по маске	N,Z,C	
UBFX	Rd, Rn, #lsb, #width	Чтение значения битового поля, интерпретируемого как число без знака	-	
UDIV	{Rd,} Rn, Rm	Деление чисел 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 числа в n-разрядное 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	-	

11.1 Встроенные функции

Стандарт ANSI языка C не обеспечивает непосредственного доступа к некоторым инструкциям процессора Cortex-M3. 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 51 – Встроенные функции CMSIS, позволяющие генерировать некоторые инструкции процессора Cortex-M3

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 52 – 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)

11.2 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.

11.2.1 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. “Формат второго операнда”.

11.2.2 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 Cortex-M3 поддерживает только инструкции из набора Thumb.

11.2.3 Формат второго операнда

Большинство команд обработки данных поддерживает гибкий формат задания второго операнда. Далее в описании синтаксиса инструкций процессора такой операнд будет обозначаться как Operand2. При этом в качестве операнда может выступать:
- константа;
- регистр с необязательным параметром сдвига.

11.2.3.1 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.

11.2.3.2 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".

11.2.4 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).

11.2.4.1 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 22. Инструкция ASR # 3.

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.

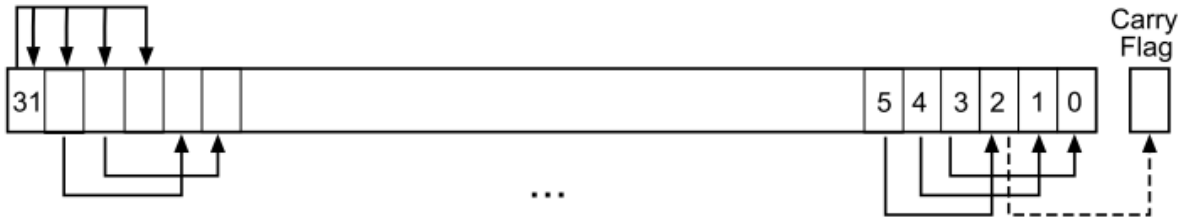


Рисунок 22. Инструкция ASR # 3

11.2.4.2 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 23 Инструкция LSR # 3.

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

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

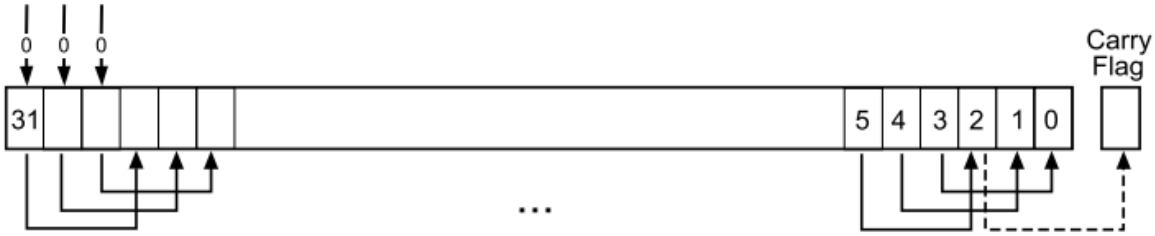


Рисунок 23. Инструкция LSR # 3

11.2.4.3 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 24. Инструкция LSL # 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.

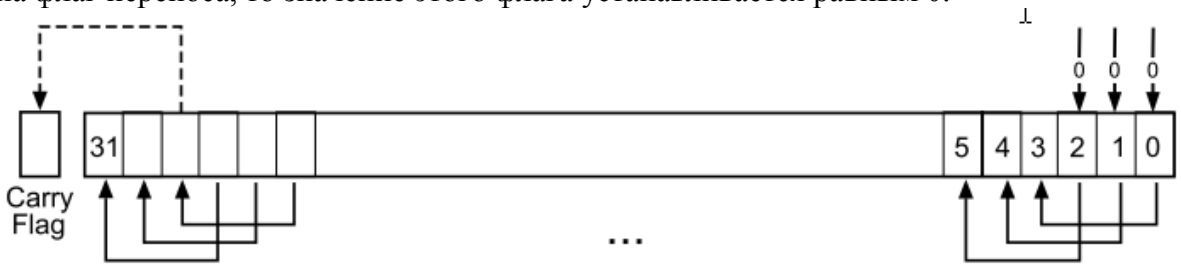


Рисунок 24. Инструкция LSL # 3

11.2.4.4 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 25. Инструкция ROR # 3.

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.

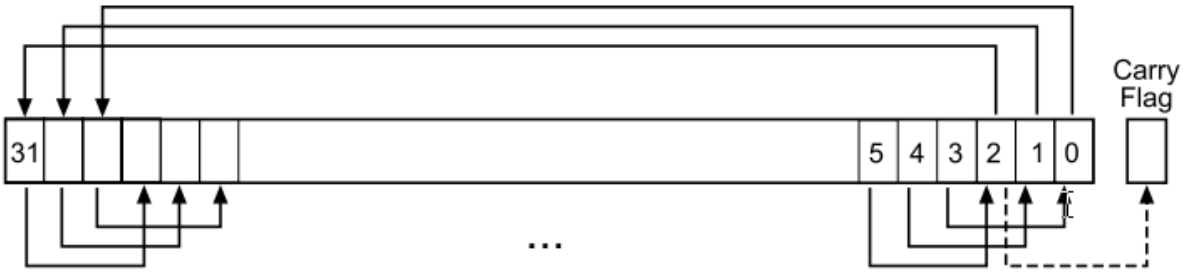


Рисунок 25. Инструкция ROR # 3

11.2.4.5 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 26. Инструкция RRX.

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.

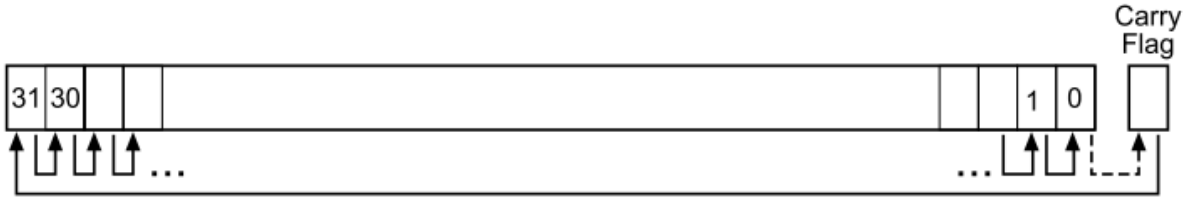


Рисунок 26. Инструкция RRX

11.2.5 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 Cortex-M3 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 ARM рекомендует программистам 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. "SCB->CCR").

11.2.6 Адресация относительно счетчика команд PC

В системе команд Cortex-M3 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].

11.2.7 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 53 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.

11.2.7.1 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 232;
- 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.

11.2.7.2 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 53). Там же указаны соответствующие логические выражения для значений флагов.

Условные команды рекомендуется использовать для снижения количества ветвлений в программе.

Таблица 53 – 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
LE	Z = 1 and N != V	Меньше или равно, знаковое сравнение
AL	1	Безусловное исполнение

11.2.7.3 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
```

11.2.7.4 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
```

11.2.8 Выбор размера кода инструкции

Многие команды процессора Cortex-M3 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
```

11.3 Команды доступа к памяти

Обобщенные данные о командах доступа к памяти представлены в Таблица 54.

Таблица 54 – 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	

11.3.1 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.
```

11.3.2 LDR и STR, непосредственно заданное смещение

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

11.3.2.1 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.

11.3.2.2 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:

11.3.2.3 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].

11.3.2.4 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]!.

11.3.2.5 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 55 показывает диапазоны значений смещения для различных форм адресации.

Таблица 55 – 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		

11.3.2.6 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.

11.3.2.7 Флаги

Данная инструкция не влияет на состояние флагов.

11.3.2.8 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.
```

11.3.3 LDR и STR, смещение задано в регистре

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

11.3.3.1 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.

11.3.3.2 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”.

11.3.3.3 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.

11.3.3.4 Флаги

Данная инструкция не влияет на состояние флагов.

11.3.3.5 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.
```

11.3.4 LDR and STR, непривилегированный доступ

Загрузка или сохранение регистра в режиме непривилегированного доступа.

11.3.4.1 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.

11.3.4.2 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.

11.3.4.3 Ограничения

В данных инструкциях:

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

11.3.4.4 Флаги

Данная инструкция не влияет на состояние флагов.

11.3.4.5 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.
```

LDR, адресация относительно счетчика команд PC

Загрузка регистра из памяти.

11.3.4.6 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".

11.3.4.7 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 56 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 56 – 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

11.3.4.8 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.

11.3.4.9 Флаги

Данная инструкция не влияет на состояние флагов.

11.3.4.10 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.
```

11.3.5 LDM и STM

Загрузка или сохранение множества регистров.

11.3.5.1 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.

11.3.5.2 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”.

11.3.5.3 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.

11.3.5.4 Флаги

Данная инструкция не влияет на состояние флагов.

11.3.5.5 Примеры

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

11.3.5.6 Примеры неправильного использования

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

11.3.6 PUSH и POP

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

11.3.6.1 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.

11.3.6.2 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”.

11.3.6.3 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.

11.3.6.4 Флаги

Данная инструкция не влияет на состояние флагов.

11.3.6.5 Примеры

```
PUSH {R0,R4-R7}
PUSH {R2,LR}
POP {R0,R10,PC}
```

11.3.7 LDREX и STREX

Эксклюзивное чтение и запись регистров.

11.3.7.1 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.

11.3.7.2 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.

Таким образом, программное обеспечение должно всегда использовать инструкции эксклюзивного чтения и записи совместно, в интересах решения задач синхронизации - см. раздел "Примитивы синхронизации".

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.

11.3.7.3 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.

11.3.7.4 Флаги

Данная инструкция не влияет на состояние флагов.

11.3.7.5 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.
```

11.3.8 CLREX

Сброс эксклюзивного доступа.

11.3.8.1 Синтаксис

CLREX{cond}

где:

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

11.3.8.2 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. Подробности см. в разделе "Примитивы синхронизации".

11.3.8.3 Флаги

Данная инструкция не влияет на состояние флагов.

11.3.8.4 Примеры

```
CLREX
```

11.4 Инструкции обработки данных

В Таблица 57 представлены инструкции обработки данных:

Таблица 57 – 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	

11.4.1 ADD, ADC, SUB, SBC и RSB

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

11.4.1.1 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.

11.4.1.2 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.

11.4.1.3 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 находиться в интервале от 0 до 4095.
  - при использовании PC в операциях сложения или вычитания биты [1:0] счетчика команд округляются до 0b00 перед выполнением операции, обеспечивая выравнивание адреса по границе слова;
  - при необходимости сформировать адрес инструкции, необходимо скорректировать значение смещения относительно PC. ARM рекомендует 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.

11.4.1.4 Флаги

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

11.4.1.5 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
```

11.4.1.6 Арифметика с повышенной разрядностью

11.4.1.7 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
```

11.4.1.8 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.
```

11.4.2 AND, ORR, EOR, BIC и ORN

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

11.4.2.1 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”.

11.4.2.2 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

11.4.2.3 Ограничения

Не допускается использованием указателя стека SP и счетчика команд PC.

11.4.2.4 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.

11.4.2.5 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
```

11.4.3 ASR, LSL, LSR, ROR и RRX

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

11.4.3.1 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.

11.4.3.2 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”.

11.4.3.3 Ограничения

Не допускается использованием указателя стека SP и счетчика команд PC.

11.4.3.4 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”.

11.4.3.5 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.
```

11.4.4 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.
```

11.4.5 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”.

Rm - 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
```

11.4.6 MOV и MVN

Загрузка в регистр прямого или инверсного значения второго операнда.

11.4.6.1 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.

11.4.6.2 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.

11.4.6.3 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, ARM настоятельно рекомендует использовать для этих целей исключительно инструкции BX и BLX, в интересах обеспечения переносимости программного обеспечения.

11.4.6.4 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.

11.4.6.5 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.
```

11.4.7 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
```

11.4.8 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
```

11.4.9 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.
```

11.5 Инструкции умножения и деления

В следующей таблице представлена информация о командах умножения и деления:

Таблица 58 – 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

11.5.1 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)
```

11.5.2 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
```

11.5.3 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.
```

11.6 Инструкции преобразования данных с насыщением

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

11.6.1 SSAT и USAT

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

11.6.1.1 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.

11.6.1.2 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 2n-1, сохраняется результат 2n-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.

11.6.1.3 Ограничения

Нельзя использовать указатель стека SP и счетчик команд PC.

11.6.1.4 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.

11.6.1.5 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/
```

11.7 Команды работы с битовыми полями

В Таблица 59 показаны инструкции, позволяющие манипулировать последовательностями смежных бит данных в регистрах или битовых полях.

Таблица 59 – 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

11.7.1 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.
```

11.7.2 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.

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 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.
```

11.7.3 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.

11.8 Инструкции передачи управления

В таблице ниже представлен список инструкций передачи управления.

Таблица 60 – 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

11.8.1 B, BL, BX и BLX

Команды ветвления.

11.8.1.1 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].

11.8.1.2 Описание

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

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

Инструкция вида B cond label – это единственный тип команды, который может находиться за пределами 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 61) 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 61 – 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

11.8.1.3 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.

11.8.1.4 Флаги

Данная инструкция не влияет на состояние флагов.

11.8.1.5 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.
```

11.8.2 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.
```

11.8.3 IT

Начало блока условно исполняемых инструкций.

11.8.3.1 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.

11.8.3.2 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.

11.8.3.3 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 находиться 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.

11.8.3.4 Флаги

Данная инструкция не влияет на состояние флагов.

11.8.3.5 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.
```

11.8.4 TBB и TBH

Табличный переход по индексу.

11.8.4.1 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.

11.8.4.2 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.

11.8.4.3 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.

11.8.4.4 Флаги

Данная инструкция не влияет на состояние флагов.

11.8.4.5 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
```

11.9 Прочие инструкции

В таблице ниже представлен список инструкций процессора Cortex-M3, не рассмотренных в предыдущих разделах.

Таблица 62 – 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

11.9.1 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.
```

11.9.2 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.
```

11.9.3 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 Synchronization Barrier.
```

11.9.4 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.
```

11.9.5 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.
```

11.9.6 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
```

11.9.7 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.
```

11.9.8 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 “Управление электропитанием”.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

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

11.9.9 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.
```

11.9.10 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.
```

11.9.11 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, разрешен или запрещен этот режим.

Флаги

Данная инструкция не влияет на состояние флагов.

Примеры

```
WFI      ; Ожидание прерывания.
```

12 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.

12.1 Описание регистров системного таймера SysTick

Таблица 63 – Описание регистров системного таймера SysTick

Адрес	Название	Тип	Доступ	Значение после сброса	Описание
0xE000E010	SysTick				Системный таймер SYSTICK
0xE000E010	CTRL	RW	привилегированный	0x00000004	SysTick->CTRL
0xE000E014	LOAD	RW	привилегированный	0x00000000	SysTick->LOAD
0xE000E018	VAL	RW	привилегированный	0x00000000	SysTick->VAL
0xE000E01C	CALIB	RO	привилегированный	0x00002904(1)	SysTick->CAL

1) Калибровочное значение системного таймера.

12.1.1 SysTick->CTRL

Регистр CTRL разрешает основные функции системного таймера.

Назначение бит:

Таблица 64 – 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.

12.1.2 SysTick->LOAD

Регистр LOAD устанавливает стартовое значение, загружаемое в регистр VAL.

Таблица 65 – 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.

12.1.3 SysTick->VAL

Регистр VAL содержит текущее значение системного таймера.

Таблица 66 – 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.

12.1.4 SysTick->CAL

Регистр CALIB показывает калибровочное значение системного таймера.

Таблица 67 – 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).

12.2 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.

13 Модуль защиты памяти MPU

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. Cortex-M3 MPU определяет:

- 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 Cortex-M3 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”.

Таблица 68 показывает возможные атрибуты MPU регионов. Здесь включены такие атрибуты памяти, как *shareable* и кэшируемость, которые не существенны во многих реализациях микроконтроллеров.

Таблица 68 – 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

13.1 Описание регистров MPU

Применяются следующие MPU регистры для определения регионов и их атрибутов.

Таблица 69 – 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

13.1.1 MPU->TYPE

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

Таблица 70 – 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.

13.1.2 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 71 – 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 разрешен, запрещение применяется к default карте памяти. Любой доступ к памяти, не покрываемой разрешённым регионом, вызывает ошибку;

1 - если MPU разрешен, разрешает применение default карты памяти как фонового региона для привилегированного программного доступа.

Когда разрешено, фоновый регион считается как номер региона `-1`. Любой регион, который 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:

Для привилегированного доступа, default карта памяти описана в разделе “Организация памяти”. Любой неадресованный доступ привилегированным программным обеспечением к разрешённому региону, ведёт себя как определено default картой памяти. Любой неадресованный 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.

13.1.3 MPU->RNR

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

Таблица 72 – 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.

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 73 – 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, 64КБ регион должен быть кратно 64КБ, например, 0x00010000 или 0x00020000.

13.1.4 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 74 – 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 78 – Кодирование привилегий доступа в поле AP.

TEX, C, B

Атрибуты доступа к памяти, см. Таблица 76 – Кодирование бит разрешения доступа.

S

Бит общего доступа, см. Таблица 75 – Пример значений поля SIZE.

SRD

Бит запрещения подрегиона. Для каждого бита в этом поле:

0 - соответствующий подрегион разрешен;

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:

$$(\text{Region size in bytes}) = 2^{(SIZE+1)}$$

Наименьший разрешенный размер региона 32 байт, соответствует значению SIZE, равному 4. В Таблица 75 представлены примеры значений SIZE, соответствующие размеру региона и значению N регистра RBAR.

Таблица 75 – Пример значений поля SIZE

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

1). Содержится в RBAR, см. раздел “Регистр базового адреса MPU региона”.

13.1.5 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 76 – 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 77 поясняет кодирование режима кэша атрибутом TEX в диапазоне значений атрибута от 4 до 7.

Таблица 77 – Кодирование режима кэша атрибутом TEX

Значение AA или BB при TEX=1xx	Соответствующий режим кэша
00	Не кэшируемая
01	Отложенное обновление, запись и чтение пакетные
10	Синхронное обновление, запись без кэширования
11	Отложенное обновление, запись без кэширования

Таблица 78 поясняет кодирование бит AP, определяющих разрешение на доступ для привилегированного и непривилегированного программного обеспечения (ПО).

Таблица 78 – Кодирование привилегий доступа в поле AP

AP[2:0]	Привилегированный доступ	Непривилегированный 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

13.1.6 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. подраздел “Поле MMFSR”.

13.1.7 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.

13.1.7.1 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.

13.1.7.2 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.
```

13.1.7.3 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.

13.1.7.4 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.

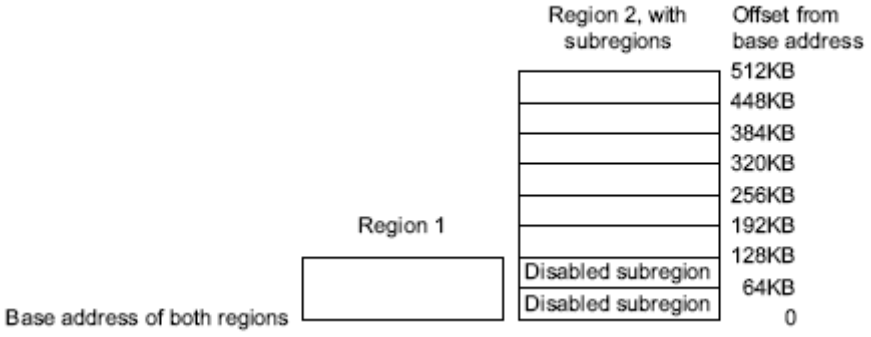


Рисунок 27. Применение SRD

13.2 Советы и особенности применения MPU

Во избежание непредвиденных ситуаций, запретите прерывания перед обновлением атрибутов региона, к которому может осуществляться доступ в обработчике прерываний.

Убедитесь, что программное обеспечение использует корректный доступ, соответствующий размеру регистров MPU:

- за исключением RASR, необходимо использовать доступ по словам;

- для RASR может использоваться доступ по байтам, полусловам или словам.

Процессор не поддерживает невыровненный доступ к регистрам MPU.

Если MPU перенастраивается, то запретите неиспользуемые регионы для предотвращения любых предыдущих настроек регионов от их влияния на новые настройки.

13.2.1 Конфигурация MPU для микроконтроллера

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

Таблица 79 – 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.

14 Сигналы тактовой частоты MDR_RST_CLK

Микроконтроллер имеет два встроенных и два внешних генератора, а также специализированный блок формирования тактовой синхронизации микроконтроллера.

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, поэтому 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 затем осуществлено переключение на них на соответствующих мультиплексорах, управляемых регистрами CPU_CLOCK и USB_CLOCK.

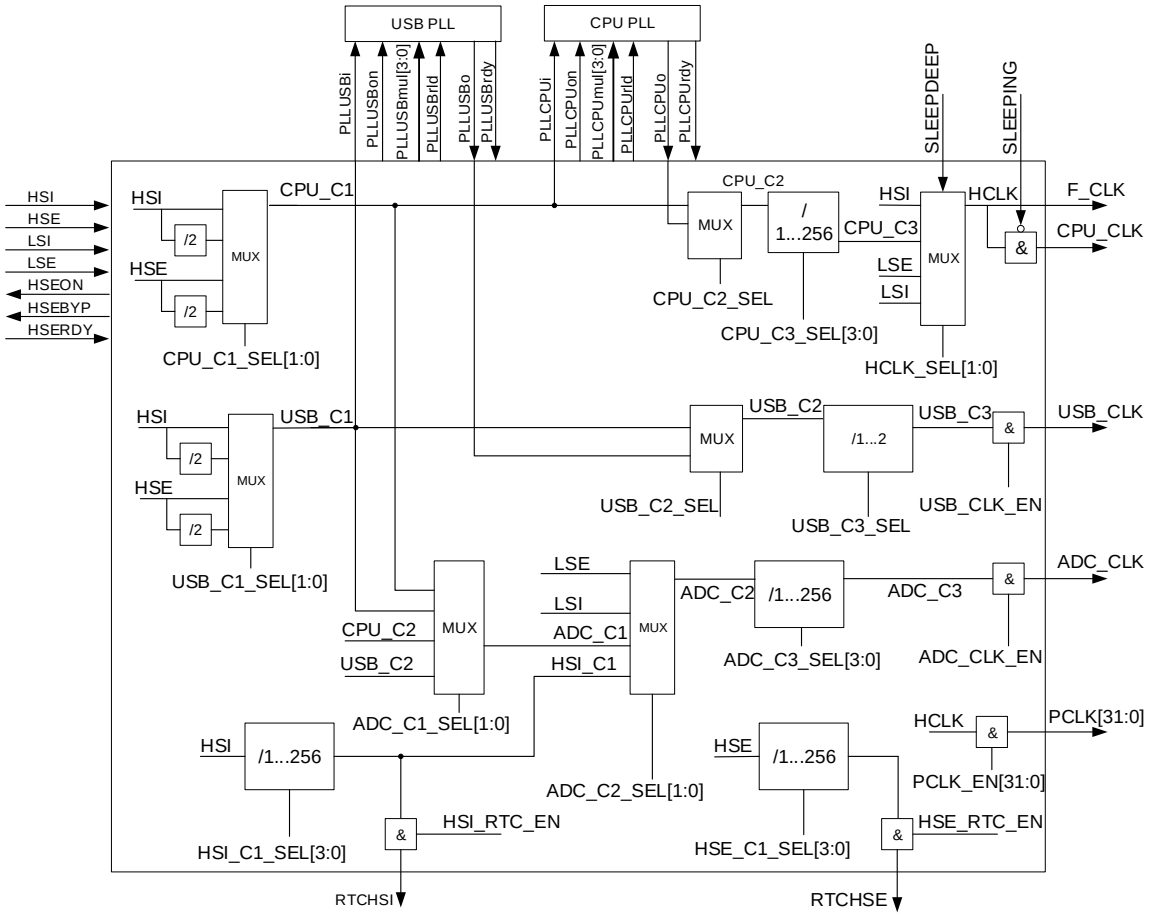


Рисунок 28. Структурная блок – схема формирования тактовой частоты

Встроенный RC генератор HSI

Генератор HSI вырабатывает тактовую частоту f_{O_HSI} с типовым значением 8 МГц. Генератор автоматически запускается при появлении питания U_{CC} и при выходе в нормальный

режим работы вырабатывает сигнал HSIRDY в регистре батарейного домена BKP_REG_0F. Первоначально процессорное ядро запускается на тактовой частоте HSI. При дальнейшей работе генератор HSI может быть отключен при помощи сигнала HSION в регистре BKP_REG_0F. Так же генератор может быть подстроен при помощи сигнала HSITRIM в регистре BKP_REG_0F.

Встроенный RC генератор LSI

Генератор LSI вырабатывает тактовую частоту f_{O_LSI} с типовым значением 40 кГц. Генератор автоматически запускается при появлении питания U_{CC} и при выходе в нормальный режим работы вырабатывает сигнал LSIRDY в регистре BKP_REG_0F. Первоначально тактовая частота генератор LSI используется для формирования дополнительной задержки tpor. При дальнейшей работе генератор LSI может быть отключен при помощи сигнала LSION в регистре BKP_REG_0F.

Внешний генератор HSE

Генератор HSE предназначен для выработки тактовой частоты 2..16 МГц с помощью внешнего резонатора. Генератор запускается при появлении питания U_{CC} и сигнала разрешения HSEON в регистре HS_CONTROL. При выходе в нормальный режим работы вырабатывает сигнал HSERDY в регистре CLOCK_STATUS. Также этот генератор 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 генератор LSE

Генератор LSE предназначен для выработки тактовой частоты 32 кГц с помощью внешнего резонатора. Генератор запускается при появлении питания BDU_{CC} и сигнала разрешения LSEON в регистре BKP_REG_0F. При выходе в нормальный режим работы вырабатывает сигнал LSERDY в регистре BKP_REG_0F. Также осциллятор может работать в режиме LSEBYP, когда входная тактовая частота с входа OSC_IN32 проходит напрямую на выход LSE. Выход OSC_OUT32 находится в этом режиме третьем состоянии. Так как генератор LSE питается от напряжения питания BDU_{CC} 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, задаваемый 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 USB тактовой частоты

Блок умножения позволяет провести умножение входной тактовой частоты на коэффициент от 2 до 16, задаваемый 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.

14.1 Описание регистров блока контроллера тактовой частоты

Таблица 80 – Описание регистров блока контроллера тактовой частоты

Базовый Адрес	Название	Описание
0x4002_0000	MDR_RST_CLK	Контроллер тактовой частоты
Смещение		
0x00	CLOCK_STATUS	MDR_RST_CLK->CLOCK_STATUS Регистр состояния блока управления тактовой частотой
0x04	PLL_CONTROL	MDR_RST_CLK->PLL_CONTROL Регистр управления блоками умножения частоты
0x08	HS_CONTROL	MDR_RST_CLK->HS_CONTROL Регистр управления высокочастотным генератором и осциллятором
0x0C	CPU_CLOCK	MDR_RST_CLK->CPU_CLOCK 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	CAN_CLOCK	MDR_RST_CLK->CAN_CLOCK Регистр управления тактовой частотой CAN
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

14.1.1 MDR_RST_CLK->CLOCK_STATUS

Таблица 81 – 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 82 – Описание бит регистра CLOCK_STATUS

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31...3	-	Зарезервировано
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

14.1.2 MDR_RST_CLK->PLL_CONTROL

Таблица 83 – 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 CPU MUL[3:0]**	**PLL USB MUL[3:0]**	**PLL CPU RLD**	**PLL CPU ON**	**PLL USB RLD**	**PLL USB ON**

Таблица 84 – Описание бит регистра PLL_CONTROL

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31…12	-	Зарезервировано
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 RLD	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

14.1.3 MDR_RST_CLK->HS_CONTROL

Таблица 85 – Регистра HS_CONTROL

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

Таблица 86 – 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

14.1.4 MDR_RST_CLK->CPU_CLOCK

Таблица 87 – 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 88 – 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 – HSI 01 – HSI/2 10 – HSE 11 – HSE/2

14.1.5 MDR_RST_CLK->USB_CLOCK

Таблица 89 – 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 90 – 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

14.1.6 MDR_RST_CLK->ADC_MCO_CLOCK

Таблица 91 – 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 92 – Описание бит регистра ADC_MCO_CLOCK

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31…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

14.1.7 MDR_RST_CLK->RTC_CLOCK

Таблица 93 – Регистр RTC_CLOCK

Номер	31…10	9	8	7…4	3…0
Доступ	U	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0
	-	**HSI RTC EN**	**HSE RTC EN**	**HSI_C1 SEL[1:0]**	**HSE_C1 SEL[1:0]**

Таблица 94 – 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 – разрешен
8	HSE RTC EN	Бит разрешения HSE RTC: 0 – запрещен; 1 – разрешен
7…4	HSI_C1 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_C1 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

14.1.8 MDR_RST_CLK->PER_CLOCK

Таблица 95 – Регистр PER_CLOCK

Номер	31...5	4	3...0
Доступ	R/W	R/W	R/W
Сброс	0	1	0
	PCLK_EN[31:5]	**PCLK_EN[4]**	**PCLK_EN[3:0]**

Таблица 96 – Описание бит регистра PER_CLOCK

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31…5	PCLK EN[31:5]	Биты разрешения тактирования периферийных блоков: 0 – запрещено; 1 – разрешено. PCLK[5] – DMA PCLK[6] – UART1 PCLK[7] – UART2 PCLK[8] – SPI1 PCLK[9] – зарезервировано 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] – зарезервировано PCLK[27] – BKP PCLK[28] – зарезервировано PCLK[29] – PORTF PCLK[30] – EXT_BUS_CNTRL PCLK[31] – зарезервировано
4	PCLK EN[4]	Биты разрешения тактирования периферийных блоков: 0 – запрещено; 1 – разрешено. PCLK[4] – RST_CLK. После сброса в состоянии 1
3…0	PCLK EN[3:0]	Биты разрешения тактирования периферийных блоков: 0 – запрещено; 1 – разрешено. PCLK[0] – CAN1 PCLK[1] – CAN2 PCLK[2] – USB PCLK[3] – EEPROM_CNTRL

14.1.9 MDR_RST_CLK->CAN_CLOCK

Таблица 97 – Регистр CAN_CLOCK

Номер	31…26	25	24	23…16	15…8	7…0
Доступ	U	R/W	R/W	U	R/W	R/W
Сброс	0	0	0	0	0	0
	-	**CAN2 CLK EN**	**CAN1 CLK EN**	**-**	**CAN2 BRG [7:0]**	**CAN1 BRG [7:0]**

Таблица 98 – Описание бит регистра CAN_CLOCK

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31...26	-	Зарезервировано
25	CAN2 CLK EN	Разрешение тактовой частоты на CAN2: 0 – нет частоты; 1 – есть частота
24	CAN1 CLK EN	Разрешение тактовой частоты на CAN2: 0 – нет частоты; 1 – есть частота
23..16	-	Зарезервировано
15…8	CAN2 BRG [7:0]	Делитель тактовой частоты CAN2 xxxxx000 – CAN2_CLK == HCLK xxxxx001 – CAN2_CLK == HCLK/2 xxxxx010 – CAN2_CLK == HCLK/4 … xxxxx111 – CAN2_CLK == HCLK/128
7…0	CAN1 BRG [7:0]	Делитель тактовой частоты CAN1 xxxxx000 – CAN1_CLK == HCLK xxxxx001 – CAN1_CLK == HCLK/2 xxxxx010 – CAN1_CLK == HCLK/4 … xxxxx111 – CAN1_CLK == HCLK/128

14.1.10 MDR_RST_CLK->TIM_CLOCK

Таблица 99 – 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 100 – 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 TIM2: 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 TIM1: xxxxx000 – TIM1_CLK == HCLK xxxxx001 – TIM1_CLK == HCLK/2 xxxxx010 – TIM1_CLK == HCLK/4 … xxxxx111 – TIM1_CLK == HCLK/128

14.1.11 MDR_RST_CLK->UART_CLOCK

Таблица 101 – 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 102 – Описание бит регистра UART_CLOCK

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31…27	-	Зарезервировано
26	-	Зарезервировано
25	UART2 CLK EN	Разрешение тактовой частоты на UART2: 0 – нет частоты; 1 – есть частота
24	UART1 CLK EN	Разрешение тактовой частоты на UART 1: 0 – нет частоты; 1 – есть частота
23..16	-	Зарезервировано
15…8	UART2 BRG [7:0]	Делитель тактовой частоты UART 2: xxxxx000 – UART 2_CLK == HCLK xxxxx001 – UART 2_CLK == HCLK/2 xxxxx010 – UART 2_CLK == HCLK/4 … xxxxx111 – UART 2_CLK == HCLK/128
7…0	UART1 BRG [7:0]	Делитель тактовой частоты UART1: xxxxx000 – UART 1_CLK == HCLK xxxxx001 – UART 1_CLK == HCLK/2 xxxxx010 – UART 1_CLK == HCLK/4 … xxxxx111 – UART 1_CLK == HCLK/128

14.1.12 MDR_RST_CLK->SSP_CLOCK

Таблица 103 – Регистр SSP_CLOCK

Номер	31…26	25	24	23…16	15…8	7…0
Доступ	U	R/W	R/W	U	R/W	R/W
Сброс	0	0	0	0	0	0
	-	**SSP2 CLK EN**	**SSP 1 CLK EN**	-	**SSP 2 BRG [7:0]**	**SSP 1 BRG [7:0]**

Таблица 104 – Описание бит регистра SSP_CLOCK

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31...27	-	Зарезервировано
26	-	Зарезервировано
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	-	Зарезервировано
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 SSP1: xxxxx000 – SSP 1_CLK == HCLK xxxxx001 – SSP 1_CLK == HCLK/2 xxxxx010 – SSP 1_CLK == HCLK/4 … xxxxx111 – SSP 1_CLK == HCLK/128

15 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.

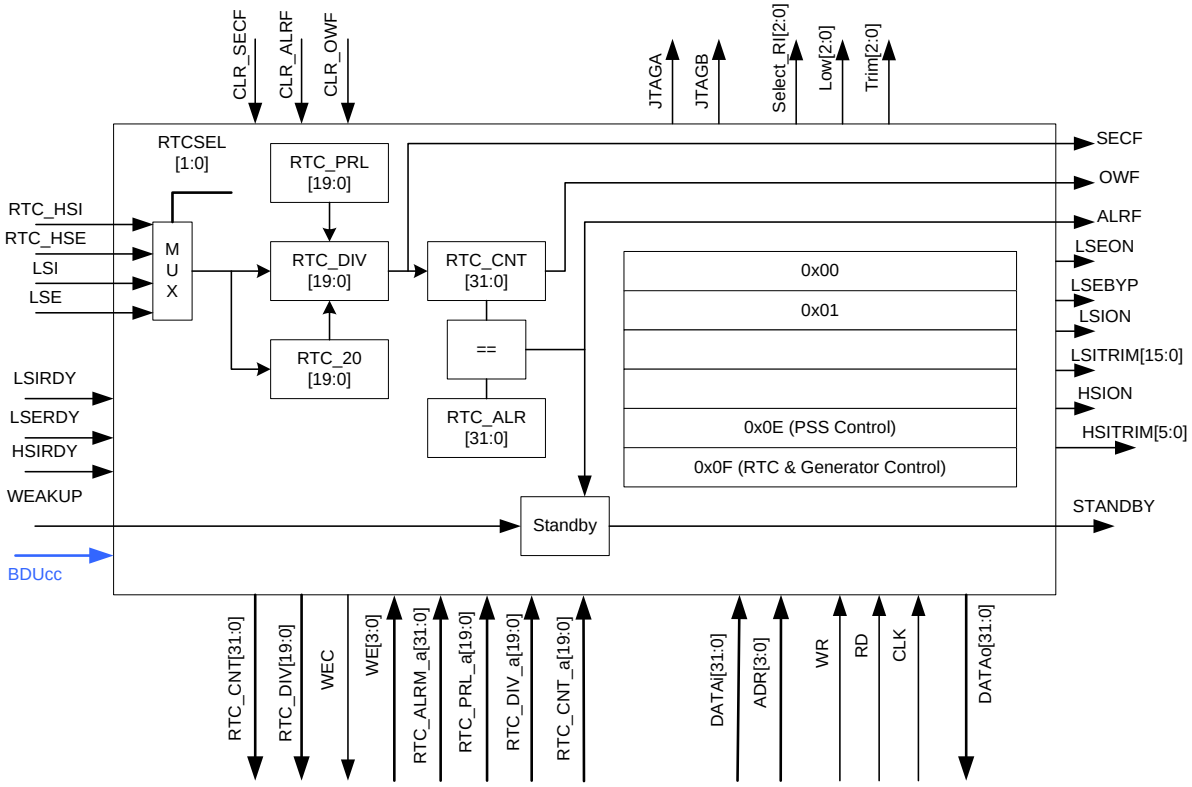


Рисунок 29. Структурная блок-схема батарейного домена и часов реального времени

15.1 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 внутренним регистром RTC_20 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-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.

15.2 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.

15.3 Описание регистров блока батарейного домена

Таблица 105 – 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->REG_[0D…00] MDR_BKP->REG_00 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

15.3.1 MDR_BKP->REG_[0D…00]

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 106 – Регистры REG_[0D…00]

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

Таблица 107 – 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

15.3.2 MDR_BKP->REG_0E

Таблица 108 – 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 109 – 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 – разрешен

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

15.3.3 MDR_BKP->REG_0F

Таблица 110 – 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	RO	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 111 – 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 HSI. Смотрите диаграмму зависимости (Рисунок 31)

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 работает в рабочем режиме
20...16	LSI TRIM[4:0]	Коэффициент подстройки частоты генератора LSI. Смотрите диаграмму зависимости (Рисунок 30)
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 = 255 тактов частота = 32760,03125 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	Бит управления генератором LSE: 0 – генератор выключен; 1 – генератор включен

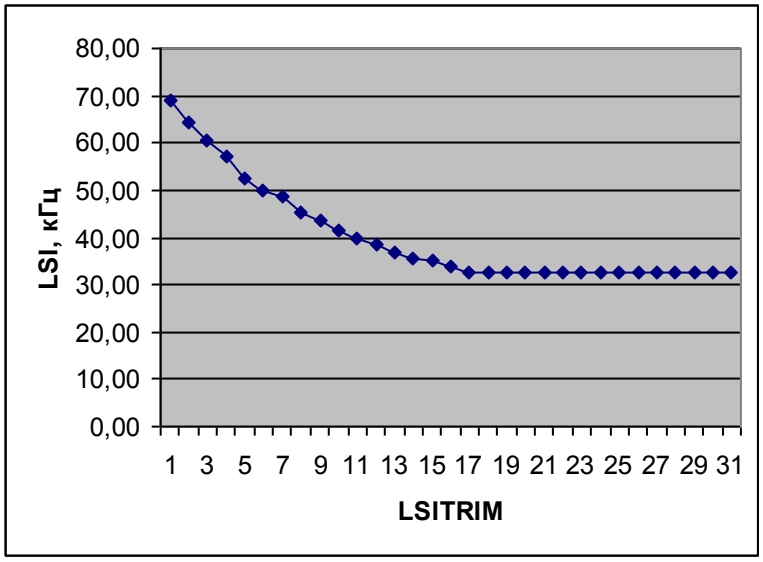


Рисунок 30. Зависимость частоты LSI от значения LSITRIM

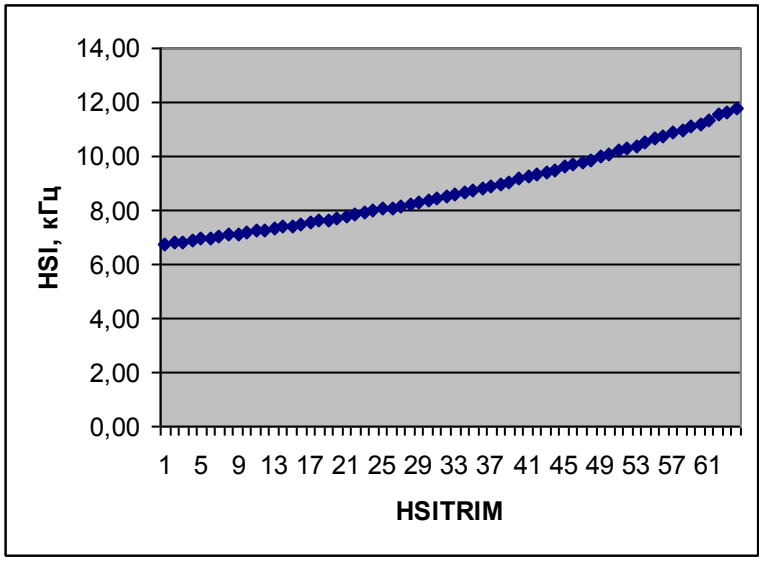


Рисунок 31. Зависимость частоты HSI от значения HSITRIM

MDR_BKP->RTC_CNT

Таблица 112 – Регистр RTC_CNT

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

Таблица 113 – 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

15.3.4 MDR_BKP->RTC_DIV

Таблица 114 – Регистр RTC_DIV

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

Таблица 115 – 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

15.3.5 MDR_BKP->RTC_PRL

Таблица 116 – Регистр RTC_PRL

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

Таблица 117 – 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

15.3.6 MDR_BKP->RTC_ALRM

Таблица 118 – Регистр RTC_ALRM

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

Таблица 119 – 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

15.3.7 MDR_BKP->RTC_CS

Таблица 120 – 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 121 – 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. Сброс Флага осуществляется записью 1
1	**SECF**	Флаг разрешения счета основного счетчика от счетчика предварительного деления: 0 – нет разрешения счета; 1 – разрешение счета. Сброс Флага осуществляется записью 1
0	**OWF**	Флаг переполнения основного счетчика RTC_CNT: 0 – нет переполнения;

		1 – было переполнение. Сброс Флага осуществляется записью 1

16 Порты ввода-вывода MDR_PORTx

Микроконтроллер имеет 6 портов ввода/вывода. Порты 16-разрядные и их выводы мультиплексируются между различными функциональными блоками, управление для каждого вывода отдельное. Для того, чтобы выводы порта перешли под управление того или иного периферийного блока, необходимо задать для нужных выводов выполняемую функцию и настройки.

При работе в режиме отладки не допускается изменение функций записью 1 в регистры RXTX и OE выводов, совмещенных с выводами JTAG. Это может привести к блокировке интерфейса отладки.

Таблица 122 – 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)	EXT_INT1	9)	-	
PA1	-	PA1		DATA1		TMR1_CH1		TMR2_CH1	
PA2	-	PA2		DATA2		TMR1_CH1N		TMR2_CH1N	
PA3	-	PA3		DATA3		TMR1_CH2		TMR2_CH2	
PA4	-	PA4		DATA4		TMR1_CH2N		TMR2_CH2N	
PA5	-	PA5		DATA5		TMR1_CH3		TMR2_CH3	
PA6	-	PA6		DATA6		CAN1_TX	2)	UART1_RXD	
PA7	-	PA7		DATA7		CAN1_RX		UART1_TXD	
PA8	-	PA8		DATA8		TMR1_CH3N		TMR2_CH3N	
PA9	-	PA9		DATA9		TMR1_CH4		TMR2_CH4	
PA10	-	PA10		DATA10		nUART1DTR	10)	TMR2_CH4N	
PA11	-	PA11		DATA11		nUART1RTS		TMR2_BLK	
PA12	-	PA12		DATA12		nUART1RI		TMR2_ETR	
PA13	-	PA13		DATA13		nUART1DCD		TMR1_CH4N	
PA14	-	PA14		DATA14		nUART1DSR		TMR1_BLK	
PA15	-	PA15		DATA15		nUART1CTS		TMR1_ETR	
Порт B									
PB0	-	PB0	JA_TDO	DATA16	1)	TMR3_CH1		UART1_TXD	
PB1	-	PB1	JA_TMS	DATA17		TMR3_CH1N		UART2_RXD	
PB2	-	PB2	JA_TCK	DATA18		TMR3_CH2		CAN1_TX	
PB3	-	PB3	JA_TDI	DATA19		TMR3_CH2N		CAN1_RX	
PB4	-	PB4	JA_TRST	DATA20		TMR3_BLK		TMR3_ETR	
PB5	-	PB5		DATA21		UART1_TXD	10)	TMR3_CH3	
PB6	-	PB6		DATA22		UART1_RXD		TMR3_CH3N	
PB7	-	PB7		DATA23		nSIROUT1		TMR3_CH4	
PB8	-	PB8		DATA24		COMP_OUT	7)	TMR3_CH4N	
PB9	-	PB9		DATA25		nSIRIN1	10)	EXT_INT4	
PB10	-	PB10		DATA26		EXT_INT2	9)	nSIROUT1	
PB11	-	PB11		DATA27		EXT_INT1		COMP_OUT	
PB12	-	PB12		DATA28		SSP1_FSS		SSP2_FSS	
PB13	-	PB13		DATA29		SSP1_CLK		SSP2_CLK	
PB14	-	PB14		DATA30		SSP1_RXD		SSP2_RXD	
PB15	-	PB15		DATA31		SSP1_TXD		SSP2_TXD	

Порт C									
PC0	-		PC0	READY [17]	1)	SCL1	11)	SSP2_FSS	
PC1	-		PC1	OE		SDA1		SSP2_CLK	
PC2	-		PC2	WE		TMR3_CH1	12)	SSP2_RXD	
PC3	-		PC3	BE0		TMR3_CH1N		SSP2_TXD	
PC4	-		PC4	BE1		TMR3_CH2		TMR1_CH1	
PC5	-		PC5	BE2		TMR3_CH2N		TMR1_CH1N	
PC6	-		PC6	BE3		TMR3_CH3		TMR1_CH2	
PC7	-		PC7	CLOCK		TMR3_CH3N		TMR1_CH2N	
PC8	-		PC8	CAN1_TX	2)	TMR3_CH4		TMR1_CH3	
PC9	-		PC9	CAN1_RX		TMR3_CH4N		TMR1_CH3N	
PC10	-		PC10	-		TMR3_ETR		TMR1_CH4	
PC11	-		PC11	-		TMR3_BLK		TMR1_CH4N	
PC12	-		PC12	-		EXT_INT2		TMR1_ETR	
PC13	-		PC13	-		EXT_INT4	9)	TMR1_BLK	
PC14	-		PC14	-		SSP2_FSS	13)	CAN2_RX	
PC15	-		PC15	-		SSP2_RXD		CAN2_TX	
Порт D									
PD0	ADC0_REF+	5	PD0 JB_TMS	TMR1_CH1N	3)	UART2_RXD	14)	TMR3_CH1	
PD1	ADC1_REF-		PD1 JB_TCK	TMR1_CH1		UART2_TXD		TMR3_CH1N	
PD2	ADC2		PD2 JB_TRST	BUSY1	1)	SSP2_RXD	13)	TMR3_CH2	
PD3	ADC3		PD3 JB_TDI	-		SSP2_FSS		TMR3_CH2N	
PD4	ADC4		PD4 JB_TDO	TMR1_ETR		nSIROUT2	14)	TMR3_BLK	
PD5	ADC5		PD5	CLE	1)	SSP2_CLK	13)	TMR2_ETR	
PD6	ADC6		PD6	ALE		SSP2_TXD	13)	TMR2_BLK	
PD7	ADC7		PD7	TMR1_BLK	3)	nSIRIN2	14)	UART1_RXD	
PD8	ADC8		PD8	TMR1_CH4N		TMR2_CH1		UART1_TXD	
PD9	ADC9		PD9	CAN2_TX	4)	TMR2_CH1N		SSP1_FSS	
PD10	ADC10		PD10	TMR1_CH2	3)	TMR2_CH2		SSP1_CLK	
PD11	ADC11		PD11	TMR1_CH2N		TMR2_CH2N		SSP1_RXD	
PD12	ADC12		PD12	TMR1_CH3		TMR2_CH3		SSP1_TXD	
PD13	ADC13		PD13	TMR1_CH3N		TMR2_CH3N		CAN1_TX	
PD14	ADC14		PD14	TMR1_CH4		TMR2_CH4		CAN1_RX	
PD15	ADC15		PD15	CAN2_RX	4)	BUSY2	1)	EXT_INT3	
Порт E									
PE0	DAC2_OUT	6	PE0	ADDR16	1)	TMR2_CH1	15)	CAN1_RX	
PE1	DAC2_REF		PE1	ADDR17		TMR2_CH1N		CAN1_TX	
PE2	COMP_IN1	7	PE2	ADDR18		TMR2_CH3		TMR3_CH1	
PE3	COMP_IN2		PE3	ADDR19		TMR2_CH3N		TMR3_CH1N	
PE4	COMP_REF+		PE4	ADDR20		TMR2_CH4N		TMR3_CH2	
PE5	COMP_REF-		PE5	ADDR21		TMR2_BLK		TMR3_CH2N	
PE6	OSC_IN32	8	PE6	ADDR22		CAN2_RX	4)	TMR3_CH3	
PE7	OSC_OUT32		PE7	ADDR23		CAN2_TX		TMR3_CH3N	
PE8	COMP_IN3	7	PE8	ADDR24		TMR2_CH4	15)	TMR3_CH4	
PE9	DAC1_OUT	6	PE9	ADDR25		TMR2_CH2		TMR3_CH4N	
PE10	DAC1_REF		PE10	ADDR26		TMR2_CH2N		TMR3_ETR	
PE11	-		PE11	ADDR27		nSIRIN1		TMR3_BLK	
PE12	-		PE12	ADDR28		SSP1_RXD	16)	UART1_RXD	
PE13	-		PE13	ADDR29		SSP1_FSS		UART1_TXD	
PE14	-		PE14	ADDR30		TMR2_ETR	15)	SCL1	
PE15	-		PE15	ADDR31		EXT_INT3	9)	SDA1	

Порт F									
PF0	-		PF0	ADDR0	1)	SSP1TXD	16)	UART2_RXD	14)
PF1	-		PF1	ADDR1		SSP1CLK		UART2_TXD	
PF2	-		PF2	ADDR2		SSP1FSS		CAN2_RX	
PF3	-		PF3	ADDR3		SSP1RXD		CAN2_TX	
PF4	-		PF4 MODE[0]	ADDR4		-		-	
PF5	-		PF5 MODE[1]	ADDR5		-		-	
PF6	-		PF6 MODE[2]	ADDR6		TMR1_CH1	3)	-	
PF7	-		PF7	ADDR7		TMR1_CH1N		TMR3_CH1	
PF8	-		PF8	ADDR8		TMR1_CH2		TMR3_CH1N	
PF9	-		PF9	ADDR9		TMR1_CH2N		TMR3_CH2	
PF10	-		PF10	ADDR10		TMR1_CH3		TMR3_CH2N	
PF11	-		PF11	ADDR11		TMR1_CH3N		TMR3_ETR	
PF12	-		PF12	ADDR12		TMR1_CH4		SSP2_FSS	
PF13	-		PF13	ADDR13		TMR1_CH4N		SSP2_CLK	
PF14	-		PF14	ADDR14		TMR1_ETR		SSP2_RXD	
PF15	-		PF15	ADDR15		TMR1_BLK		SSP2_TXD	

Примечания:

1) Выводы управляются системной шиной EXT_BUS.
2) Выводы управляются контроллером интерфейса CAN1.
3) Выводы управляются Таймером 1.
4) Выводы управляются контроллером интерфейса CAN2.
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) Реализовано только в 1986BE94x

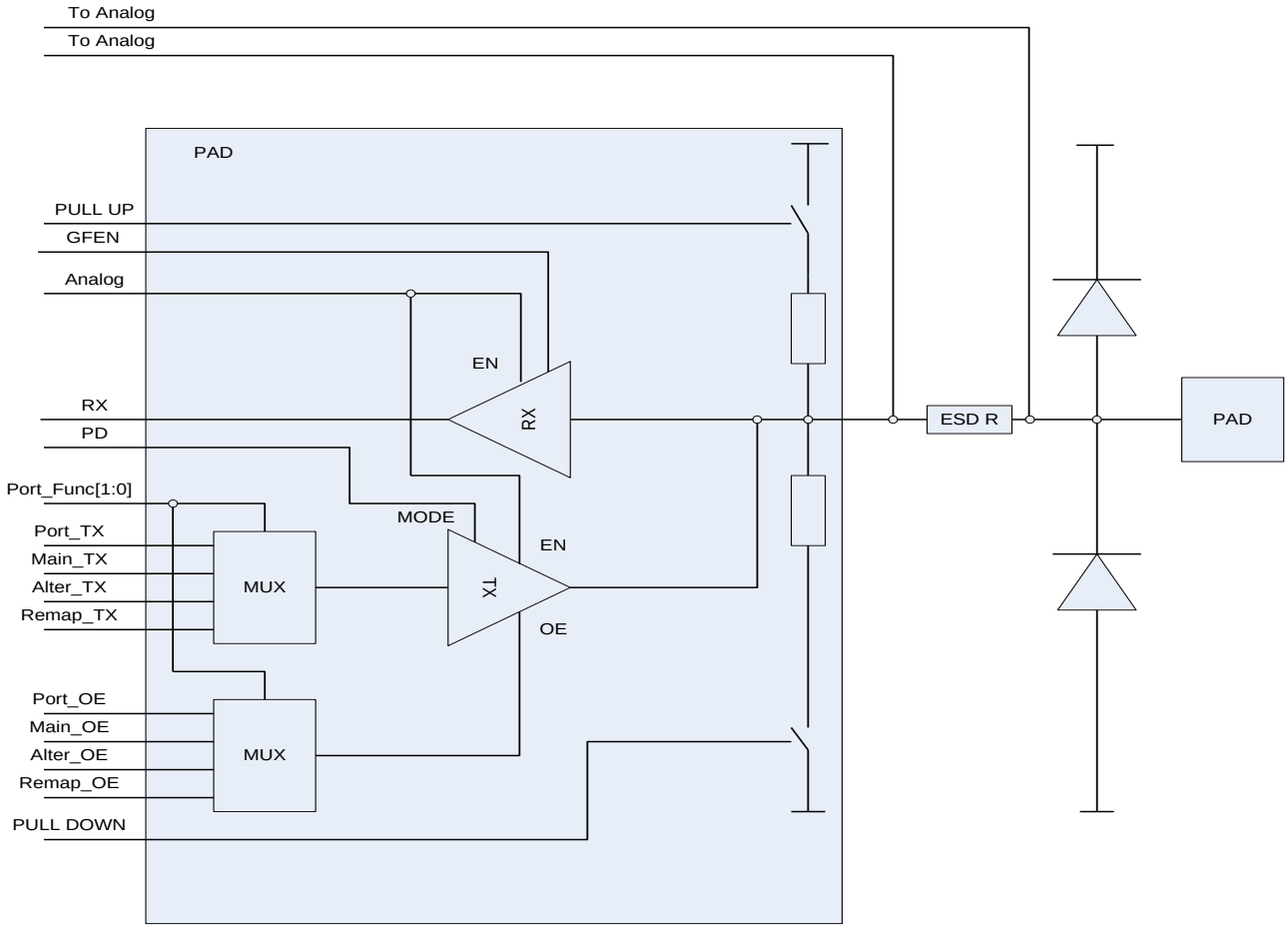


Рисунок 32. Порты ввода-вывода

16.1 Описание регистров портов ввода-вывода

Таблица 123 – Описание регистров портов ввода-вывода

Базовый Адрес	Название	Описание
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

16.1.1 MDR_PORTx->RXTX

Таблица 124 – Регистр RXTX

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

Таблица 125 – 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

16.1.2 MDR_PORTx->OE

Таблица 126 – Регистр OE

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

Таблица 127 – 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

16.1.3 MDR_PORTx->FUNC

Таблица 128 – 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 129 – 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

16.1.4 MDR_PORTx->ANALOG

Таблица 130 – Регистр ANALOG

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

Таблица 131 – 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

16.1.5 MDR_PORTx->PULL

Таблица 132 – Регистр PULL

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

Таблица 133 – 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)

16.1.6 MDR_PORTx->PD

Таблица 134 – Регистр PD

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

Таблица 135 – 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 PD[15:0]	Режим работы контроллера. Режим работы выхода: 0 – управляемый драйвер; 1 – открытый сток

16.1.7 MDR_PORTx->PWR

Таблица 136 – 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 137 – 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 (порядка 100 нс) 10 – быстрый фронт (порядка 20 нс) 11 – максимально быстрый фронт (порядка 10 нс)

16.1.8 MDR_PORTx->GFEN

Таблица 138 – Регистр GFEN

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

Таблица 139 – 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 (фильтрация импульсов до 10 нс)

17 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 140 – 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 141 – 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

17.1.1 MDR_POWER->PVDCS

Таблица 142 – 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 143 – 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 *Примечание –* сброс результата сравнения необходимо проводить с подтверждением - сбрасывать дважды.

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
6	PVBD	Результат сравнения напряжения батарейного питания Устанавливается событием, очищается записью 0. Если событие продолжается запись игнорируется: 0 – напряжение питания меньше чем уровень, задаваемый PBLS; 1 – напряжение питания больше чем уровень, задаваемый PBLS *Примечание –* сброс результата сравнения необходимо проводить с подтверждением - сбрасывать дважды.
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

18 Внешняя системная шина MDR_EBC (кроме 1986BE94T)

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

Таблица 144 – 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 начинает выполняться программа
0x3000_0000 – 0x3FFF_FFFF	256 Мбайт	Область памяти секции DATA отображаемая на внешнюю системную шину с доступом через S Bus. К этой области имеет доступ DMA контроллер
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.

18.1 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.

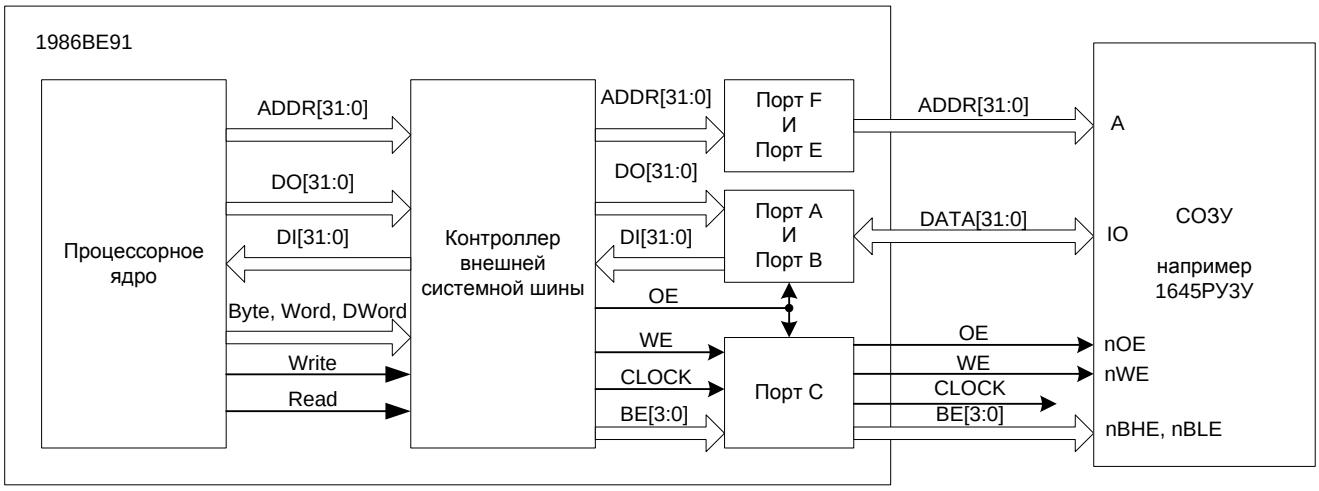


Рисунок 33. Обмен по внешней системной шине

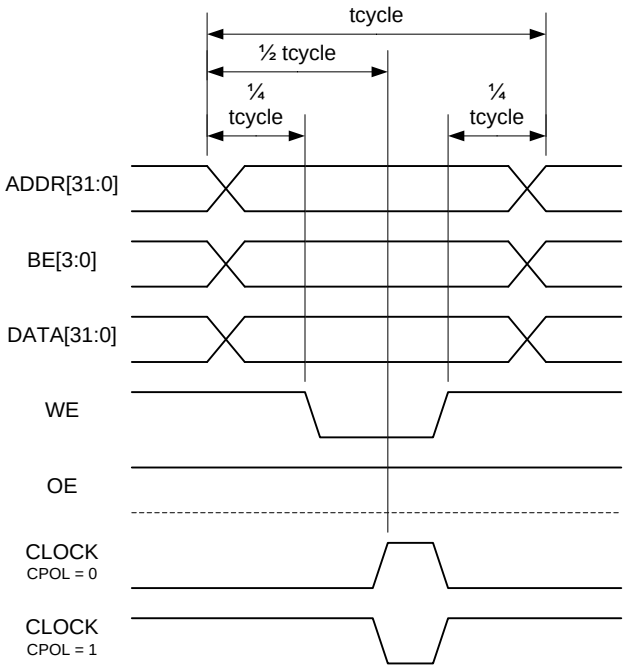


Рисунок 34. Диаграмма записи

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.

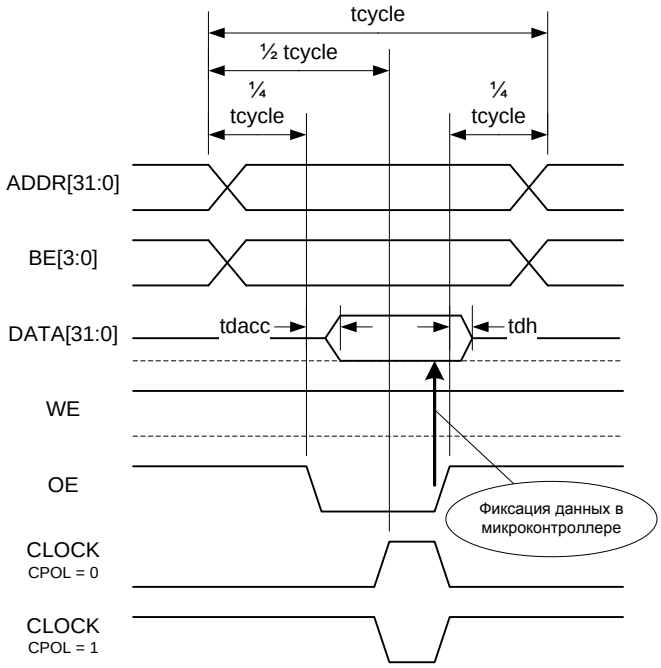


Рисунок 35. Диаграмма чтения

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 145. 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

18.2 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.

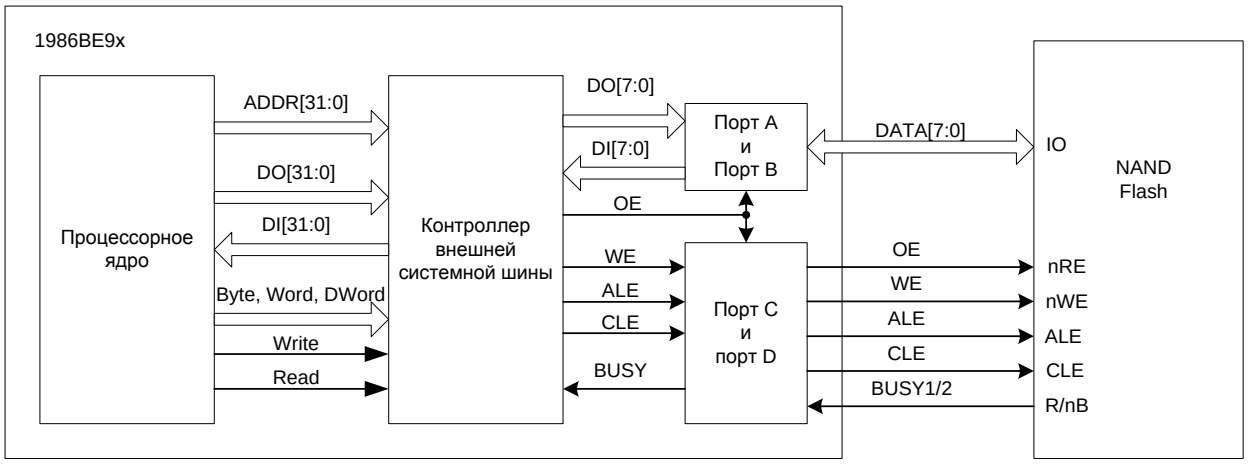


Рисунок 36. Подключение внешней NAND Flash

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 BUSY2 – 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 146).

Таблица 146 – 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	A[23:22] не имеют значения A[21] сигнализирует о необходимости снятия сигнала выбора кристалла по завершению текущей команды, если этот бит в нуле, то устройство остается выбранным по завершению команды

Адрес обращения	Фаза команды	Фаза данных
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
```

18.3 Описание регистров блока контроллера внешней системной шины

Таблица 147 – 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

18.3.1 MDR_EBC->CONTROL

Таблица 148 – 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 149 – 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

18.3.2 MDR_EBC->NAND_CYCLES

Таблица 150 – 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 151 – 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

19 Внешняя системная шина MDR_EBC (для 1986BE94T)

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

Таблица 152 – 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 начинает выполняться программа
0x3000_0000 – 0x3FFF_FFFF	256 Мбайт	Область памяти секции DATA отображаемая на внешнюю системную шину с доступом через S Bus. К этой области имеет доступ DMA контроллер
0x5000_0000 – 0x5FFF_FFFF	256 Мбайт	Область памяти секции PERIPHERAL и EXTERNAL BUS отображаемая на внешнюю системную шину с доступом через S Bus. К этой области имеет доступ DMA контроллер
0x6000_0000 – 0xDFFF_FFFF	2 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. Для каждого из 4 регионов можно задавать собственные настройки длительности транзакции на шине.

19.1 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] при EXT_BUS_CTRL_MODE=0 или ENABLE=0 для соответствующего региона, или через биты WS_ACTIVE[6:0], WS_SETUP[2:0] и WS_HOLD[2:0] при EXT_BUS_CTRL_MODE=1 или ENABLE=1 для соответствующего региона. 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. На вывод CLOCK могут выдаваться одиночные импульсы внутри транзакции при EXT_BUS_CTRL_MODE=0 либо непрерывно тактовая частота ядра при EXT_BUS_CTRL_MODE=1 и CLOCKOUT_EN=1. Полярность сигнала на выходе CLOCK задается битом CPOL.

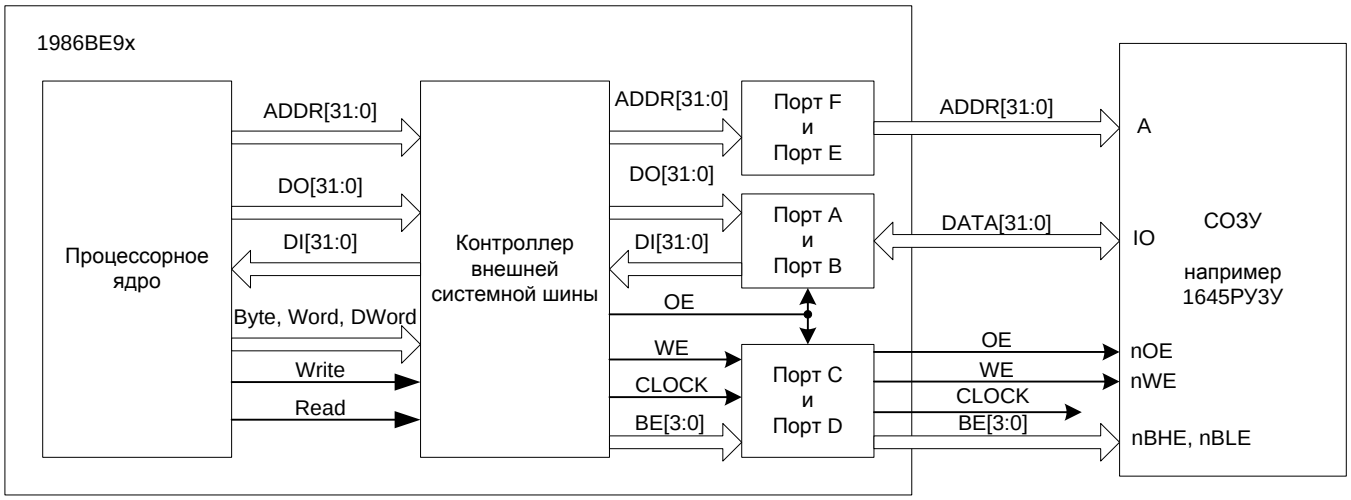


Рисунок 37. Обмен по внешней системной шине при задании длительности через биты WAIT_STATE

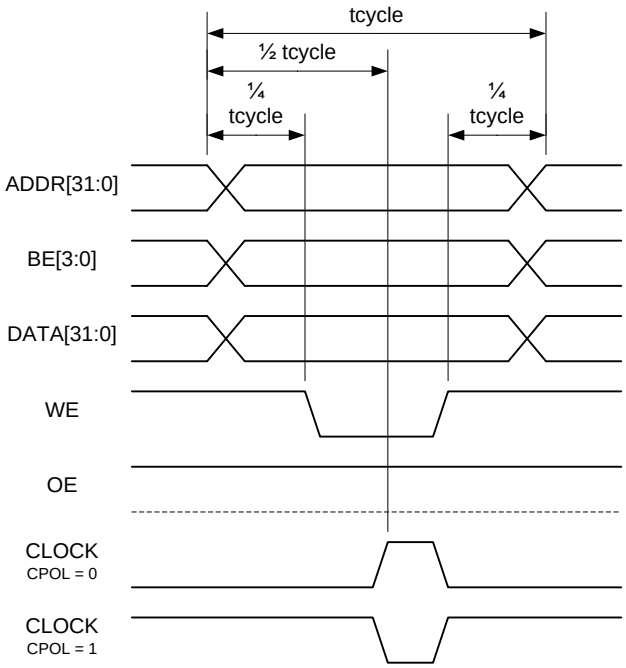


Рисунок 38. Диаграмма записи

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.

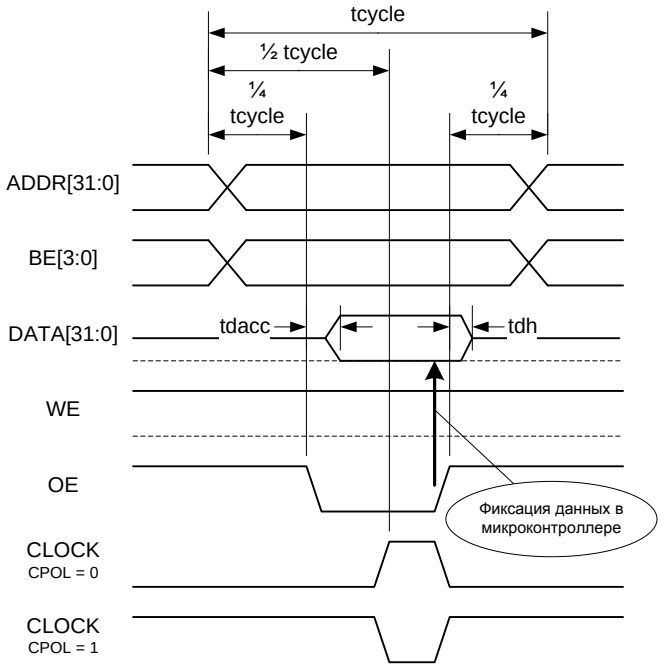


Рисунок 39. Диаграмма чтения

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 153 – Длительность фаз обращения в тактах процессора при EXT_BUS_CTRL_MODE=0 или ENABLE=0 для соответствующего региона

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

19.1.1 Обмен по внешней системной шине при задании длительности через WS_ACTIVE, WS_SETUP, WS_HOLD

Каждый адресный диапазон имеет собственный регистр конфигурации контроллера внешней шины REGx_CNTRL, содержащий следующие основные поля:

WS_SETUP [2:0] – задает время предустановки сигналов WE/OE в цикле записи/чтения, выраженное в количестве тактов системной частоты (WS_SETUP + 1) в диапазоне от 1 до 8;

WS_ACTIVE [6:0] – задает длительность низкого уровня сигналов WE/OE в цикле записи/чтения, выраженное в количестве тактов системной частоты (WS_ACTIVE + 1) в диапазоне от 1 до 128;

WS_HOLD [2:0] – задает время удержания сигналов WE/OE в цикле записи/чтения, выраженное в количестве тактов системной частоты (WS_HOLD + 1) в диапазоне от 1 до 8;

ENABLE – бит разрешения использования собственных настроек длительности:

0 – для определения длительности используется биты WAIT_STATE;

1 – при EXT_BUS_CNTRL_MODE = 1 для определения длительности используется биты WS_ACTIVE, WS_SETUP и WS_HOLD.

19.2 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.

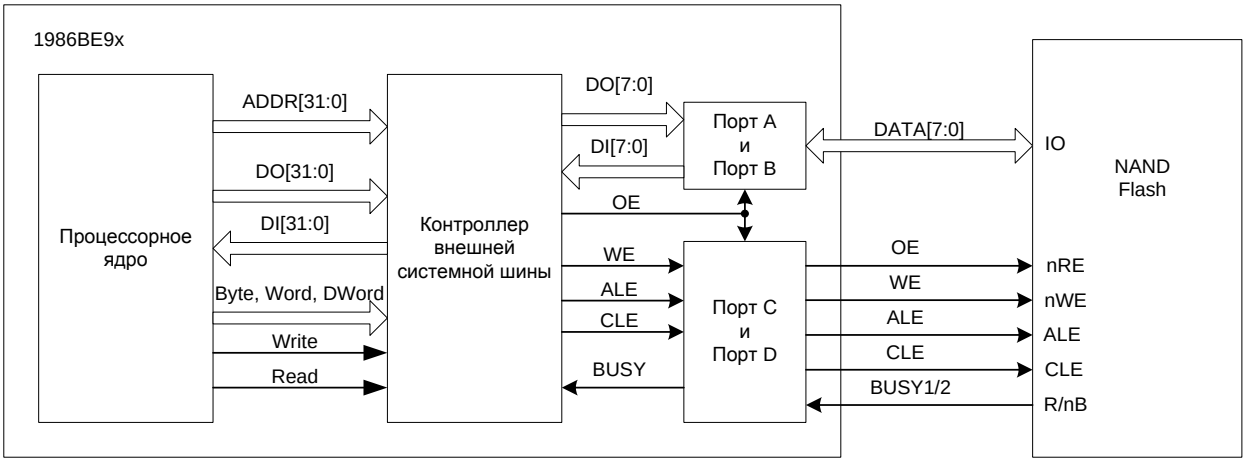


Рисунок 40. Подключение внешней NAND Flash

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 BUSY2 – 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 в Таблица 146.

Таблица 154 – 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
```

19.3 Описание регистров блока контроллера внешней системной шины

Таблица 155 – 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
0x58	REG0_CNTRL	Регистр MDR_EBC->REG0_CNTRL управления регионом 0 внешней системной шиной
0x5C	REG1_CNTRL	Регистр MDR_EBC->REG1_CNTRL управления регионом 1 внешней системной шиной
0x60	REG2_CNTRL	Регистр MDR_EBC->REG2_CNTRL управления регионом 2 внешней системной шиной
0x64	REG3_CNTRL	Регистр MDR_EBC->REG3_CNTRL управления регионом 3 внешней системной шиной

19.3.1 MDR_EBC->CONTROL

В таблицах 156 и 157 цветом обозначены биты в микросхемах 1986BE94, которые отличаются от битов в микросхемах 1986BE91, 986BE92 и 986BE93:

- Красным цветом – добавленные биты управления;
- Зеленым цветом – добавленные статусные биты.

Таблица 156 – Регистр CONTROL

Номер	31…18	17	16	15...12	11	110	9	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	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Срос	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
	-	FF_RDY	FS_RDY	**WAIT_STATE [3:0]**	EN_FF_RDY	EN_FS_RDY	**EXT_BUS_ CNTRL_MODE**	**CLOCKOUT_EN**	**BUSY**	**LOW16**	**LOW8**	**ENDIAN**	**CPOL**	**NAND**	**RAM**	**ROM**

Таблица 157 – Описание бит регистра CONTROL

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31...16	-	Зарезервировано
17	FF_RDY	Флаг возникновения ошибки на шине при отсутствии сигнала READY за отведенное время ожидания (очищается записью 1) 0 – нет ошибки 1 – есть ошибка
16	FS_RDY	Флаг возникновения ошибки на шине при отсутствии сигнала READY перед началом транзакции (очищается записью 1) 0 – нет ошибки 1 – есть ошибка
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	EN_FF_RDY	Бит разрешения формирования ошибки на шине при отсутствии сигнала READY за отведенное время ожидания (имеет смысл при работе с диапазоном, у которого выставлен бит USE_READY=1) 0 – не формировать ошибку BUS FAULT при отсутствии сигнала READY после превышения максимального времени ожидания 1 – формировать ошибку BUS FAULT при отсутствии сигнала READY после превышения максимального времени ожидания

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
10	EN_FS_RDY	Бит разрешения формирования ошибки на шине при отсутствии сигнала READY перед началом транзакции (имеет смысл при работе с диапазоном, у которого выставлен бит USE_READY=1) 0 – не формировать ошибку BUS FAULT при отсутствии сигнала READY переел началом транзакции 1 – формировать ошибку BUS FAULT при отсутствии сигнала READY переел началом транзакции
9	EXT_BUS_MODE	Общий бит разрешения работы с собственными настройками временных интервалов 0 – настройки задаются битами WAIT_STATE 1 – если для региона установлен ENABLE=1, то временные настройки для этого региона определяются битами WS_ACTIVE, WS_SETUP, WS_HOLD
8	CLOCKOUT_EN	Бит разрешения выдачи системной частоты микроконтроллера на вывод CLOCK 0 – выдается единичный сигнал тактирования внутри фазы WE или OE. 1 – при EXT_BUS_MODE=1 выдается системная частота
7	BUSY	Сигнал занятости NAND Flash-памяти: 1 – операция завершена; 0 – операция не завершена
6	LOW16	Выравнивание данных по 16 младшим разрядам внешней системной шины PA[15:0] 0 – обычный режим работы шины 1 – данные записываются и читаются всегда с 16 младших разрядов данных PA[15:0] Чтение/запись 8-ми бит не выполняется При чтении/записи 16-ти бит, выполняется одно обращение При чтении/записи 32-х бит, выполняется два обращения
5	LOW8	Выравнивание данных по 8 младшим разрядам внешней системной шины PA[7:0] 0 – обычный режим работы шины 1 – данные записываются и читаются всегда с 8 младших разрядов данных PA[7:0] Чтение/запись 16-ти бит не выполняется При чтении/записи 8-ми бит, выполняется одно обращение При чтении/записи 32-х бит, выполняется три обращения
4	ENDIAN	Всегда записывать ноль
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

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

19.3.2 MDR_EBC->NAND_CYCLES

Таблица 158 – 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 159 – 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

19.3.3 MDR_EBC->REGx_CNTRL

Красным цветом обозначены добавленные биты управления в микросхемах 1986BE94, которых нет в микросхемах 1986BE91, 986BE92 и 986BE93.

Номер	31…15	14	13…11	10…8	7…1	0
Доступ	U	R/W	R/W	R/W	R/W	R/W
Срос		0	010	010	100000	0
	-	**USE_READY**	**WS_HOLD**	**WS_SETUP**	**WS_ACTIVE**	**ENABLE**

№	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31..15		Зарезервировано
14	USE_READY	Разрешение опроса внешнего сигнала READY на выводе PC[0], динамически определяющего аппаратные состояния ожидания в цикле обмена по внешней системной шине. 1 – опрашивается 0 – не опрашивается Опрос сигнала READY производится на последнем такте фазы ACTIVE, если READY находится в активном состоянии – осуществляется переход к фазе HOLD и завершение цикла обмена, в противном случае производится повторение опроса на каждом последующем такте, пока количество тактов ожидания не превысит 256. **После этого обмен завершается в любом случае**
13..11	WS_HOLD[2:0]	Время удержания сигналов nWE/nOE, выраженное в количестве тактов системной частоты в диапазоне от 1 до 8 плюс один такт
10..8	WS_SETUP[2:0]	Время предустановки сигналов nWE/nOE в цикле записи/чтения, выраженное в количестве тактов системной частоты в диапазоне от 1 до 8 плюс один такт
7..1	WS_ACTIVE[6:0]	Длительность низкого уровня сигналов nWE/nOE в цикле записи/чтения, выраженное в количестве тактов системной частоты в диапазоне от 1 до 128 плюс один такт
0	ENABLE	Разрешение настройки параметров обмена соответствующего диапазона адресов, имеет значение только при EXT_BUS_CNTRL_MODE = 1 1 – разрешена 0 – запрещена

20 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.

20.1 Инициализация контроллера при включении

При включении питания в первую очередь должны быть заданы параметры тактового сигнала блока USB. Параметры задаются в блоке «Сигналы тактовой частоты». 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).

20.2 Задание параметров шины USB и события подключения/отключения

Контроллер USB может быть сконфигурирован как USB Host или как USB Device. Конфигурация задается битом HOST_MODE в регистре HSCR (0 – режим Device, 1 – режим Host). Прием/передача через физический интерфейс USB разрешаются установкой бит EN_RX и EN_TX в этом же регистре. В режиме приема имеется возможность отключить передатчик в целях экономии потребления (EN_TX=0). Отключение всего блока в целом осуществляется при EN_RX=0.

В режиме Device параметры шины задаются в регистре SC. Скорость задается битом SCFSR (0 – 1,5 Мбит/с, 1 – 12 Мбит/с), полярность битом SCFSP (0 – Low speed, 1 – Full speed) этого регистра.

В режиме Host параметры шины задаются в регистре HTXLC. Скорость задается битом FSLR (0 – 1,5 Мбит/с, 1 – 12 Мбит/с), полярность битом FSPL (0 – Low speed, 1 – Full speed) этого регистра.

В режиме Host контроллер автоматически определяет подключение или отключение устройства к шине. Бит CONEV регистра USB_HSI устанавливается в 1 при возникновении одного из событий.

20.3 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.

20.4 Транзакция IN (USB Device)

Если оконечная точка не готова (бит EPRDY = 0 в регистре SEP[x].CTRL), то контроллер отправляет NAK пакет (Рисунок 41 а). Бит NAKSENT регистра SEP[x].STS устанавливается в 1.

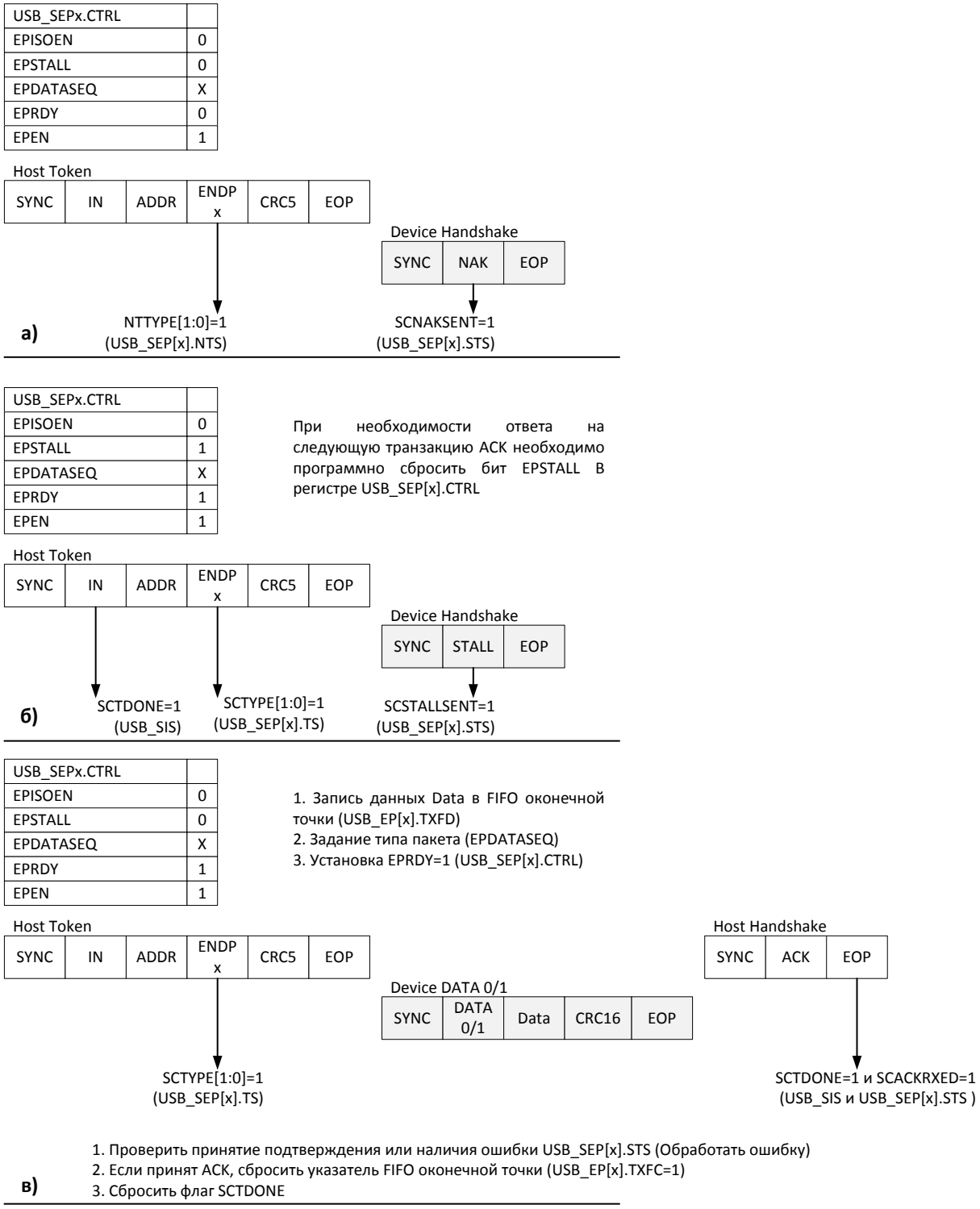


Рисунок 41. Транзакция IN (USB Device)
а – оконечная точка не готова;
б – установлен бит EPSTALL;
в – оконечная точка готова

Если оконечная точка готова и установлен бит EPSSTALL в регистре SEP[x].CTRL, то контроллер отправляет STALL пакет (Рисунок 41 б). Бит SCSTALLSENT регистра SEP[x].STS устанавливается в 1.

Если оконечная точка готова (Рисунок 41 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.

20.5 Транзакция SETUP/OUT (USB Device)

Если оконечная точка не готова (бит EPRDY = 0 в регистре SEP[x].CTRL), то контроллер отправляет NAK пакет (Рисунок 42 а). Бит NAKSENT регистра SEP[x].ST устанавливается в 1.

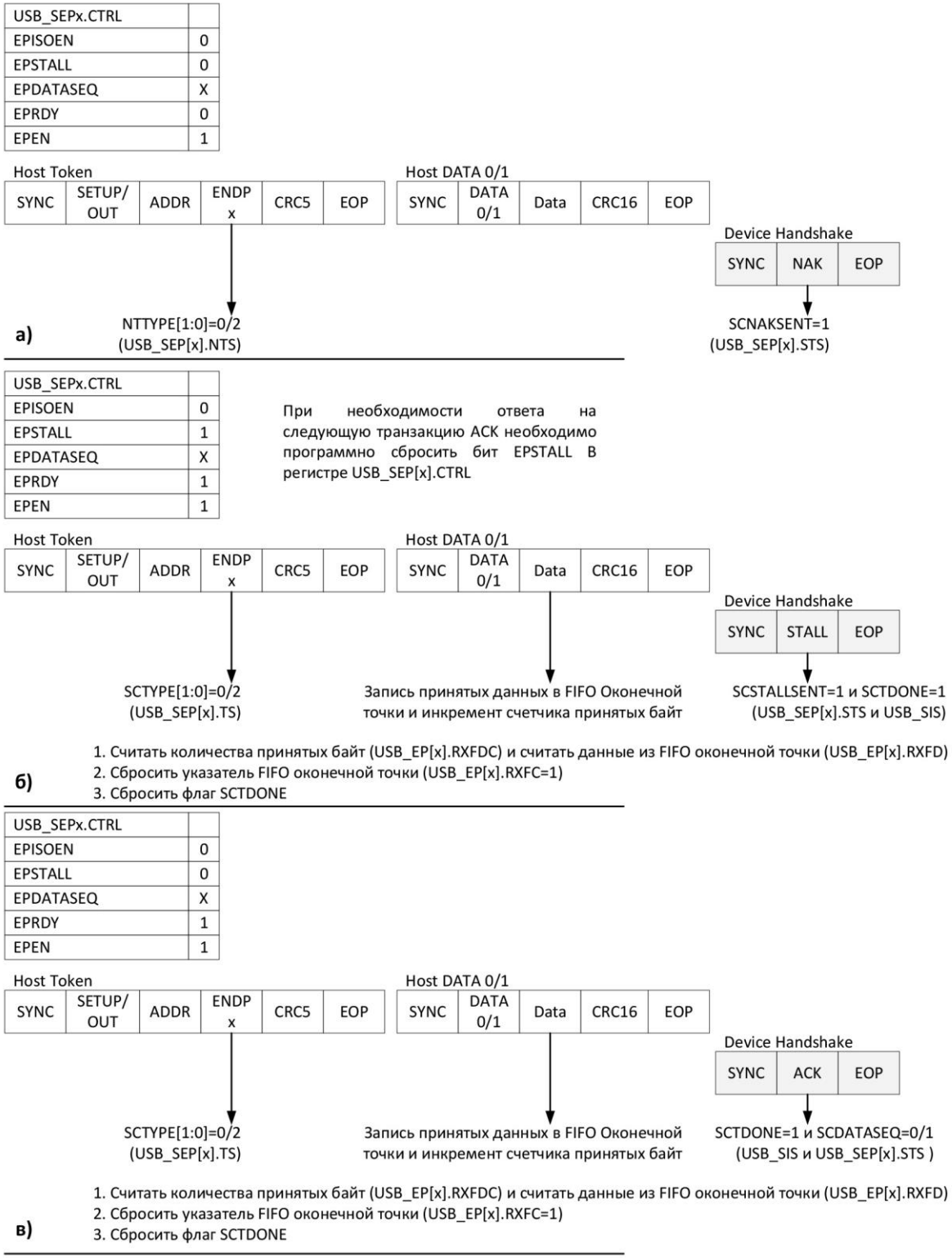


Рисунок 42. Транзакция SETUP/OUT (USB Device)

а – оконечная точка не готова;
б – установлен бит EPSTALL;
в – оконечная точка готова

Если оконечная точка готова и установлен бит EPSSTALL в регистре SEP[x].CTRL, то контроллер отправляет STALL пакет (Рисунок **42** б). Бит SCSTALLSENT регистра SEP[x].STS устанавливается в 1.

Если оконечная точка готова (Рисунок **42** 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.

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.

20.6 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 HTXFC 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 HSI). Этот бит перед началом каждой транзакции должен быть очищен записью 1. PID принятого пакета записывается в регистре HRXP.

Если в ответ получен пакет NAK (Рисунок 43 а), то бит NAKRXED = 1 (регистр HRXS).

Если в ответ получен пакет STALL (Рисунок 43 б), то бит STALLRXED = 1 (регистр HRXS).

Если в ответ получен пакет ACK (Рисунок 43 в), то бит ACKRXED = 1 (регистр HRXS).

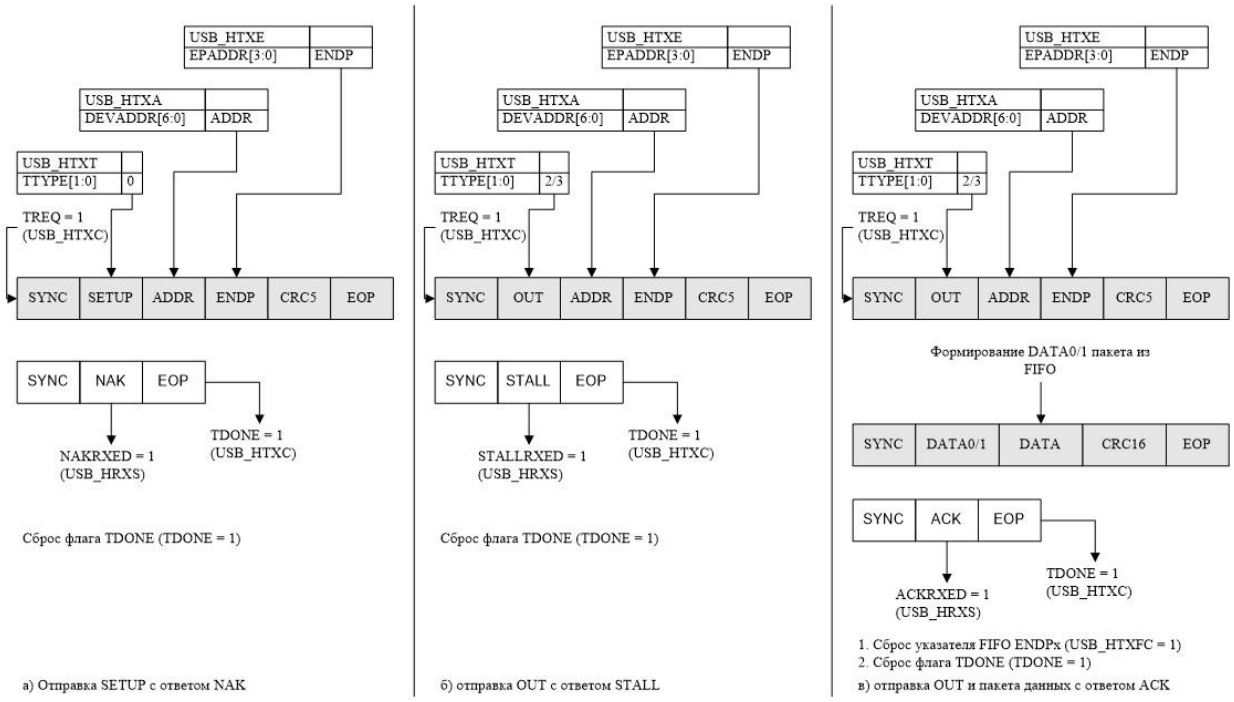


Рисунок 43 (а, б, в). Транзакция SETUP/OUT (USB Host)

20.7 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 HSI). Этот бит перед началом каждой транзакцией должен быть очищен записью 1. PID принятого пакета записывается в регистре HRXP.

Если в ответ получен пакет NAK (Рисунок 44 а), то бит NAKRXED = 1 (регистр HRXS).

Если в ответ получен пакет STALL (Рисунок 44 б), то бит STALLRXED = 1 (регистр HRXS).

Если приходит DATA0/DATA1 пакет (Рисунок 44 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).

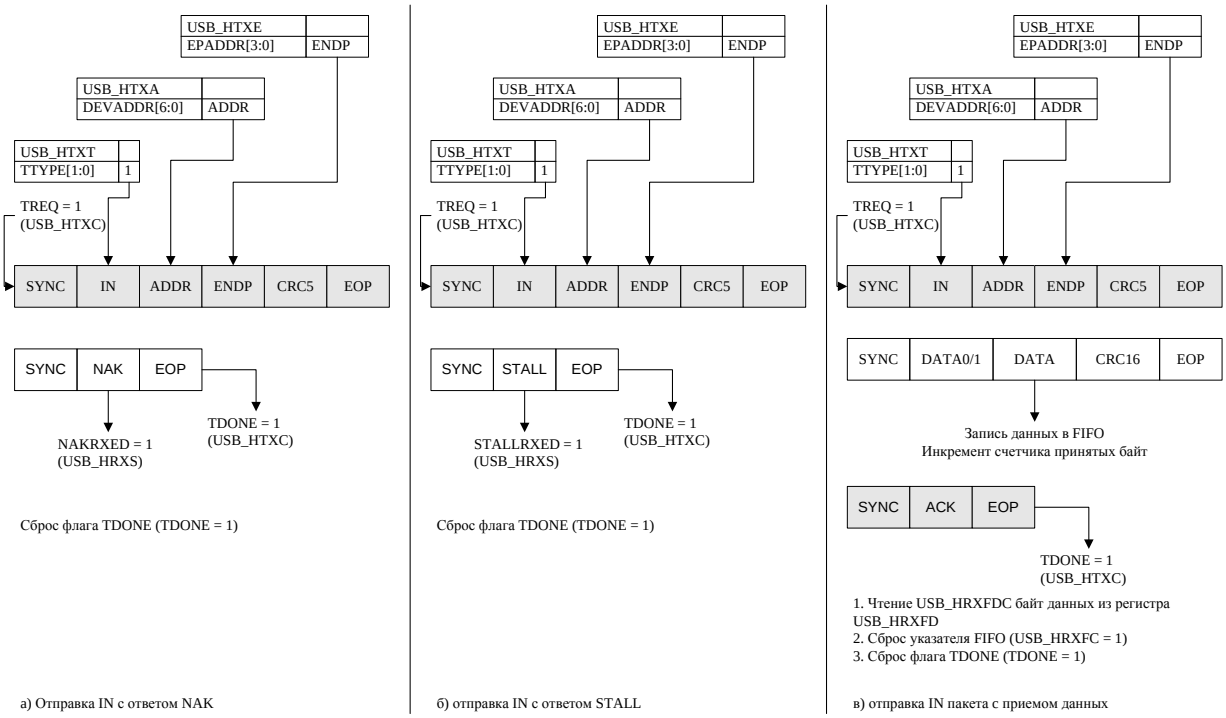


Рисунок 44 (а, б, в). Транзакция IN (USB Host)

20.8 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 HSI). Этот бит должен быть очищен записью 1.

Контроллер ведет счет времени во фрейме таймером. Таймер увеличивается на частоте 48 МГц и имеет 48000 тактов в 1 мс фрейме. Старший байт таймера содержится в регистре HSTM. Этот регистр может быть использован для вычисления, оставшегося во фрейме времени.

20.9 Описание регистров управление контроллером USB интерфейса

Таблица 160 – 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->HSI	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

Базовый Адрес	Название	Описание
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->HRXFDC_L MDR_USB->HRXFDC_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[x].CTRL	Управление очередью нулевой оконечной точки
0x104 0x114 0x124 0x134	MDR_USB->SEP[x].STS	Состояние оконечной точки
0x108 0x118 0x128 0x138	MDR_USB->SEP[x].TS	Состояние типа передачи оконечной точки
0x10C 0x11C 0x12C 0x13C	MDR_USB->SEP[x].NTS	Состояние передачи NAK оконечной точки
0x140	MDR_USB->SC	Управление контроллеров SLAVE
0x144	MDR_USB->SLS	Отображение состояния линий USB шины
0x148	MDR_USB->SIS	Флаги событий контроллера SLAVE
0x14C	MDR_USB->SIM	Флаги разрешения прерываний от контроллера SLAVE
0x150	MDR_USB->SA	Функциональный адрес контроллера
0x154 0x158	MDR_USB->SFN_L MDR_USB->SFN_H	Номер фрейма

Базовый Адрес	Название	Описание
0x180 0x200 0x280 0x300	MDR_USB->SEP[x].RXFD	Принятые данные оконечной точки
0x188 0x18C 0x208 0x20C 0x288 0x28C 0x308 0x30C	MDR_USB->SEP[x].RXFDC_L MDR_USB->SEP[x].RXFDC_H	Число данных в оконечной точке
0x190 0x210 0x290 0x310	MDR_USB->SEP[x].RXFC	Управление очередью приема оконечной точки
0x1C0 0x240 0x2C0 0x340	MDR_USB->SEP[x].TXFD	Данные для передачи через оконечную точку
0x1D0 0x250 0x2D0 0x350	MDR_USB->SEP[x].TXFDC	Управление очередью передачи оконечной точки

20.9.1 MDR_USB->HSCR

Таблица 161 – 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 162 – 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

20.9.2 MDR_USB->HSVR

Таблица 163 – Регистр HSVR

Номер	31...8	7...4	3...0
Доступ	U	RO	RO
Сброс	0	0	0
	-	**REVISION**	**VERSION**

Таблица 164 – 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

20.9.3 Регистры HOST режима

20.9.3.1 MDR_USB->HTXC

Таблица 165 – 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 166 – 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 с 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

20.9.3.2 MDR_USB->HTXT

Таблица 167 – Регистр HTXT

Номер	31...2	1	0
Доступ	U	R/W	R/W
Сброс	0	0	0
	-	**TTYPE**	

Таблица 168 – 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

20.9.3.3 MDR_USB->HTXLC

Таблица 169 – 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 170 – 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/с 0 – 1,5 Мбит/с
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+

20.9.3.4 MDR_USB->HTXSE

Таблица 171 – Регистр HTXSE

Номер	31…1	0

Доступ	U	R/W
Сброс	0	0
	-	**SOFEN**

Таблица 172 – 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 состояния FSPL. Если FSPL не установлен, то автоматически будет отправляться EOP каждые 1 мс. Это необходимо при работе с low speed устройством напрямую (не через хаб). 0 – запрет автоматической отправки SOF/EOP и позволяет подсоединенным устройствам перейти в suspend режим

20.9.3.5 MDR_USB->HTXA

Таблица 173 – Регистр HTXA

Номер	31...7	6...0
Доступ	U	R/W
Сброс	0	0
	-	**DEVADDR[6:0]**

Таблица 174 – 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

20.9.3.6 MDR_USB->HTXE

Таблица 175 – Регистр HTXE

Номер	31...4	3...0
Доступ	U	R/W
Сброс	0	0
	-	**EPADDR[3:0]**

Таблица 176 – 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

20.9.3.7 MDR_USB->HFN

Таблица 177 – Регистр HFN

Номер	31...11	10…0
Доступ	U	R/W
Сброс	0	0
	-	FNUM[10:0]

Таблица 178 – 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

20.9.3.8 MDR_USB->HSI

Таблица 179 – Регистр HSI

Номер	31...4	3	2	1	0
Доступ	U	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0
	-	**SOFS**	**CONEV**	**RESUME**	**TDONE**

Таблица 180 – Описание бит регистра HSI

№ 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

20.9.3.9 MDR_USB->HIM

Таблица 181 – 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 182 – 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 прерывания при передаче SOF. 0 – запрещение выработки прерывания
2	CONEVIE	1 – разрешение выработки прерывания при подсоединении или отсоединении. 0 – запрещение выработки прерывания
1	RESUMEIE	1 – разрешение выработки прерывания при повторе передачи. 0 – запрещение выработки прерывания
0	TDONEIE	1 – разрешение выработки прерывания при окончании передачи. 0 – запрещение выработки прерывания

20.9.3.10 MDR_USB->HRXS

Таблица 183 – 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
	-	DATA SEQ	ACK RXED	STALL RXED	NAK RXED	RX TO	RXOF	BSERR	CRCER

Таблица 184 – 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

20.9.3.11 MDR_USB->HRXP

Таблица 185 – Регистр HRXP

Номер	31...4	3…0
Доступ	U	R/W
Сброс	0	0
	-	**RPID[3:0]**

Таблица 186 – 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

20.9.3.12 MDR_USB->HRXA

Таблица 187 – Регистр HRXA

Номер	31…7	6…0
Доступ	U	R/W
Сброс	0	0
	-	**RADDR[6:0]**

Таблица 188 – 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

20.9.3.13 MDR_USB->HRXE

Таблица 189 – Регистр HRXE

Номер	31...4	3...0
Доступ	U	R/W
Сброс	0	0
	-	**RXENDP[3:0]**

Таблица 190 – 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

20.9.3.14 MDR_USB->HRXCS

Таблица 191 – Регистр HRXCS

Номер	31...2	1	0
Доступ	U	R/W	R/W
Сброс	0	0	0
	-	**RXLS[1:0]**	

Таблица 192 – 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

20.9.3.15 MDR_USB->HSTM

Таблица 193 – Регистр HSTM

Номер	31...8	7...0
Доступ	U	R/W
Сброс	0	0
	-	**HSTM[7:0]**

Таблица 194 – 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

20.9.3.16 MDR_USB->HRXFD

Таблица 195 – Регистр HRXFD

Номер	31...8	7...0
Доступ	U	R/W
Сброс	0	0
	-	**RX FIFO DATA[7:0]**

Таблица 196 – 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

20.9.3.17 MDR_USB->HRXFDC

Таблица 197 – Регистр HRXFDC

Номер	31...16	15...0
Доступ	U	R/W
Сброс	0	0
	-	**FIFO DATA COUNT[15:0]**

Таблица 198 – Описание бит регистра HRXFDC

№ 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

20.9.3.18 MDR_USB->HRXFC

Таблица 199 – Регистр HRXFC

Номер	31...1	0
Доступ	U	R/W
Сброс	0	0
	-	**FIFO FORCE EMPTY**

Таблица 200 – 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

20.9.3.19 MDR_USB->HTXFD

Таблица 201 – Регистр HTXFD

Номер	31...8	7...0
Доступ	U	R/W
Сброс	0	0
	-	**TX FIFO DATA[7:0]**

Таблица 202 – 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

20.9.3.20 MDR_USB->HTXFC

Таблица 203 – Регистр HTXFC

Номер	31...1	0
Доступ	U	R/W
Сброс	0	0
	-	**FIFO FORCE EMPTY**

Таблица 204 – 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

20.9.4 USB Slave (Device)

20.9.4.1 MDR_USB->SEP[x].CTRL

Таблица 205 – 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 206 – 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

20.9.4.2 MDR_USB->SEP[x].STS

Таблица 207 – 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 208 – 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

20.9.4.3 MDR_USB->SEP[x].TS

Таблица 209 – Регистр SEP[x].TS

Номер	31…2	1	0
Доступ	U	R/W	R/W
Сброс	0	0	0
	-	**SCTTYPE[1:0]**	

Таблица 210 – 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

20.9.4.4 MDR_USB->SEP[x].NTS

Таблица 211 – Регистр SEP[x].NTS

Номер	31…2	1	0
Доступ	U	R/W	R/W
Сброс	0	0	0
	-	**NTTYPE[1:0]**	

Таблица 212 – 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

20.9.4.5 MDR_USB->SC

Таблица 213 – 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 214 – 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

20.9.4.6 MDR_USB->SLS

Таблица 215 – Регистр SLS

Номер	31…2	1	0
Доступ	U	R/W	R/W
Сброс	0	0	0
	-	**SCRXLS[1:0]**	

Таблица 216 – 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

20.9.4.7 MDR_USB->SIS

Таблица 217 – Регистр SIS

Номер	31…6	5	4	3	2	1	0
Доступ	U	U	R/W	R/W	R/W	R/W	R/W
Сброс	0	1	0	0	0	0	0
	-	-	SC NAK SENT	SC SOF REC	SC RESET EV	SC RESUME	SC TDONE

Таблица 218 – Описание бит регистра USB_SIS

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31...6	-	Зарезервировано
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

20.9.4.8 MDR_USB->SIM

Таблица 219 – 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 220 – 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

20.9.4.9 MDR_USB->SA

Таблица 221 – Регистр SA

Номер	31…7	6…0
Доступ	U	R/W
Сброс	0	0
	-	**SDEVADDR[6:0]**

Таблица 222 – 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

20.9.4.10 MDR_USB->SFN

Таблица 223 – Регистр SFN

Номер	31…11	10…0
Доступ	U	R/W
Сброс	0	0
	-	**FRAME NUM [10:0]**

Таблица 224 – 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

20.9.4.11 MDR_USB->SEP[x].RXFD

Таблица 225 – Регистр SEP[x].RXFD

Номер	31…8	7…0
Доступ	U	R/W
Сброс	0	0
	-	**RX FIFO DATA[7:0]**

Таблица 226 – 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

20.9.4.12 MDR_USB->SEP[x].RXFDC

Таблица 227 – Регистр SEP[x].RXFDC

Номер	31…16	15…0
Доступ	U	R/W
Сброс	0	0
	-	**FIFO DATA COUNT [15:0]**

Таблица 228 – 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

20.9.4.13 MDR_USB->SEP[x].RXFC

Таблица 229 – Регистр SEP[x].RXFC

Номер	31…1	0
Доступ	U	R/W
Сброс	0	0
	-	**FIFO FORCE EMPTY**

Таблица 230 – Описание бит регистра SEP[x].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

20.9.4.14 MDR_USB->SEP[x].TXFD

Таблица 231 – Регистр SEP[x].TXFD

Номер	31…8	7…0
Доступ	U	R/W
Сброс	0	0
	-	**TX FIFO DATA[7:0]**

Таблица 232 – 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

20.9.4.15 MDR_USB->SEP[x].TXFDC

Таблица 233 – Регистр SEP[x].TXFDC

Номер	31…1	0
Доступ	U	R/W
Сброс	0	0
	-	**FIFO FORCE EMPTY**

Таблица 234 – 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

21 Контроллер интерфейса MDR_CAN

В микроконтроллере реализовано два независимых контроллера интерфейса CAN. Они являются полнофункциональными CAN-узлами, отвечающими требованиям к активным и пассивным устройствам CAN 2.0A и 2.0B и поддерживающими передачу данных на скорости не более 1 Мбит/сек.

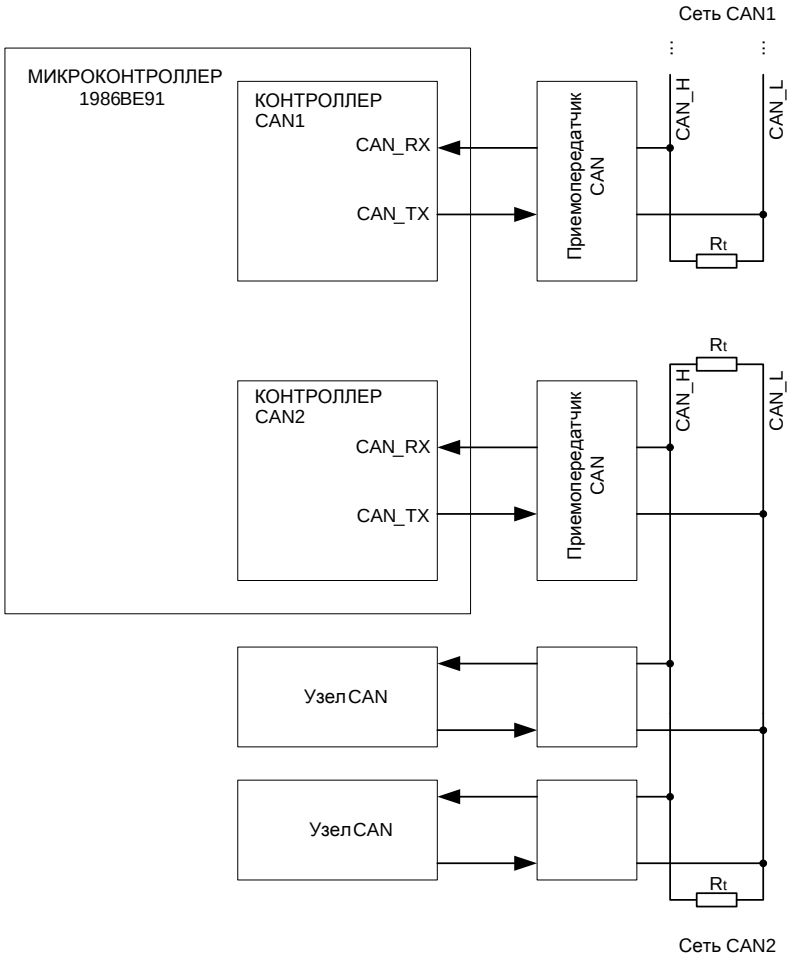


Рисунок 45. Структурная блок – схема организации сети CAN

Интерфейс CAN позволяет обмениваться сообщениями в сети равноправных устройств. При передаче сообщения в сети CAN все узлы сети получают это сообщение. В сообщении передается уникальный идентификатор и данные. Все сообщения в протоколе CAN могут содержать не более восьми байтов данных. При возникновении коллизий (одновременная передача сообщений различными узлами) при передаче идентификатора происходит арбитраж, и узел, передающий сообщение с большим номером идентификатора, уступает сеть узлу, передающему сообщение с меньшим номером идентификатора.

Особенности:

- поддержка CAN протокола версии CAN 2.0 A и B;
- скорость передачи до 1 Мбит/с;
- 32 буфера приема/передачи;
- поддержка приоритетов сообщений;
- 32 фильтра приема;
- маскирование прерываний.

21.1 Режимы работы

CAN-контроллер поддерживает несколько режимов работы: нормальный режим для приема и передачи пакетов сообщений, режим работы только на прием, режим самотестирования и режим инициализации для задания параметров связи.

- Режим нормальной передачи
  (регистр CAN_STATUS : ROM = 0, STM = 0)

Выводы CAN_TX и CAN_RX подключены к шине.

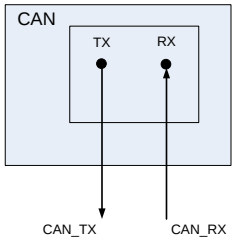


Рисунок 46. Режим нормальной передачи

В этом режиме можно установить флаги разрешения приема своих пакетов и разрешения подтверждения своих пакетов посылкой ACK (регистр CAN_CONTROL поля SAP и ROP).

- Режим работы только на прием - Receive Only Mode
  (регистр CAN_STATUS: ROM = 1, STM = 0)

Контроллер CAN интерфейса принимает, но не посылает никакой информации, т.е. линия TX всегда в «1», но внутри контроллера все управляющие сигналы проходят.

- Режим самотестирования - Self Test Mode
  (регистр CAN_STATUS : STM = 1, ROM = 0)

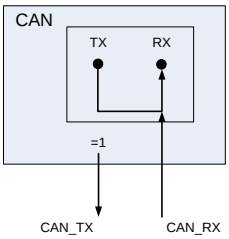


Рисунок 47. Режим работы только на прием - Receive Only Mode

Выводы CAN_TX и CAN_RX отключены, вся передаваемая информация видна только внутри контроллера.

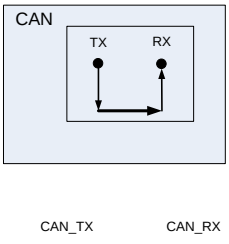


Рисунок 48. Режим самотестирования - Self Test Mode

Для успешного приема своих сообщений необходимо установить флаги разрешения приема своих пакетов и разрешения подтверждения своих пакетов посылкой ACK (регистр CAN_CONTROL поля SAP и ROP). В этом режиме передаваемые сообщения сразу же принимаются в приемный буфер. Режим самотестирования полезен в период отладки кода программы.

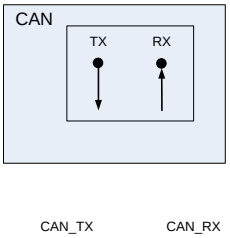


Рисунок 49. Режим инициализации для задания параметров связи

Еще одна важная функция CAN-контроллера – фильтрация получаемых сообщений. Поскольку CAN является широковещательной шиной, каждое переданное сообщение принимается всеми узлами шины. В CAN-шине любой разумной степени сложности передается достаточно большое число сообщений. Задачей каждого подключенного к CAN-узлу ЦПУ является реагирование на CAN-сообщения. Таким образом, чтобы избавить CAN-контроллер от проблемы приема в буфер нежелательных сообщений, необходима их фильтрация. У CAN-контроллера микроконтроллеров 1986BE9x имеется 32 регистра фильтров и 32 регистра масок, которые можно использовать для блокировки всех CAN-сообщений, кроме избранных сообщений или групп сообщений.

21.2 Типы пакетов сообщений

Информация на шине представлена в виде фиксированных сообщений различной, но ограниченной длины. Когда шина свободна, любой подключенный узел может начать передавать новое сообщение. При передаче информации с помощью протокола CAN используется четыре типа пакетов:

- **пакет удаленного запроса данных** передается узлом, чтобы запросить передачу пакета данных с тем же самым идентификатором;
- **пакет ошибки** передается любым узлом при обнаружении ошибочного состояния на шине. Пакет ошибки передается сразу же после обнаружения ошибки и накладывается на передаваемый пакет так, чтобы испортить его окончательно. Таким образом, если один из узлов обнаружил ошибку, он усиливает ошибку для того, чтобы ее обнаружили и другие узлы;
- **пакет перегрузки** используется для обеспечения дополнительной задержки между предшествующим и последующим кадрами данных или кадрами удаленного запроса данных. Он передается в редких случаях, подробнее можно прочесть в стандарте ISO 11898-1. Контроллер CAN интерфейса отсылает пакет перегрузки в соответствии со стандартом;
- основными пакетами на шине CAN являются **пакеты данных.** В пакете данных узел передает данные от передатчика приемнику. Пакеты могут быть стандартными и расширенными. Отличие пакетов заключается в размере полей идентификатора. Пакеты с 11 разрядным идентификатором – называются стандартными пакетами, пакеты, содержащие 29 разрядные идентификаторы, называются расширенными пакетами. При передаче идентификационной информации происходит

автоматический арбитраж на шине CAN таким образом, чтобы пакет с меньшим значением поля ID остался на шине. На шине не допускается одновременная передача двух или более сообщений с одним и тем же идентификатором. Размер передаваемых данных кодируется в поле DLC и может составлять от 0 до 8 байт. После передачи поля данных контроллер автоматически передает рассчитанное значение CRC. Если хотя бы один из узлов принял пакет, то он выставляет ACK подтверждение на шине, если хотя бы один из узлов обнаружит ошибку, то на шину будет выставлен пакет ошибки. Таким образом, гарантируются доставки сообщений.

Пакеты данных и пакеты удаленного запроса данных отделяются от предшествующих пакетов межкадровым пространством.

21.3 Структура пакета данных (Data Frame)

Пакет данных состоит из 7 различных полей:

- "начало пакета" (SOF-start of frame);
- "поле арбитража" (arbitration field);
- "поле контроля" (control field);
- "поле данных" (data field);
- "поле CRC" (CRC field);
- "поле подтверждения" (ACK field);
- "конец пакета" (end of frame).

Поле данных может иметь нулевую длину.

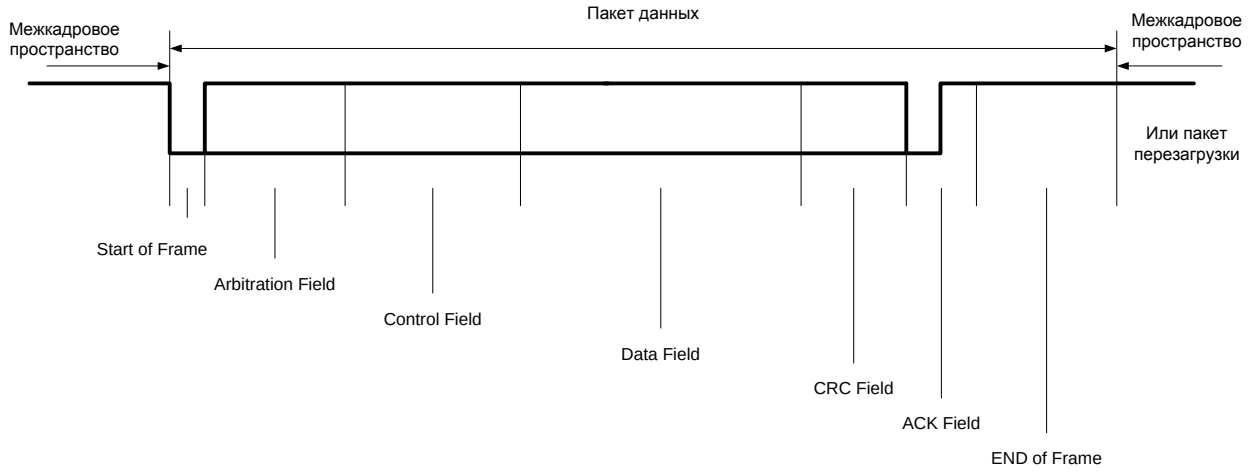


Рисунок 50. Пакет сообщения CAN

В терминах протокола CAN логическая единица называется рецессивным битом, а логический ноль называется доминантным битом. Во всех случаях доминантный бит будет подавлять рецессивный. То есть, если несколько узлов выставят на шину рецессивный бит, а один – доминантный, то обратно всеми узлами будет считан доминантный бит.

21.3.1 Начало пакета (Start of frame)

Начало пакета отмечает начало пакета данных или пакета удаленного запроса данных. Это поле состоит их одиночного доминантного бита. Узлу разрешено начать передачу, когда шина свободна. Все узлы должны синхронизироваться по фронту, вызванному передачей поля «начало пакета» узла, начавшего передачу первым.

21.3.2 Поле арбитража (Arbitration field)

Формат поля арбитража отличается для стандартного и расширенного форматов:

– в стандартном формате поле арбитража состоит из 11 разрядного идентификатора и RTR-бита;

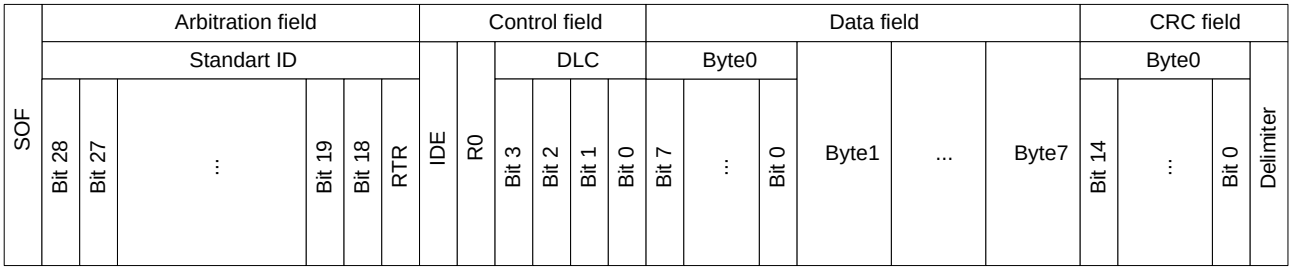


Рисунок 51. Структура стандартного пакета данных

– в расширенном формате поле арбитража состоит из 29 разрядного идентификатора, SRR-бита, IDE-бита и RTR-бита.

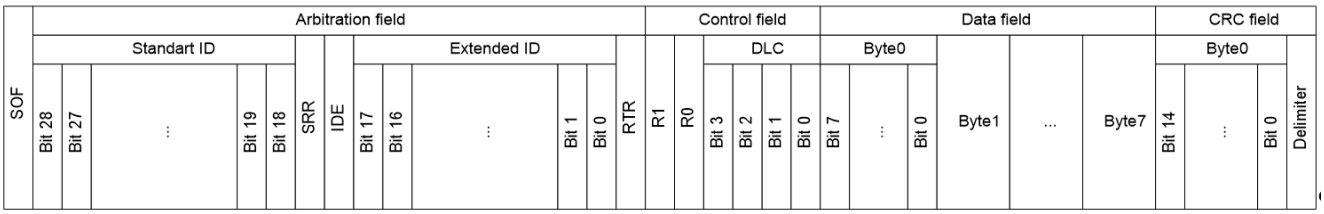


Рисунок 52. Структура расширенного пакета данных

21.3.2.1 Идентификатор

Идентификатор – стандартный формат. Длина идентификатора - 11 бит и соответствует Standart ID в расширенном формате. Эти биты передаются в порядке Bit28 … Bit18. Самый младший бит - Bit18. 7 старших бит (Bit28 - Bit 22) не должны быть все единичными битами.

Идентификатор – расширенный формат. В отличие от стандартного идентификатора, расширенный идентификатор состоит из 29 бит. Его формат содержит две секции:

– Standart ID - 11 бит
– Extended ID - 18 бит

Standart ID состоит из 11 бит. Эта секция передается в порядке от Bit28 … Bit18. Это эквивалентно формату стандартного идентификатора. Standart ID определяет базовый приоритет расширенного пакета.

Extended ID состоит из 18 бит. Эта секция передается в порядке от Bit17 до Bit0. В стандартном пакете идентификатор сопровождается RTR битом.

21.3.2.2 Бит RTR

Бит запроса удаленной передачи. В пакетах данных RTR бит должен быть передан нулевым уровнем. Внутри пакета удаленного запроса данных RTR бит должен быть единичным. В расширенном пакете сначала передается Standart ID, с последующими битами IDE и SRR. Extended ID передается после SRR бита.

21.3.2.3 Бит SRR (расширенный формат)

Заменитель бита удаленного запроса. SRR - единичный бит. Он передается в расширенных пакетах в позиции RTR бита. Таким образом, он заменяет RTR - бит стандартного пакета.

Следовательно, при одновременной передаче стандартного пакета и расширенного пакета, Standart ID которого совпадает с идентификатором стандартного пакета, стандартный пакет преобладает над расширенным пакетом.

21.3.2.4 Бит IDE (расширенный формат)

Бит расширения идентификатора

IDE бит принадлежит:

- полю арбитража для расширенного формата;
- полю управления для стандартного формата.

IDE бит в стандартном формате передается нулевым уровнем, в расширенном формате IDE бит – единичный уровень.

21.3.3 Поле управления (Control field)

Поле управления состоит из шести бит. Формат поля управления отличается для стандартного и расширенного формата.

Пакеты в стандартном формате включают: код длины данных (DLC), бит IDE, который передается нулевым уровнем (см. выше), и зарезервированный бит r0.

Пакеты в расширенном формате включают код длины данных и два зарезервированных бита r1 и r0. Зарезервированные биты должны быть посланы нулевым уровнем, но приемники принимают единичные и нулевые уровни биты во всех комбинациях.

Код длины данных (Data length code)

Число байт в поле данных обозначается кодом длины данных. Этот код длины данных, размером 4 бита, передается внутри поля управления. Допустимое число байт данных: {0,1, ..., 7,8}. Другие величины использоваться не могут.

21.3.4 Поле данных (Data field)

Поле данных состоит из данных, которые будут переданы внутри пакета данных. Оно может содержать от 0 до 8 байт, каждый содержит 8 бит, которые передаются, начиная со старшего значащего бита.

21.3.5 Поле CRC (CRC field)

Содержит последовательность CRC и CRC - разделитель. При вычислении 15 битного CRC кода используется последовательность бит, состоящая из полей: "начало пакета", "поле арбитража", "управляющее поле", "поле данных" (если есть). Последовательность CRC сопровождается разделителем CRC, который состоит из одного единичного бита.

21.3.6 Поле подтверждения (ACK field)

Поле подтверждения имеет длину два бита и содержит: "область подтверждения" и разделитель подтверждения. В поле подтверждения передающий узел посылает два бита с единичным уровнем. Приемник, который получил сообщение правильно (CRC соответствует),

сообщает об этом передатчику, посылая бит с нулевым уровнем в течение приема поля "область подтверждения".

21.3.7 Конец пакета (End of frame)

Каждый пакет данных и пакет удаленного запроса данных ограничен последовательностью флагов, состоящей из семи единичных бит.

21.3.8 Структура пакета удаленного запроса данных (Remote frame)

Узел, действующий как приемник некоторых данных, может инициировать передачу соответственных данных исходными узлами, посылая пакет удаленного запроса данных. Пакет удаленного запроса данных существует и в стандартном формате, и в расширенном формате. В обоих случаях он состоит из шести битовых полей:

- "начало пакета" (Start of frame);
- "поле арбитража" (Arbitration field);
- "управляющее поле" (Control field);
- "поле CRC" (CRC - field);
- "поле подтверждения" (ACK field);
- "конец пакета" (End of frame).

В отличие от обычного пакета данных, RTR бит пакета удаленного запроса данных - единичный. В этом пакете отсутствует поле данных. При этом значение кода длины данных может принимать любое значение в пределах допустимого диапазона [0,8]. Значение кода длины данных соответствует коду длины данных кадра данных. RTR бит указывает, является ли переданный кадр кадром данных.

21.3.9 Арбитраж на шине

Арбитраж сообщений гарантирует, что наиболее важное сообщение захватит шину и будет передано без задержки. Затем будут переданы приостановленные сообщения согласно их приоритетам (сообщение с наименьшим идентификатором передается первым).

Если планируется передача сообщения, и шина свободна, то сообщение будет передано и сможет быть принято любым заинтересованным в нем узлом. Если передача сообщения запланирована, а шина активна, то прежде чем приступить к передаче сообщения, необходимо дождаться освобождения шины. Если запланирована передача нескольких сообщений, то при освобождении шины они начнут передаваться одновременно, синхронизируясь по признаку начала пакета. В этом случае на шине начнется процесс арбитража, задача которого – определить, какое именно из сообщений захватит шину и будет передано.

Арбитраж сообщений на шине CAN осуществляется методом, который называется «неразрушающий побитовый арбитраж».

На рисунке изображены три сообщения, ожидающие передачи. После освобождения шины и синхронизации пакетов сообщений по старт-биту на шину начинают выдаваться все три идентификатора. При передаче первых двух бит все три узла выставляют на шину одинаковые логические уровни и соответственно считывают те же значения, поэтому они все продолжают передачу. Однако при передаче третьего бита узлы A и C выставляют на шину доминантный бит, а узел B выставляет рецессивный бит, но при этом считывает с шины доминантный. В результате узел B освобождает шину и начинает следить за ее состоянием. Узлы A и C продолжают передачу, пока ситуация не повторится; теперь узел C выдаёт рецессивный бит, а узел A – доминантный. При этом узел C прекращает передачу и начинает следить за состоянием шины. С этого момента шина захватывается узлом A. После передачи

сообщения узлом A узлы B и C начинают передачу, причем узел C захватит шину и передает свое сообщение. Если бы узлу A снова надо было передавать сообщение, он снова захватил бы шину. Таким образом, первым на шине CAN передается сообщение с наименьшим идентификатором.

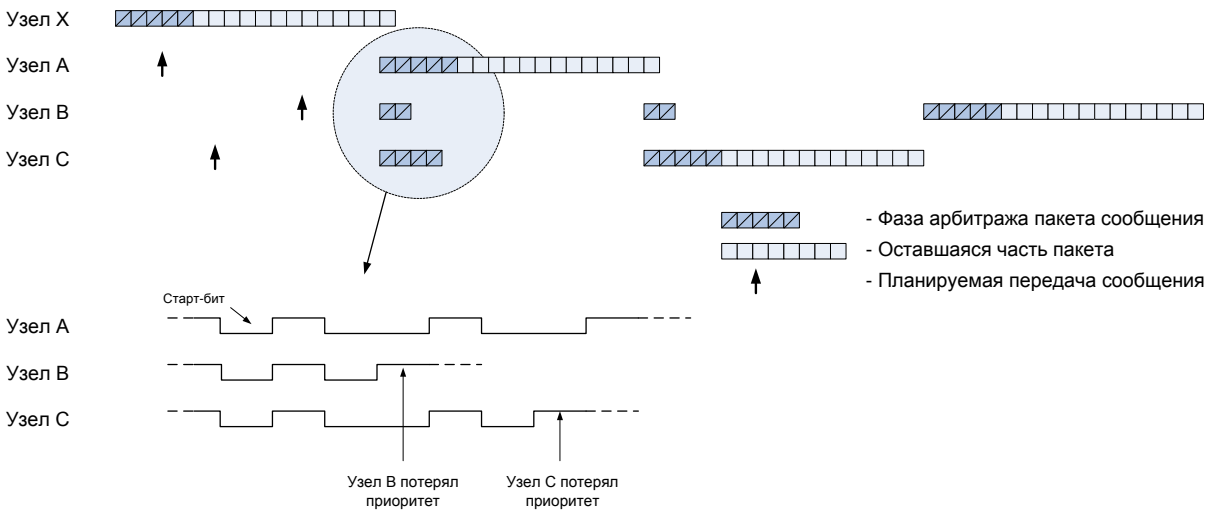


Рисунок 53. Арбитраж на шине CAN

В случае «проигрыша» арбитража в регистре статуса контроллера CAN будет установлен флаг ID_LOWER.

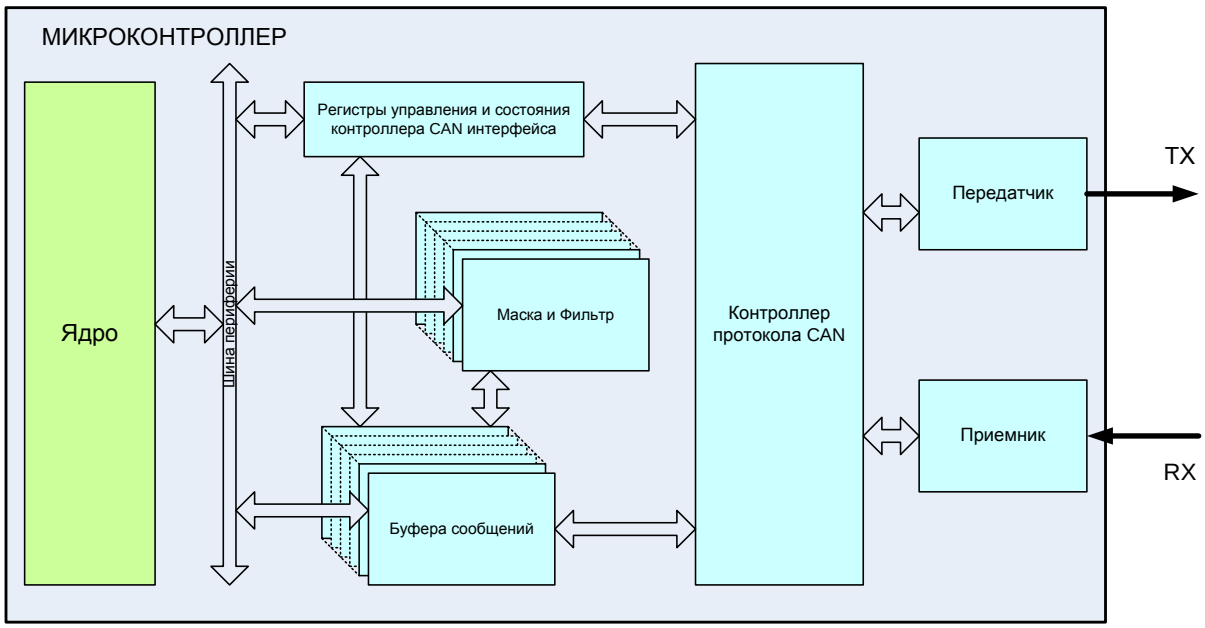


Рисунок 54. Структурная блок-схема контроллера CAN

21.4 Инициализация

Перед началом работы с контроллерами CAN в первую очередь должны быть заданы параметры их тактового сигнала. Параметры задаются в блоке «Сигналы тактовой частоты».

Для задания тактовой частоты блока необходимо установить бит разрешения тактирования блока (бит 0 для CAN1, бит 1 для CAN2 регистра PER_CLOCK). В регистре CAN_CLOCK установить бит CANyCLKEN, чтобы разрешить тактовую частоту для определенного контроллера CAN, задать коэффициент деления тактовой частоты HCLK для каждого CAN контроллера.

После подачи тактового сигнала на блок таймера можно приступать к работе с ним.

Для работы контроллера шины CAN он должен быть настроен на соответствующую скорость шины CAN. Для этого должны быть заданы соответствующим образом поля SB, SJW, SEG2, SEG1, PSEG и BRP в регистре CAN_BITTMNG. После этого должны быть заданы работающие буфера сообщений путем задания бит EN (разрешение работы) RXTXn (1 – прием, 0 – передача) в регистре BUF_xx_CON. После этого должен быть выдан общий сигнал разрешения работы контроллера через задание бита CANEN в регистре CONTROL. После этого контроллер CAN начинает работу.

21.5 Передача сообщений

Для передачи сообщения необходимо в разрешенный для работы и конфигурируемый на передачу буфер записать сообщение для передачи (задать значения регистрам CAN_BUF[x].ID, CAN_BUF[x].DLC, CAN_BUF[x].DATAL и CAN_BUF[x].DATAH), после чего установить бит TX_REQ. После установки этого бита сообщение будет поставлено в очередь на отправку. После отправки сообщения бит TX_REQ будет автоматически сброшен. Если в нескольких буферах есть сообщения на отправку, то порядок отправки определяется по полю PRIOR_0. Если у сообщения бит PRIOR_0 выставлен в ноль, то оно отправляется в первую очередь. Если есть несколько сообщений с одинаковым приоритетом, то порядок отправки определяется порядковым номером буфера, буфер с меньшим порядковым номером имеет больший приоритет. Значение полей ID для выбора порядка отправки в рамках контроллера CAN (одного узла) значения не имеет. По ID выбирается приоритет между различными узлами.

21.6 Передача сообщений по Remote Transmit Request (RTR)

Для автоматической отправки сообщения по запросу Remote Transmit Request необходимо задать режим маскирования для данного буфера таким образом, чтобы он принимал только сообщения от устройства, которое может выслать RTR запрос. В регистре INT_TX при необходимости настроить генерацию прерывания передачи для соответствующего буфера. В регистре управления этим буфером (BUFF_CON[x]) проверить, что флаг TX_REQ = 0, задать приоритет отправляемого сообщения PRIOR_0, установить разрешение ответа при приеме RTR в буфер (RTR_EN=1), задать RX_TX = 0 для разрешения отправки сообщения и задать EN = 1 для разрешения работы буфера. В регистре идентификации задать необходимые SID и EID, в регистре BUF_xx_DLC указать формат пакета (расширенный или стандартный) и указать длину передаваемых данных в поле DLC. В регистрах данных CAN_BUF[x].DATAL и CAN_BUF[x].DATAH задать необходимые для отправки данные. Далее можно переходить к выполнению остальной части программы с отправкой CAN сообщений. Отправка сообщения буфером будет произведена по RTR запросу, удовлетворяющему механизму фильтрации для принимаемых сообщений, который выбран для данного буфера.

21.7 Прием сообщений

Для приема сообщений необходимо иметь свободные и разрешенные для работы буфера, сконфигурированные на прием сообщений. При этом если по шине CAN будут передаваться сообщения от других узлов, они будут сохраняться в этих буферах.

21.8 Автоматическая фильтрация принимаемых сообщений

Для уменьшения затрат процессорного ядра на обработку принимаемых сообщений, контроллер CAN интерфейса может автоматически фильтровать принимаемые сообщения. Для каждого буфера могут быть заданы маска (CAN_BUF_FILTER[x].MASK) и фильтр (CAN_BUF_FILTER[x].FILTER) таким образом, что в этот буфер будут приниматься только те сообщения, для которых выполняется условие:

```
ID & CAN_BUF_FILTER[x].MASK == CAN_BUF_FILTER[x].FILTER
```

Если принимаемое сообщение не может быть помещено ни в один из буферов, оно будет проигнорировано. Если сообщение может быть принято более чем одним буфером, оно будет помещено в буфер с меньшим порядковым номером. При инициализации после включения питания или сброса CAN_BUF_FILTER[x].MASK и CAN_BUF_FILTER[x].FILTER для всех буферов имеют произвольное значение, таким образом, необходимо перед началом работы их проинициализировать. Для приема всех сообщений без фильтрации необходимо задать им нулевое значение. Специального бита для включения или выключения фильтрации нет.

21.9 Перезапись принятых сообщений

В буфере может быть включено разрешение перезаписи принятого сообщения. Если принимаемое сообщение не может быть сохранено в свободный буфер, то оно может быть сохранено в буфер с ранее полученным сообщением, если для него выставлен бит OVER_EN. При этом выставляется флаг OVER_WR. Таким образом, если у буфера разрешена перезапись принятых сообщений, после прочтения сообщения необходимо проверить флаг OVER_WR. Если он выставлен в 1, то необходимо сбросить OVER_WR (не сбрасывая флаг RX_FULL), затем еще раз прочесть сообщение, после чего снова проверить флаг OVER_WR и, если он не выставлен повторно, то сбросить флаг RX_FULL. И считанное значение считать корректным.

Прибегать к помощи механизма перезаписи принятых сообщений можно только в случае, когда допустима потеря сообщений, работа с перезаписью сообщений не гарантирует прием всех сообщений, а только позволяет принять сообщение корректно, так как момент чтения сообщения может совпасть с моментом сохранения нового сообщения. При этом первая часть считанного процессорным ядром сообщения будет от первого сообщения, вторая от второго. Если же между сбросом флага OVER_WR, чтением сообщения и при следующей проверке OVER_WR он оказался не выставлен, это означает, что в момент чтения сообщения из буфера в него не сохранялось новое сообщение.

21.10 Задание скорости передачи и момента семплирования

Все узлы шины CAN должны работать на одной скорости. Протокол CAN использует кодирование без возврата в ноль (NRZ). Также при передаче не передаются тактовые сигналы. Таким образом, приемники должны синхронизоваться с тактовым сигналом передатчика. Поскольку все узлы имеют свои индивидуальные тактовые генераторы, все приемники имеют специальный блок синхронизации DPLL.

Максимальная скорость передачи CAN 1 Мбит/сек. Время битового интервала Nominal Bit Time определяется как

```
TBIT = 1/Скорость передачи
```

Блок DPLL разбивает битовый интервал на интервалы Time Quanta (TQ). Битовый интервал состоит из 4 частей:

- Synchronization Segment (Sync_Seg);
- Propagation Time Segment (PSEG);

– Phase Buffer Segment 1 (SEG1);
– Phase Buffer Segment 2 (SEG2).

По определению Nominal Bit Time программируется длительностью от 8 до 25 TQ. В этом случае

```
Nominal Bit Time = TQ * (Sync_Seg + PSEG + SEG1 + SEG2)
```

Время TQ фиксировано и определяется периодом генератора и программируемым предделителем BRP со значением от 1 до 65536:

```
TQ (µs) = ((BRP+1))/CANx_CLK (MHz)
```

или

```
TQ (µs) = ((BRP+1)) * Tclk (µs)
```

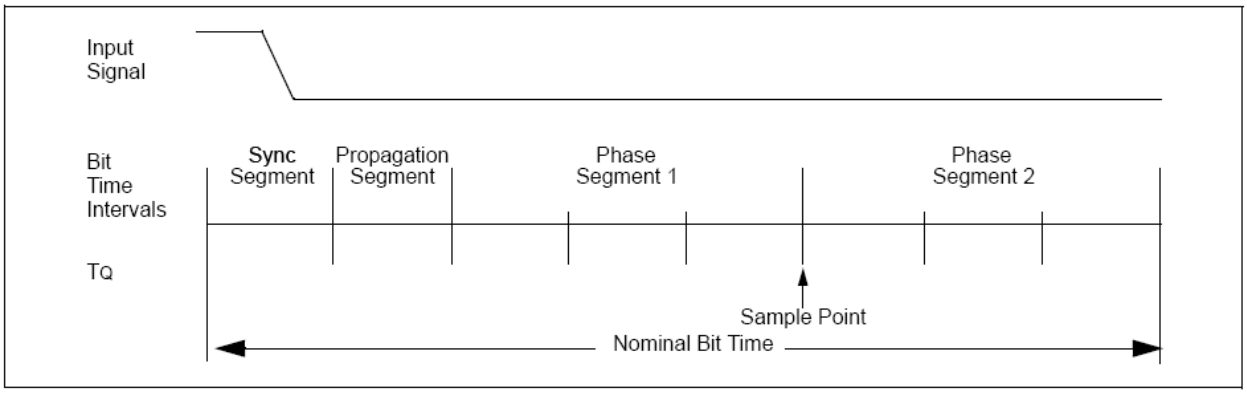


Рисунок 55. Структура битового интервала

Synchronization Segment

Эта часть битового интервала, в которой должно происходить переключение сигнала. Длительность этого интервала 1 TQ. Если переключение происходит в этой области, то приемник синхронизирован с передатчиком.

Propagation Time Segment

Эта часть предназначена, чтобы компенсировать физические задержки времени распространения сигнала в шине и внутренние задержки в узлах. Длительность этого интервала может быть запрограммирована от 1 до 8 TQ

Phase Buffer Segments

Эти интервалы предназначены для более точной установки точки семплирования, которая располагается между ними. Длительности этих интервалов могут быть запрограммированы между 1 и 8 TQ.

21.11 Синхронизация

При обнаружении фронта принимаемого сигнала этот момент принимается как граница между битовыми интервалами; в зависимости от того, на какой интервал приходится фронт, DPLL выполняет различного рода действия по подсинхронизации данных.

Hard Synchronization

Жесткая синхронизация выполняется однократно во время начала приема сообщения. Независимо от того, в каком состоянии находился DPLL при возникновении фронта, он переводится в Sync_Seg.

Resynchronization

Если фронт принимаемого сигнала отклоняется от Sync_Seg, длительность Phase Segment 1 может быть увеличена, а Phase Segment 2 уменьшена, чтобы в следующий раз фронт прошел в нужном месте. Величина изменения Phase Segment 1 и Phase Segment 2 варьируется в зависимости от значения отклонения фронта, но не превышает значения Synchronization Jump Width (SJW).

21.12 Обработка ошибок

В спецификации протокола CAN определено пять методов ограничения распространения ошибок, реализованных на аппаратном уровне. При обнаружении любой ошибки передающее устройство повторяет посылку пакета, поэтому ядру не нужно вмешиваться до тех пор, пока не возникнет грубая ошибка. Предусмотрено три метода обнаружения ошибок на уровне пакетов (контроль формата, CRC и подтверждение) и два метода на уровне бит (контроль бит и битстаффинг). Для реализации этих методов используется несколько полей, добавляемых к основному сообщению. При приеме осуществляется проверка, все ли поля присутствуют в сообщении. Если нет, то сообщение игнорируется, генерируется кадр ошибки и в регистре статуса контроллера STATUS устанавливается флаг ошибки формата пакета FRAME_ERR.

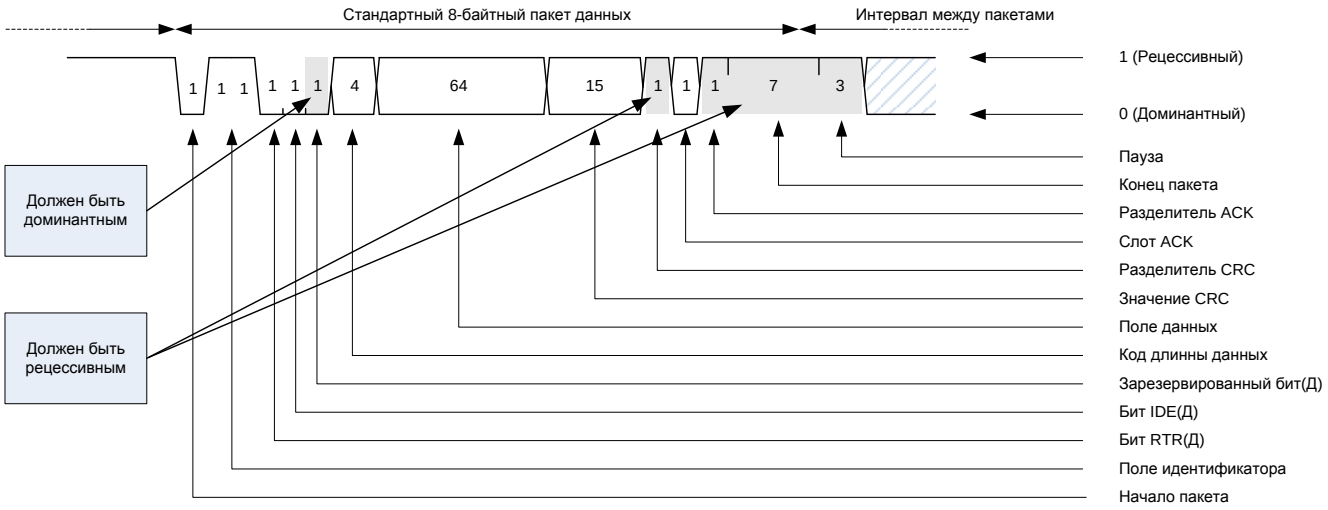


Рисунок 56. Контроль формата пакета

Каждое сообщение должно подтверждаться вставкой доминантного бита в поле подтверждения. Если подтверждения нет, передающий узел будет передавать сообщение до тех пор, пока не получит подтверждение, при этом в регистре статуса контроллера STATUS будет установлен флаг ошибки подтверждения ACK_ERR.

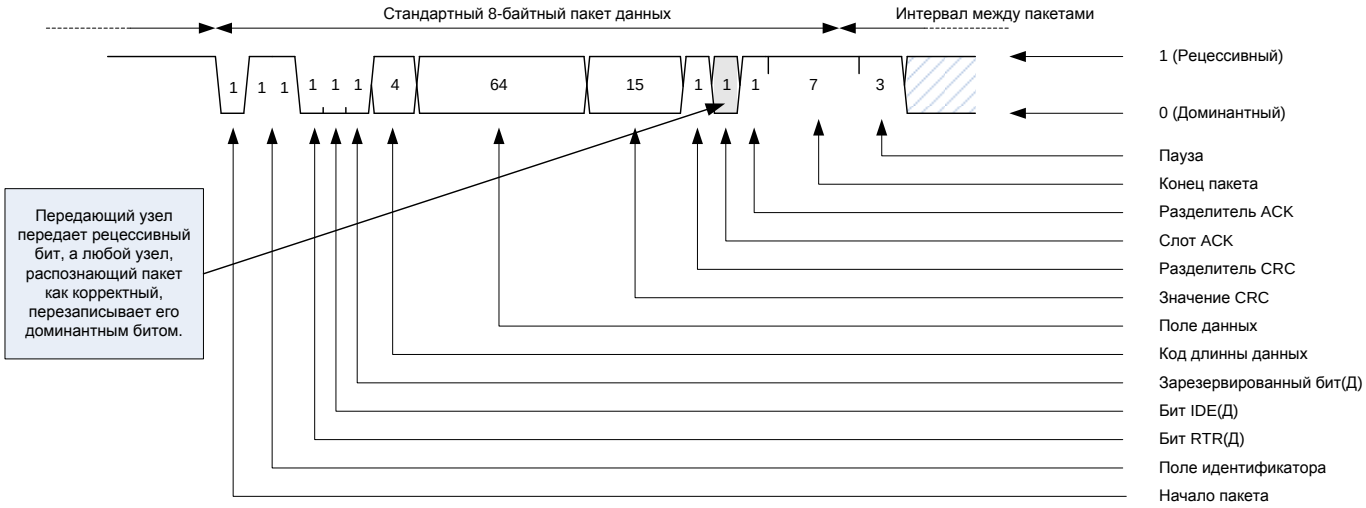


Рисунок 57. Контроль подтверждения

Пакет сообщения CAN содержит 15-битное значение CRC, которое автоматически генерируется передатчиком и проверяется приемником. С помощью этого кода можно обнаружить и исправить ошибку в 4-х битах сообщения от начала кадра до начала поля CRC. Если CRC неверен и сообщение игнорируется, то передается кадр ошибки и в регистре статуса контроллера STATUS будет установлен флаг ошибки контрольной суммы пакета CRC_ERR.

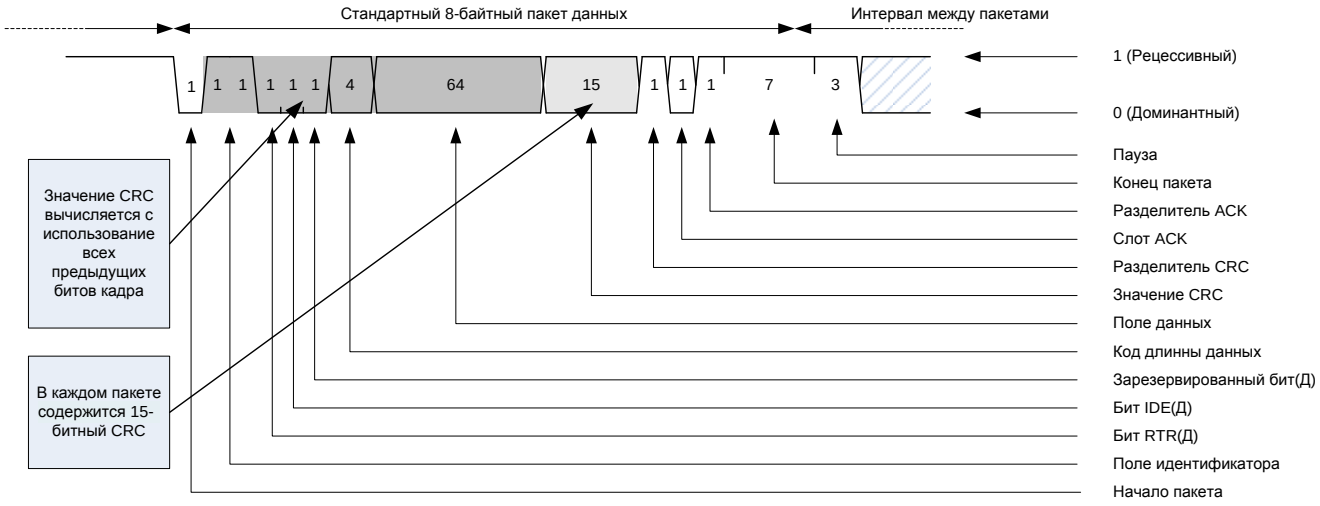


Рисунок 58. Контроль CRC

После того, как узел выиграет арбитраж, он начинает передачу своего сообщения по шине. Как и во время арбитража, CAN-контроллер считывает обратно каждый бит, выдаваемый им на шину. Поскольку узел уже выиграл арбитраж, больше никто не должен передавать данные на шине, поэтому значение каждого выданного на шину бита должно соответствовать значению, считанному обратно с шины. Если считано неверное значение, передатчик генерирует кадр ошибки, в регистре статуса контроллера STATUS устанавливает флаг ошибки передаваемых бит пакета BIT_ERR и сообщение снова ставит в очередь. Это сообщение будет послано в следующем слоте сообщений, однако при этом оно должно пройти через процесс арбитража с другими запланированными сообщениями.

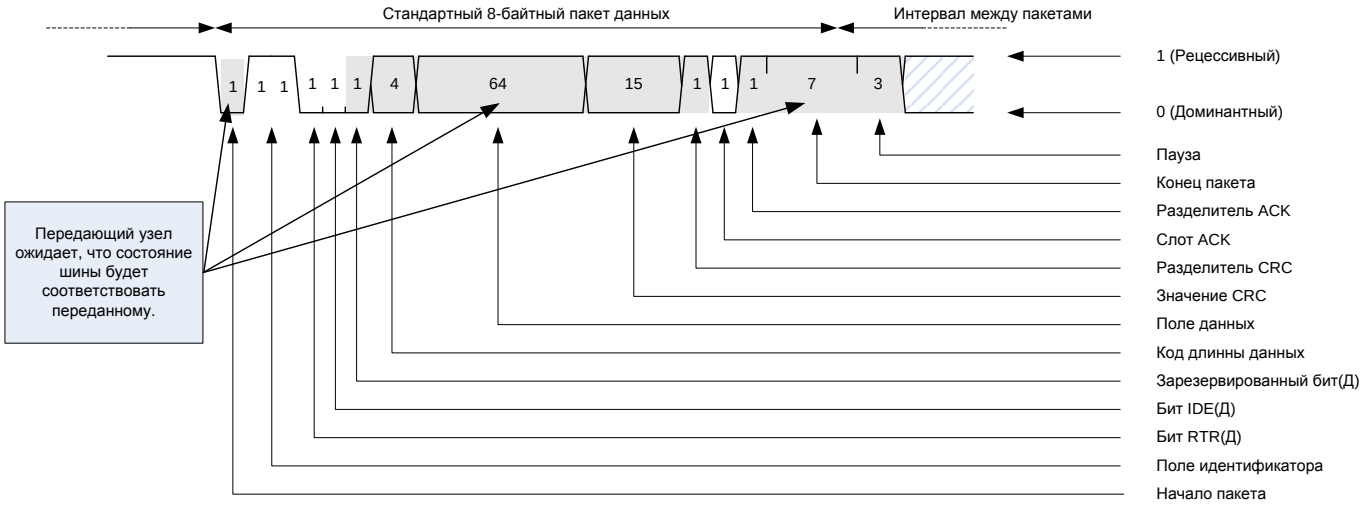


Рисунок 59. Контроль передаваемых бит

На уровне бит в протоколе CAN реализован также метод вставки бита (битстаффинг). После каждой последовательности из пяти доминантных бит вставляется рецессивный бит; если рецессивный бит не обнаружен, в регистре статуса устанавливается флаг ошибки вставленных бит пакета BIT_STUF_ERR. Этот метод позволяет предотвратить появление на шине постоянных уровней и обеспечивает наличие в потоке бит достаточного количества переходов, используемых для повторной синхронизации. Кадр ошибки в протоколе CAN представляет собой простую последовательность из шести доминантных бит. Это позволяет любому контроллеру CAN формировать на шине сообщение об ошибке сразу после ее обнаружения, не дожидаясь конца сообщения.

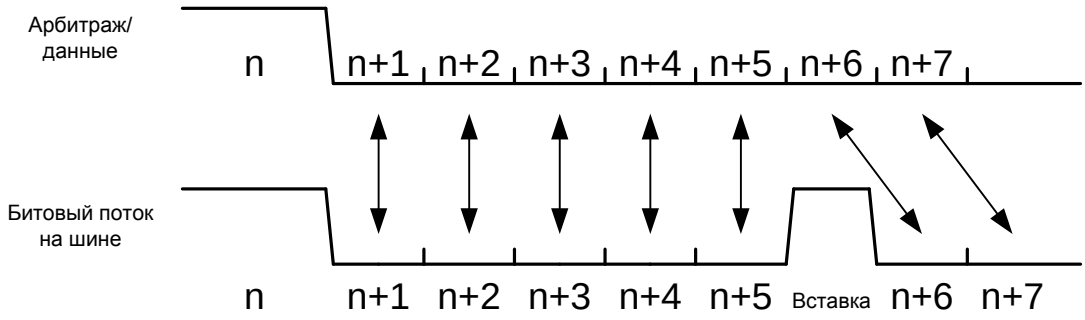


Рисунок 60. Битстаффинг

В каждом CAN контроллере имеется два счетчика. Этими счетчиками являются счетчик ошибок приема (регистр STATUS, поле RX_ERR_CNT) и счетчик ошибок передачи (регистр STATUS, поле TX_ERR_CNT). Изменение состояния этих счетчиков происходит при приеме или передаче кадра ошибки. Когда любой счетчик достигает значения 128, контроллер CAN переходит в режим «error passive». В этом режиме он продолжает отзываться на кадры ошибки, однако при генерации кадра ошибки он вместо доминантных бит выставляет на шину рецессивные. Если счетчик ошибок передачи достигает значения 255, то контроллер CAN переходит в режим «bus-off» и больше не принимает участия в обмене по шине. Для возобновления обмена (переход CAN-контроллера в состояние Error Active) необходимо либо вмешательство ядра микроконтроллера, которое повторно инициализирует CAN-контроллер, либо на шине CAN не должно быть активности (шина в рецессивном состоянии) в течение времени, равного времени передачи 128 x 11 = 1408 бит. Текущий статус состояния контроллера можно посмотреть в регистре статуса контроллера STATUS.

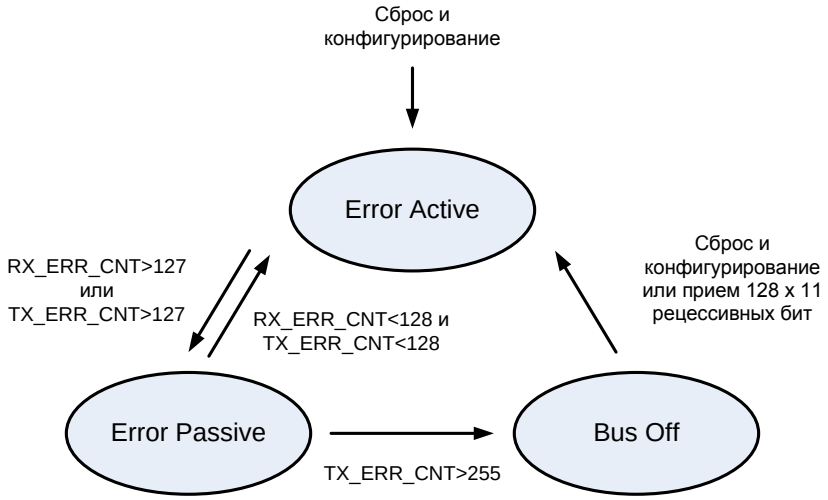


Рисунок 61. Счетчики ошибок

Контроллер CAN имеет несколько механизмов обнаружения ошибок. Во-первых, из регистра состояния контроллера CAN_STATUS можно считать текущее состояние счетчиков ошибок приема и передачи. Также в этом регистре содержится флаг превышения счетчиками ошибок порогового значения ERROR_OVER. Это значение произвольно и записывается в регистр CAN_OVER. Как и регистры синхронизации, регистр CAN_OVER можно изменять только при нахождении контроллера в состоянии сброса.

21.13 Прерывания

В контроллере CAN в качестве источников прерывания выступают буфера сообщений. Генерируемые прерывания делятся на три группы:

– Прерывания передачи (по одному для каждого буфера)
– Прерывания приема (по одному для каждого буфера)
– Прерывания ошибки

При возникновении какого-либо прерывания и наличии сигналов разрешения этих прерываний, буфер вырабатывает прерывание. Контроллер CAN объединяет прерывания приема, передачи и ошибки в каждом буфере и вырабатывает прерывание, отображаемое в регистре прерываний периферии. Если прерывание разрешено в регистре, процессор выполняет переход на обработчик прерываний. Обработчик прерываний должен выполнить действия по обработке прерывания и снять его выставление. Прерывание передачи/приема для каждого буфера может быть замаскировано путем установления соответствующего бита в регистрах CAN_INT_TX/CAN_INT_RX. Также есть возможность группового маскирования прерываний по приему, по передаче и по ошибке (см. регистр CAN_INT_EN).

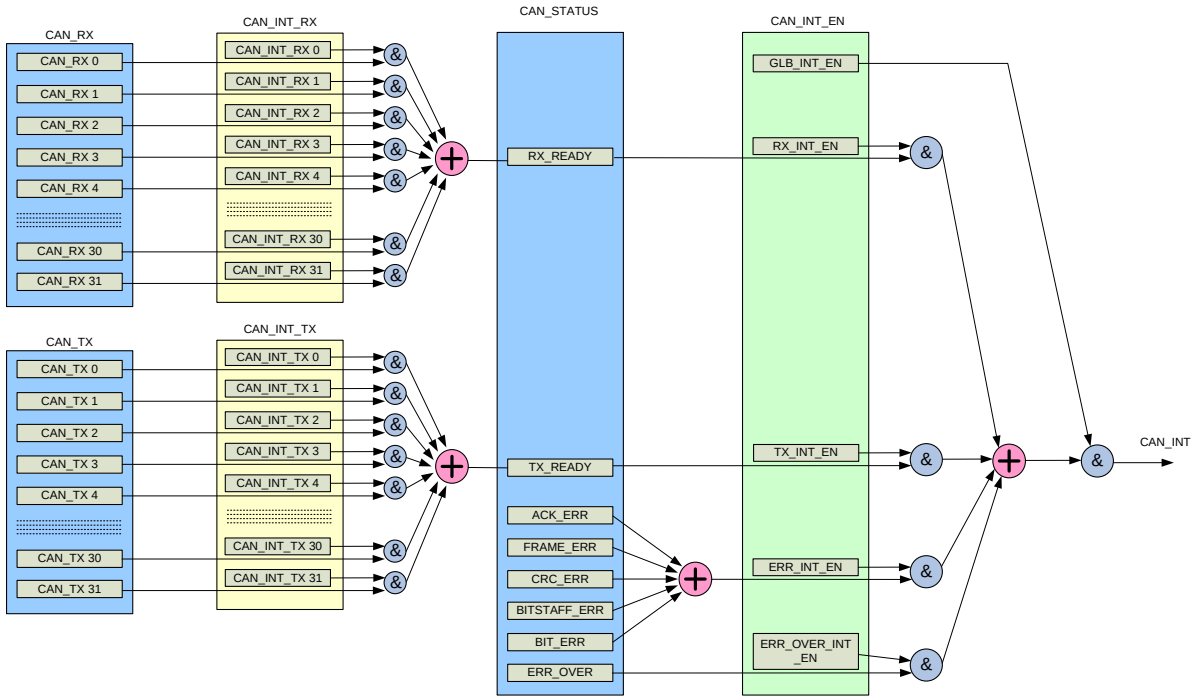


Рисунок 62. Схема формирования прерывания блока CAN

21.14 Описание регистров контроллера CAN

Таблица 235 – Описание регистров контроллера CAN

Базовый адрес	Название	Описание
0x4000_0000	MDR_CAN1	Контроллер интерфейса CAN1
0x4000_8000	MDR_CAN2	Контроллер интерфейса CAN2
Смещение		
0x00	CONTROL	**MDR_CANx->CONTROL** Регистр управление контроллером CAN
0x04	STATUS	**MDR_CANx->STATUS** Регистр состояния контроллера CAN
0x08	BITTMNG	**MDR_CANx->BITTMNG** Регистр задания скорости работы
0x10	INT_EN	**MDR_CANx->INT_EN** Регистр разрешения прерываний контроллера
0x1C	OVER	**MDR_CANx->OVER** Регистр границы счетчика ошибок
0x20	RXID	**MDR_CANx->**RXID Регистр принятого ID сообщения
0x24	RXDLC	**MDR_CANx->**RXDLC Регистр принятого DLC сообщения

Базовый адрес	Название	Описание
0x28	RXDATAL	**MDR_CANx->**RXDATAL Регистр принятых данных
0x2C	RXDATAH	**MDR_CANx->RXDATAH** Регистр принятых данных
0x30	TXID	**MDR_CANx->**TXID Регистр передаваемого ID сообщения
0x34	TXDLC	**MDR_CANx->**TXDLC Регистр передаваемого DLC сообщения
0x38	DATAL	**MDR_CANx->**TXDATAL Регистр передаваемых данных
0x3C	DATAH	**MDR_CANx->**TXDATAH Регистр передаваемых данных
0x40	BUF_CON[0]	**MDR_CANx->BUF_CON** Регистр управления буфером 01
	…	
0xBC	BUF_CON[31]	**MDR_CANx->BUF_CON** Регистр управления буфером 32
0xC0	INT_RX	Флаги разрешения прерываний от приемных буферов MDR_CANx->INT_RX
0xC4	RX	Флаги RX_FULL от приемных буферов MDR_CANx->RX
0xC8	INT_TX	Флаги разрешения прерываний от передающих буферов MDR_CANx->INT_TX
0xCC	TX	Флаги ~TX_REQ от передающих буферов MDR_CANx->TX
0x200	CAN_BUF[0].ID	**MDR_CANx->CAN_BUF[x].**ID ID сообщения буфера 01
0x204	CAN_BUF[0].DLC	**MDR_CANx->CAN_BUF[x].**DLC DLC сообщения буфера 01
0x208	CAN_BUF[0].DATAL	**MDR_CANx->CAN_BUF[x].**DATAL Данные сообщения буфера 01
0x20C	CAN_BUF[0].DATAH	**MDR_CANx->CAN_BUF[x].**DATAH Данные сообщения буфера 01

Базовый адрес	Название	Описание
0x210	CAN_BUF[1].ID	**MDR_CANx->CAN_BUF[x].**ID ID сообщения буфера 02
	…	
0x3FC	CAN_BUF[31].DATAH	**MDR_CANx->CAN_BUF[x].**DATAH Данные сообщения буфера 32
0x500	CAN_BUF_FILTER[0].MASK	**MDR_CANx->CAN_BUF_FILTER[x].**MASK Маска для приема сообщения в буфер 01
0x504	CAN_BUF_FILTER[0].FILTER	**MDR_CANx->CAN_BUF_FILTER[x].**FILTER Фильтр для приема сообщения в буфер 01
0x508	CAN_BUF_FILTER[1].MASK	**MDR_CANx->CAN_BUF_FILTER[x].**MASK Маска для приема сообщения в буфер 02
	…	
0x5FC	CAN_BUF_FILTER[31].FILTER	**MDR_CANx->CAN_BUF_FILTER[x].**FILTER Фильтр для приема сообщения в буфер 32

21.14.1 MDR_CANx->CONTROL

Таблица 236 – Регистр управления контроллером CONTROL

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
	-	**ROP**	**SAP**	**STM**	**ROM**	**CAN EN**

Таблица 237 – Описание бит регистра CONTROL

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31...5	-	Зарезервировано
4	ROP	Прием собственных пакетов (Receive own packets): 1 – контроллер принимает собственные пакеты; 0 – контроллер принимает только чужие пакеты
3	SAP	Подтверждение собственных пакетов (Send ACK on own packets): 1 – контроллер подтверждает прием собственных пакетов; 0 – контроллер подтверждает прием только чужих пакетов
2	STM	Режим самотестирования (Self Test Mode): 1 – контроллер работает в режиме самотестирования; 0 – контроллер работает в нормальном режиме
1	ROM	Режим «Только прием» (Read Only Mode): 1 – контроллер работает только на прием; 0 – контроллер работает в нормальном режиме
0	CAN_EN	Режим работы контроллера CAN: 1 – разрешение работы; 0 – работа запрещена

21.14.2 MDR_CANx->STATUS

Таблица 238 – Регистр состояния контроллера STATUS

Номер	7	6	5	4	3	2	1	0
Доступ	R/W	R/W	R/W	R/W	R/W	R/W	RO	RO
Сброс	0	0	0	0	0	0	0	0
	ACK ERR	**FRAME ERR**	**CRC ERR**	**BIT STUFF ERR**	**BIT ERR**	**ERROR OVER**	**TX READY**	**RX READY**

Номер	31…24	23…16	15…13	12	11	10…9	8
Доступ	RO	RO	U	RO	RO	RO	R/W
Сброс	0	0	0	0	0	0	0
	TX ERR CNT [7:0]	**RX ERR CNT [7:0]**	**-**	**TX ERR CNT8**	**RX ERR CNT8**	**ERR STATUS[1:0]**	**ID LOWER**

Таблица 239 – Описание бит регистра STATUS

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31...24	TX ERR CNT [7:0]	Счетчик ошибок передатчика TEC, биты [7:0]: TEC > 127, ERROR PASSIVE
23…16	RX ERR CNT [7:0]	Счетчик ошибок приемника REC, биты [7:0]: REC > 127, ERROR PASSIVE
15…13	-	
12	TX ERR CNT8	Счетчик ошибок передатчика TEC, бит 8: 0 – TEC менее 255; 1 – TEC более 255
11	RX ERR CNT8	Счетчик ошибок приемника REC, бит 8: 0 – REC менее 255; 1 – REC более 255
10…9	ERR STATUS[1:0]	Статус состояния контроллера CAN: 00 – ERROR ACTIVE, при возникновении ошибки отсылается флаг активной ошибки; 01 – ERROR PASSIVE, при возникновении ошибки отсылается флаг пассивной ошибки; 1x – BUS OFF, ожидается восстановление шины
8	ID LOWER	Флаг «проигрыша» арбитража: 0 – при передаче не было проигрыша арбитража; 1 – при передаче был проигран арбитраж
7	ACK ERR	Флаг ошибки подтверждения приема: 0 – нет ошибки; 1 – есть ошибка
6	FRAME ERR	Флаг ошибки формата пакета: 0 – нет ошибки; 1 – есть ошибка

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
5	CRC ERR	Флаг ошибки контрольной суммы пакета: 0 – нет ошибки; 1 – есть ошибка
4	BIT STUFF ERR	Флаг ошибки вставленных бит пакета: 0 – нет ошибки; 1 – есть ошибка
3	BIT ERR	Флаг ошибки передаваемых бит пакета: 0 – нет ошибки; 1 – есть ошибка
2	ERROR OVER	Флаг превышения TEC и REC уровня, заданного ERROR_MAX: 0 – ERROR_MAX < TEC и REC; 1 – ERROR_MAX ≥ TEC или REC
1	TX READY	Флаг наличия буферов для отправки: 0 – нет буферов готовых для отправки сообщений; 1 – есть буфер готовый для отправки сообщений
0	RX READY	Флаг наличия принятых сообщений: 0 – нет буферов с принятыми сообщениями; 1 – есть буфер с принятым сообщением

21.14.3 MDR_CANx->BITTMNG

Таблица 240 – Регистр задания скорости работы BITTMNG

Номер	31…28	27	26…25	24…22	21…19	18…16	15…0
Доступ	U	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0	0
	-	**SB**	**SJW [1:0]**	**SEG2 [2:0]**	**SEG1 [2:0]**	**PSEG [2:0]**	**BRP [15:0]**

Таблица 241 – Описание бит регистра BITTMNG

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31...28	-	Зарезервировано
27	SB	Семплирование: 0 – однократное; 1 – трехкратное с мажоритарным контролем
26…25	SJW [1:0]	Значение размера фазы SJW: 11 = Synchronization jump width time = 4 x TQ 10 = Synchronization jump width time = 3 x TQ 01 = Synchronization jump width time = 2 x TQ 00 = Synchronization jump width time = 1 x TQ SJW – это максимальное значение, на которое происходит подстройка приема и передачи при работе на шине CAN. Приемник подстраивается на значение ошибки, но не более чем SJW
24…22	SEG2 [2:0]	Значение размера фазы SEG2: 111 = Phase Segment 2 time = 8 x TQ 110 = Phase Segment 2 time = 7 x TQ 101 = Phase Segment 2 time = 6 x TQ 100 = Phase Segment 2 time = 5 x TQ 011 = Phase Segment 2 time = 4 x TQ 010 = Phase Segment 2 time = 3 x TQ 001 = Phase Segment 2 time = 2 x TQ 000 = Phase Segment 2 time = 1 x TQ SEG2 – это время, используемое для сокращения битового интервала при подстройке
21…19	SEG1 [2:0]	Значение размера фазы SEG1: 111 = Phase Segment 1 time = 8 x TQ 110 = Phase Segment 1 time = 7 x TQ 101 = Phase Segment 1 time = 6 x TQ 100 = Phase Segment 1 time = 5 x TQ 011 = Phase Segment 1 time = 4 x TQ 010 = Phase Segment 1 time = 3 x TQ 001 = Phase Segment 1 time = 2 x TQ 000 = Phase Segment 1 time = 1 x TQ SEG1 – это время, используемое для увеличения битового интервала при подстройке

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
18…16	PSEG[2:0]	Значение размера фазы PSEG 111 = Propagation time = 8 x TQ 110 = Propagation time = 7 x TQ 101 = Propagation time = 6 x TQ 100 = Propagation time = 5 x TQ 011 = Propagation time = 4 x TQ 010 = Propagation time = 3 x TQ 001 = Propagation time = 2 x TQ 000 = Propagation time = 1 x TQ PSEG - это время компенсирующее задержку распространения сигналов в шине CAN
15…0	BRP [15:0]	Предделитель системной частоты: CLK = CANx_CLK/(BRP + 1) TQ(us) = 1/ CLK(MHz) = (BRP + 1)/CANx_CLK(MHz)

21.14.4 MDR_CANx->INT_EN

Таблица 242 – Регистр разрешения прерываний INT_EN

Номер	31…5	4	3	2	1	0
Доступ	U	U	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0
	-	**ERR OVER INT EN**	**ERR INT EN**	**TX INT EN**	**RX INT EN**	**GLB INT EN**

Таблица 243 – Описание бит регистра INT_EN

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31...5	-	Зарезервировано
4	ERR OVER INT EN	Флаг разрешения прерывания по превышению TEC или REC допустимого значения в ERROR_MAX: 0 – запрещено прерывание; 1 – разрешено прерывание
3	ERR INT EN	Флаг разрешения прерывания по возникновению ошибки: 0 – запрещено прерывание; 1 – разрешено прерывание
2	TX INT EN	Флаг разрешения прерывания по возможности передачи: 0 – запрещено прерывание; 1 – разрешено прерывание
1	RX INT EN	Флаг разрешения прерывания по приему сообщений: 0 – запрещено прерывание; 1 – разрешено прерывание
0	GLB INT EN	Общий флаг разрешения прерывания блока CAN: 0 – запрещено прерывание; 1 – разрешено прерывание

21.14.5 MDR_CANx->OVER

Таблица 244 – Регистр границы счета ошибок OVER

Номер	31…8	7…0
Доступ	U	R/W
Сброс	0	0
	-	**ERROR_MAX[7:0]**

Таблица 245 – Описание бит регистра OVER

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31...8	-	Зарезервировано
7…0	ERROR MAX [7:0]	Регистр границы счетчика ошибок Допустимое значение счетчиков ошибок TEC и REC, при превышении которого вырабатывается флаг ERROR_OVER

21.14.6 MDR_CANx->BUF_CON[x]

Таблица 246 – Регистр управления буфером BUF_CON[x]

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
	-	**OVER WR**	**RX FULL**	**TX REQ**	**PRIO R 0**	**RTR EN**	**OVER EN**	**RX TXn**	**EN**

Таблица 247 – Описание бит регистра BUF_CON[x]

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31...8	-	Зарезервировано
7	OVER_WR	Флаг перезаписи принятого сообщения: 0 – не было перезаписи; 1 – была перезапись принятого сообщения
6	RX_FULL	Флаг готовности приема: 0 – нет принятого сообщения; 1 – принятое сообщение в буфере
5	TX_REQ	Запрос на отправку сообщения: 0 – нет запроса или отправлено; 1 – запрос на отправку
4	PRIOR_0	Приоритет при отправке: 0 – приоритет; 1 – нет приоритета
3	RTR_EN	Режим ответа на RTR: 0 – не отвечать при приеме RTR; 1 – ответить при приеме RTR в буфер
2	OVER_EN	Разрешение перезаписи принятого сообщения: 0 – не разрешена перезапись; 1 – разрешена перезапись сообщения

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
1	RX_TXn	Режим работы буфера: 0 – на передачу; 1 – на прием
0	EN	Разрешение работы буфера: 0 – отключен; 1 – работает

21.14.7 MDR_CANx->INT_RX

Таблица 248 – Регистр разрешения прерываний от приемных буферов INT_RX

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

Таблица 249 – Описание бит регистра INT_RX

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...0	CAN INT RX[31:0]	Флаги разрешения прерываний от буферов по приему сообщений: CAN_INT_RX[0] – для первого буфера CAN_INT_RX[1] – для второго буфера и так далее

21.14.8 MDR_CANx->RX

Таблица 250 – Регистр RX флагов RX_FULL от приемных буферов

Номер	31…0
Доступ	RO
Сброс	0
	CAN_RX[31:0]

Таблица 251 – Описание бит регистра RX

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...0	CAN RX[31:0]	Флаги RX_FULL разрешенных на прием буферов: CAN_RX[0] – флаг RX_FULL от первого буфера CAN_RX[1] – флаг RX_FULL от второго буфера и так далее, доступны только на чтение

21.14.9 MDR_CANx->INT_TX

Таблица 252 – Регистр разрешения прерываний от передающих буферов INT_TX

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

Таблица 253 – Описание бит регистра INT_TX

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31…0	CAN INT TX[31:0]	Флаги разрешения прерываний от буферов по передаче сообщений: CAN_INT_TX[0] – для первого буфера CAN_INT_TX[1] – для второго буфера и так далее

21.14.10 MDR_CANx->TX

Таблица 254 – Регистр TX флагов ~TX_REQ от передающих буферов

Номер	31…0
Доступ	RO
Сброс	0
	CAN_TX[31:0]

Таблица 255 – Описание бит TX

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31...0	CAN TX[31:0]	Флаги ~TX_REQ разрешенных на передачу буферов: CAN_TX[0] – флаг ~TX_REQ от первого буфера CAN_TX[1] – флаг ~TX_REQ от второго буфера и так далее, доступны только на чтение

21.14.11 MDR_CANx->RXID
MDR_CANx->TXID
MDR_CANx->CAN_BUF[x].ID
MDR_CANx->CAN_BUF_FILTER[x].MASK
MDR_CANx->CAN_BUF_FILTER[x].FILTER

Таблица 256 – Регистры RXID, TXID и CAN_BUF[x].ID идентификаторов

Номер	31…29	28…18	17…0
Доступ	U	R/W	R/W
Сброс	0	0	0
	-	**SID [10:0]**	**EID [17:0]**

Таблица 257 – Описание бит регистров RXID, TXID и CAN_BUF[x].ID

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...29	-	Зарезервировано
28…18	SID [10:0]	Поле SID. Для стандартного и расширенного пакетов CAN. Чем меньше значение поля, тем больший приоритет имеет пакет при арбитраже
17…0	EID [17:0]	Поле EID. Для расширенных пакетов CAN. Чем меньше значение поля, тем больший приоритет имеет пакет при арбитраже

21.14.12 MDR_CANx->RXDLC
MDR_CANx->TXDLC
MDR_CANx->CAN_BUF[x].DLC

Таблица 258 – Регистры RXDLC, TXDLC и CAN_BUF[x].DLC сообщения

Номер	31…13	12	11	10	9	8	7…4	3…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
	-	**IDE**	**SRR**	**R0**	**R1**	**RTR**	**-**	**DLC [3:0]**

Таблица 259 – Описание бит регистров RXDLC, TXDLC и CAN_BUF[x].DLC

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...13	-	Зарезервировано
12	IDE	Поле IDE. Поле, обозначающее формат пакета: 0 – стандартный пакет; 1 – расширенный пакет
11	SRR	Поле SRR, расширенного формата. Всегда должно быть равно “1”
10	R0	Поле R0. Всегда должно быть равно “0”
9	R1	Поле R1, расширенного формата. Всегда должно быть равно “1”
8	RTR	Поле RTR, запроса обратного ответа: 0 – нет запроса; 1 – есть запрос. Если узел получил пакет с запросом обратного ответа, то он должен ответить
7…4	-	Зарезервировано

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
3…0	DLC[3:0]	Поле DLC, длина передаваемых данных в пакете: 0000 – нет данных 0001 – 1 байт 0010 – 2 байт 0011 – 3 байт 0100 – 4 байт 0101 – 5 байт 0110 – 6 байт 0111 – 7 байт 1000 – 8 байт 1xxx – 8 байт и недопустимо

21.14.13 MDR_CANx->RXDATAL
MDR_CANx->TXDATAL
MDR_CANx->CAN_BUF[x].DATAL

Таблица 260 – Регистры RXDATAL, TXDATAL и CAN_BUF[x].DATAL данных сообщения

Номер	31…24	23…16	15…8	7…0
Доступ	R/W	R/W	R/W	R/W
Сброс	0	0	0	0
	DB3[7:0]	**DB2[7:0]**	**DB1[7:0]**	**DB0[7:0]**

Таблица 261 – Описание бит регистров RXDATAL, TXDATAL и CAN_BUF[x].DATAL

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31…24	DB3[7:0]	Поле DB3. Четвертый байт, передаваемый в пакете
23…16	DB2[7:0]	Поле DB2. Третий байт, передаваемый в пакете
15…8	DB1[7:0]	Поле DB1. Второй байт, передаваемый в пакете
7…0	DB0[7:0]	Поле DB0. Первый байт, передаваемый в пакете

21.14.14 MDR_CANx->RXDATAH
MDR_CANx->TXDATAH
MDR_CANx->CAN_BUF[x].DATAH

Таблица 262 – Регистры RXDATAH, TXDATAH и CAN_BUF[x].DATAH данных сообщения

Номер	31…24	23…16	15…8	7…0
Доступ	R/W	R/W	R/W	R/W
Сброс	0	0	0	0
	DB7[7:0]	**DB6[7:0]**	**DB5[7:0]**	**DB4[7:0]**

Таблица 263 – Описание бит регистров RXDATAH, TXDATAH и CAN_BUF[x].DATAH

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31…24	DB7[7:0]	Поле DB7. Восьмой байт, передаваемый в пакете
23…16	DB6[7:0]	Поле DB6. Седьмой байт, передаваемый в пакете
15…8	DB5[7:0]	Поле DB5. Шестой байт, передаваемый в пакете
7…0	DB4[7:0]	Поле DB4. Пятый байт, передаваемый в пакете

22 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 трех 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.

22.1 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.

22.1.1 Структурная схема

Структурная схема блока Таймер представлена на Рисунок 63

Таймер содержит основной 16-битный счетчик CNT, блок регистров управления и четыре канала схем захвата/ШИМ.

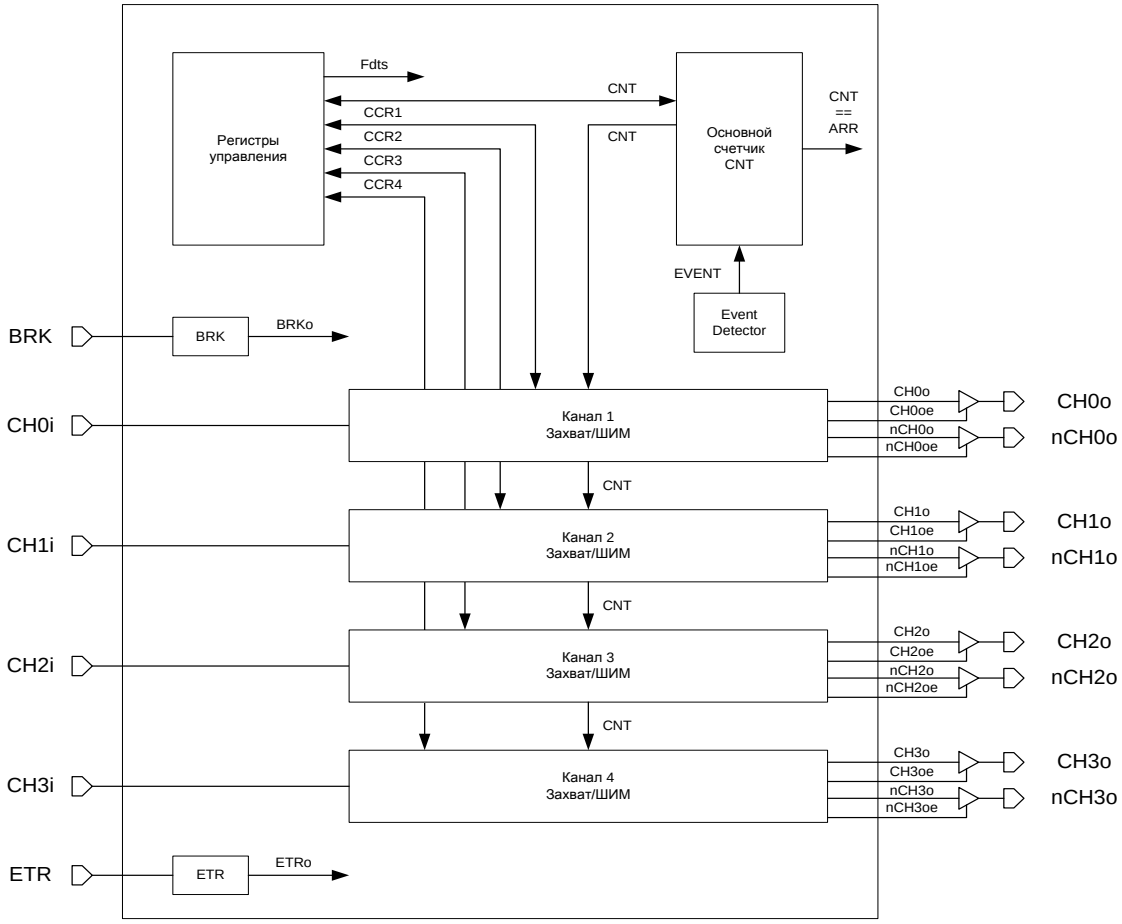


Рисунок 63. Структурная схема таймера

Таймер позволяет работать в режимах:

- таймер;
- расширенный таймер, с объединением нескольких таймеров;
- режим захвата;
- ШИМ.

22.1.2 Соответствие выводов таймера выводам микроконтроллера

BRK – выводы TMRx_BLK
ETR – выводы TMRx_ETR
CH0i – выводы TMRx_CH1
CH1i – выводы TMRx_CH2
CH2i – выводы TMRx_CH3
CH3i – выводы TMRx_CH4

CH0o – выводы TMRx_CH1
CH1o – выводы TMRx_CH2
CH2o – выводы TMRx_CH3
CH3o – выводы TMRx_CH4

nCH0o – выводы TMRx_CH1N
nCH1o – выводы TMRx_CH2N
nCH2o – выводы TMRx_CH3N
nCH3o – выводы TMRx_CH4N

22.1.3 Инициализация таймера

Перед началом работы с таймерами в первую очередь должны быть включены тактовые сигналы. Параметры задаются в блоке «Сигналы тактовой частоты».

Для задания тактовой частоты блока необходимо установить бит разрешения тактирования блока (бит 14 для таймера 1, бит 15 для таймера 2, бит 16 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.

22.1.4 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.

22.2 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.
```

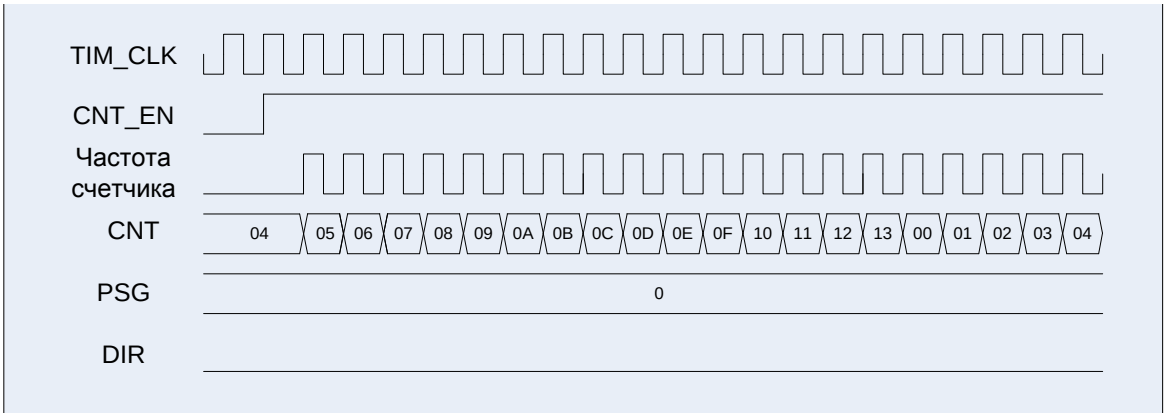


Рисунок 64. Диаграммы работы таймера, счет вверх

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.
```

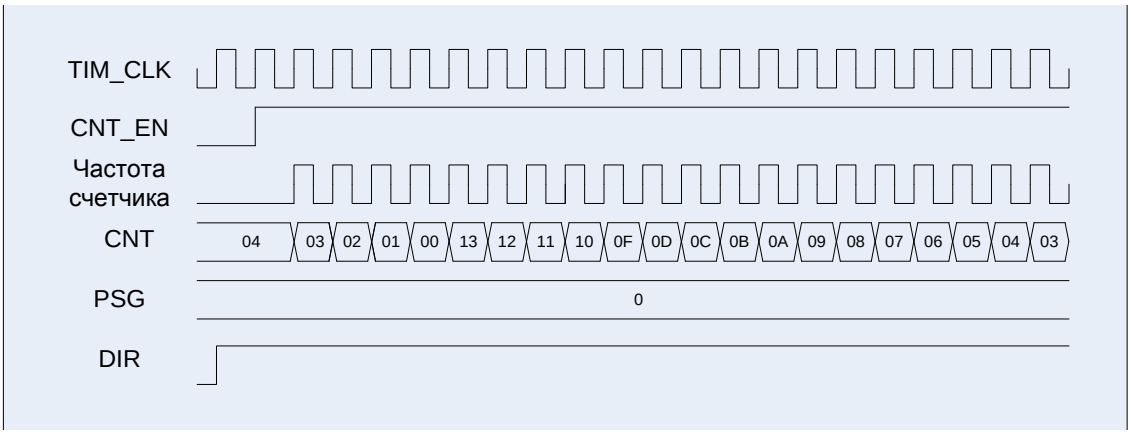


Рисунок 65. Диаграммы работы таймера, счет вниз

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.
```

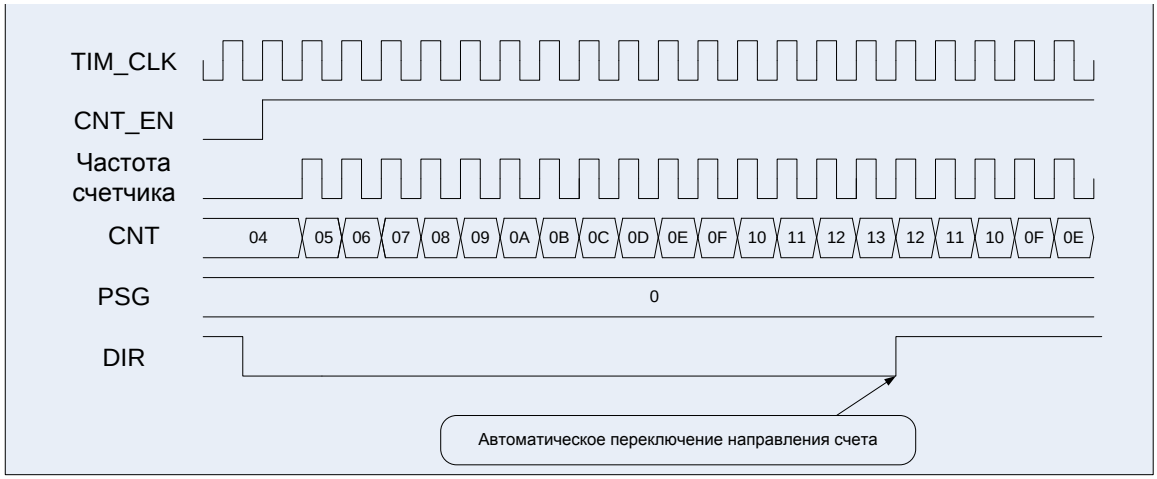


Рисунок 66. Диаграммы работы таймера, счет вверх/вниз, сначала вверх

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.
```

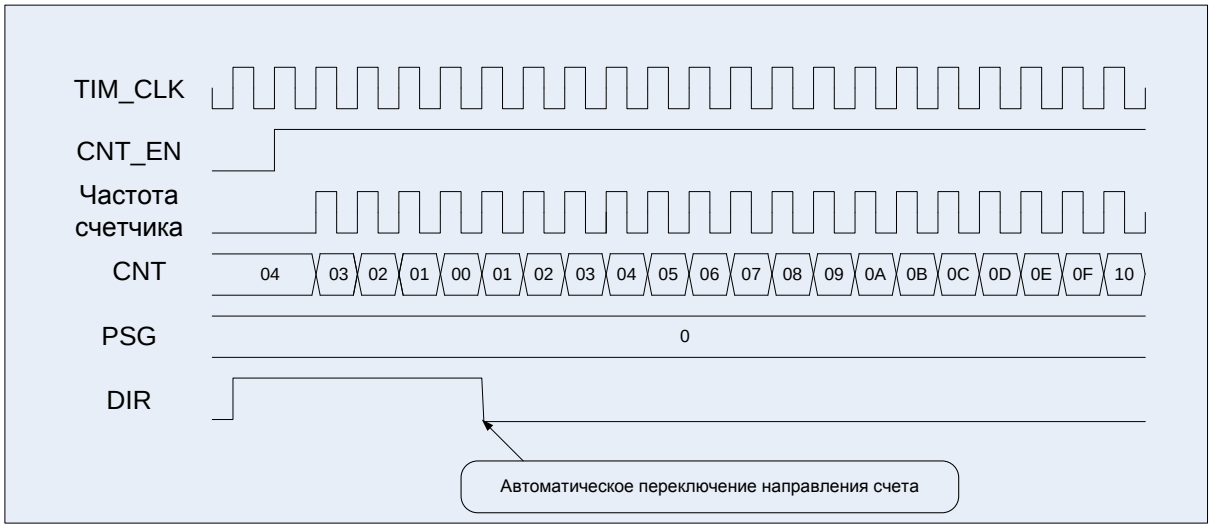


Рисунок 67. Диаграммы работы таймера, счет вверх/вниз, сначала вниз

22.3 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.

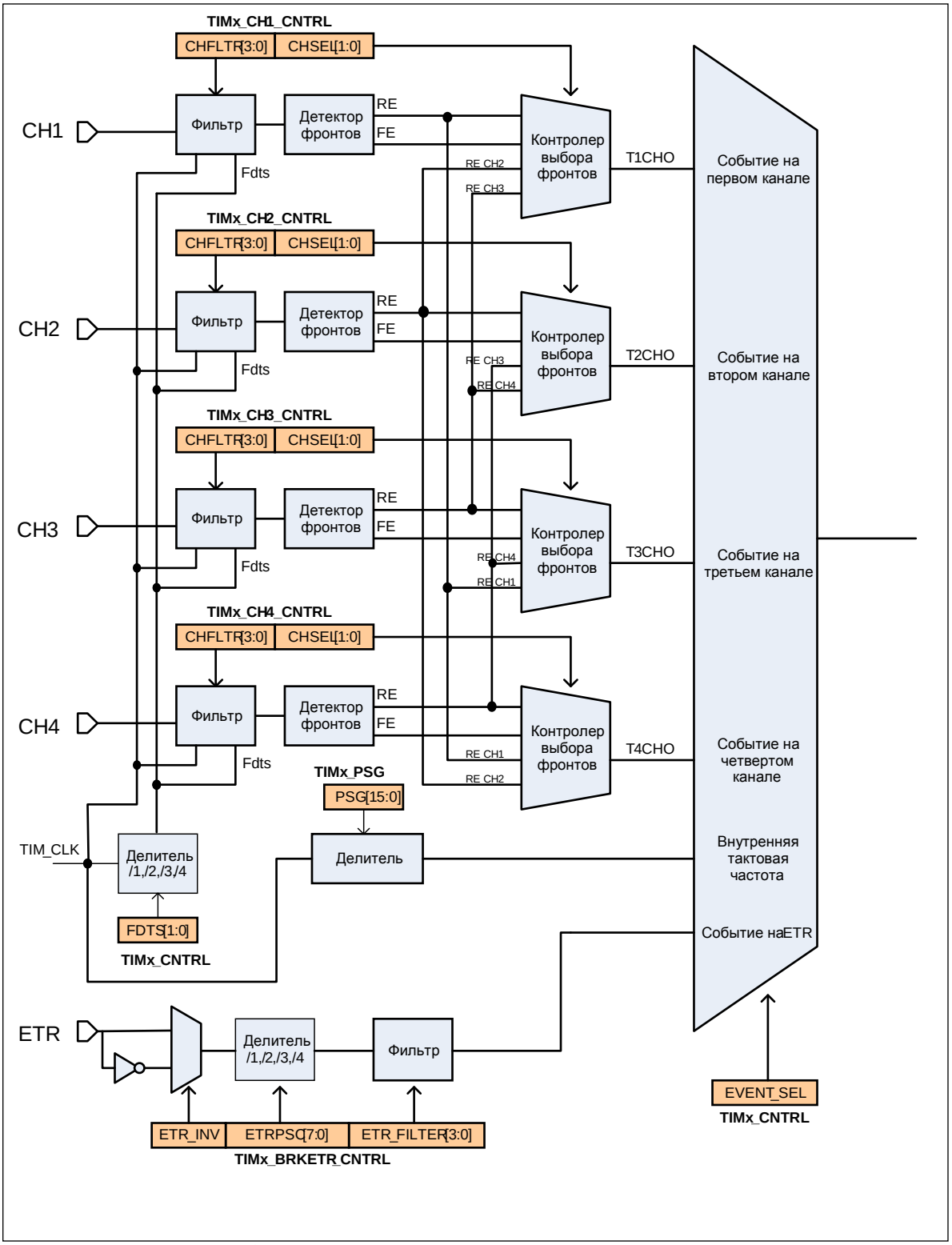


Рисунок 68. Структурная схема формирования события для счета

22.3.1 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 69. Диаграммы работы счетчика: счет вверх), DIR = 1 – счетчик считает вниз (см. Рисунок 70. Диаграммы работы счетчика: счет вниз).

CNT_MODE = 00, EVENT_SEL = 0000, DIR = 0

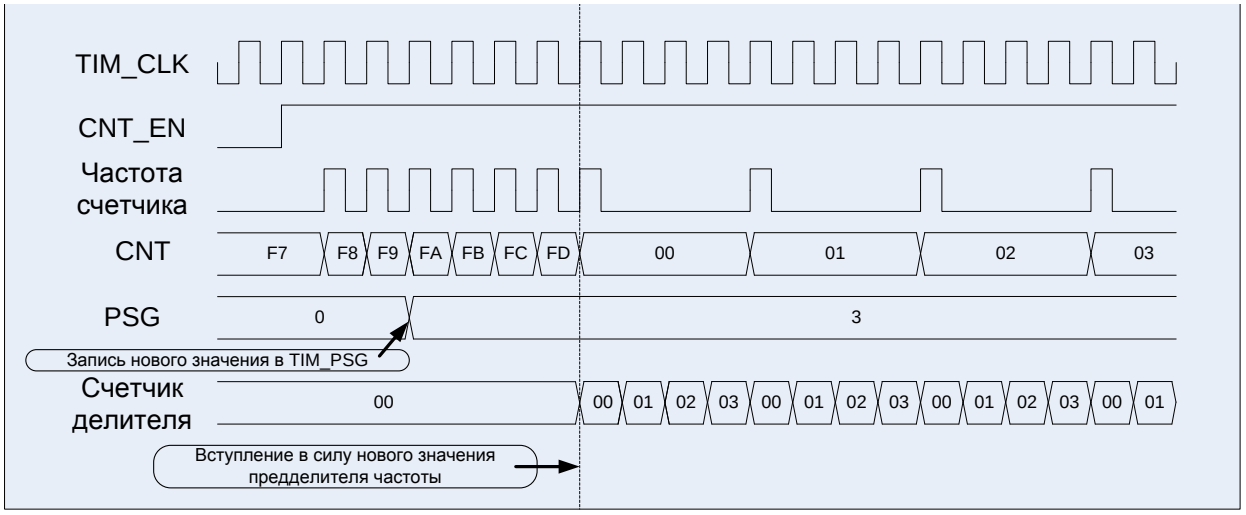


Рисунок 69. Диаграммы работы счетчика: счет вверх

CNT_MODE = 00, EVENT_SEL = 0000, DIR = 1

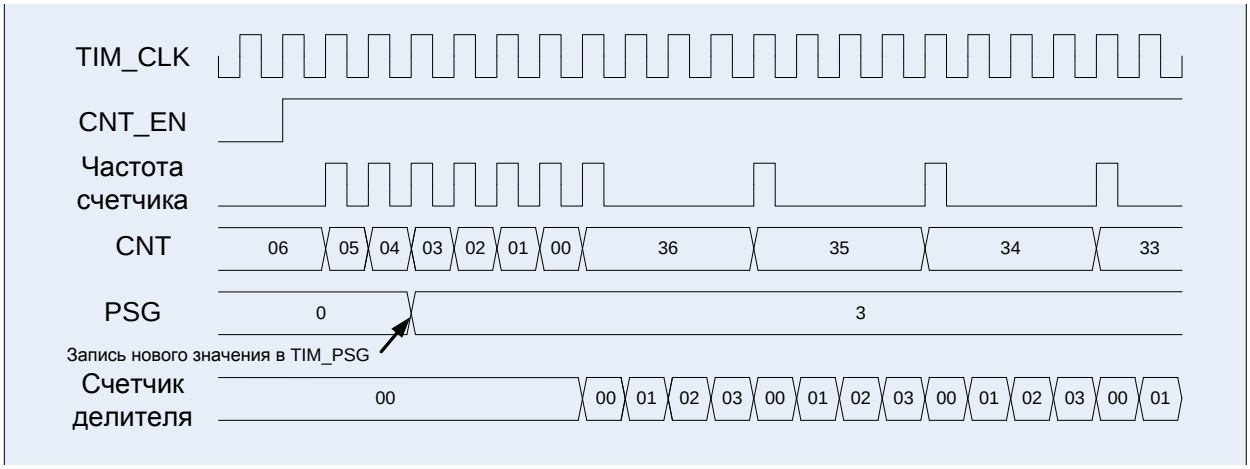


Рисунок 70. Диаграммы работы счетчика: счет вниз

Если CNT_MODE = 00, то направление счета определяется полем DIR, если CNT_MODE = 01, счетчик считает вверх/вниз с автоматическим изменением DIR (см. Рисунок 71).

CNT_MODE = 01, EVENT_SEL = 0000, DIR = 1

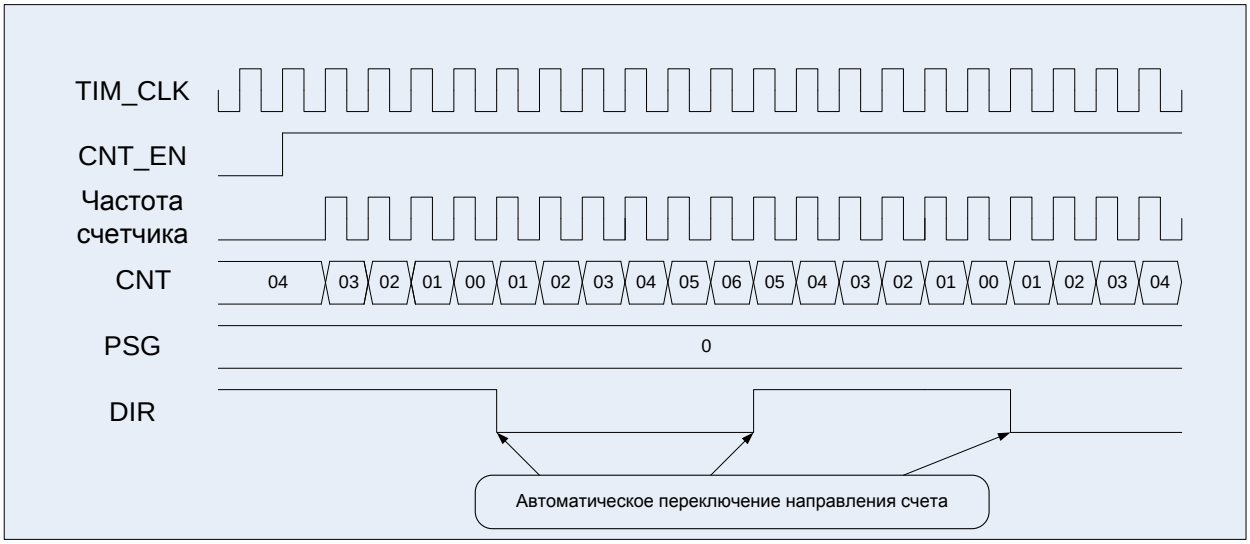


Рисунок 71. Диаграммы работы счетчика: счет вниз/вверх

22.3.2 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 двух 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.

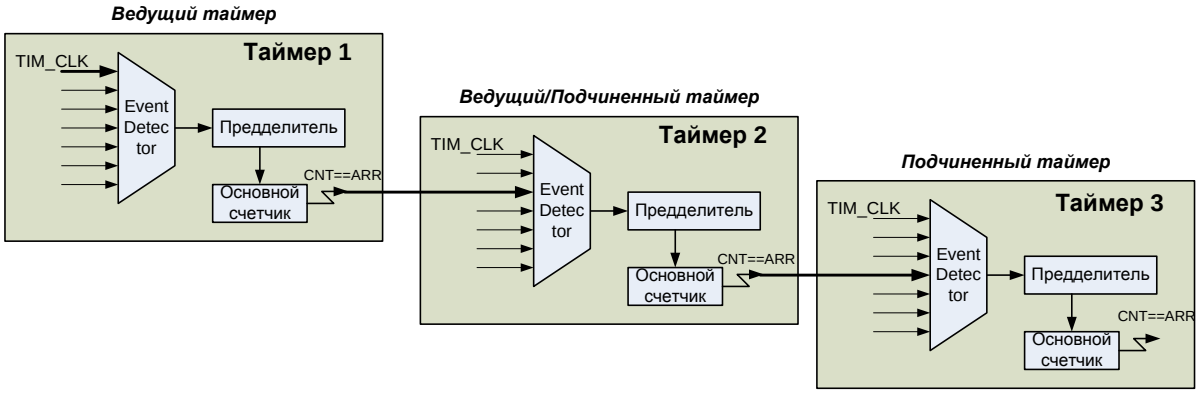


Рисунок 72. Пример каскадного соединения таймеров

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;

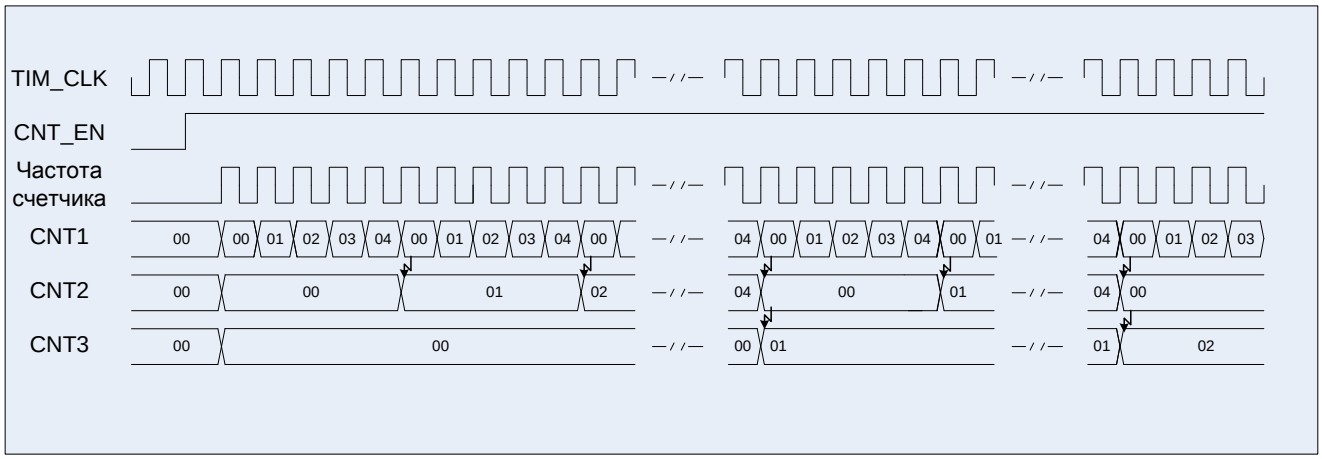


Рисунок 73. Диаграммы работы трех таймеров в каскаде

22.3.3 Внешний тактовый сигнал «Режим 1». События на линиях TxCHO данного счетчика

Этот режим выбирается, когда EVENT_SEL = 01xx в регистре CNTRL. Счетчик считает только по переднему фронту (см. Рисунок 74). Биты CHSEL[1:0] регистра MDR_TIMERx->CHy_CNTRL не имеют значения. Канал таймера работает с битами CHSEL[1:0] только в режиме захвата, генерируя прерывание по соответствующему фронту.

На входе сигнала стоит фильтр, с помощью которого можно контролировать длительность сигнала, для фильтрации можно использовать как сигнал TIM_CLK, при этом может быть идентифицированная длительность 1, 2, 4, 8 TIM_CLK, также можно при фильтровании использовать производную от TIM_CLK частоту FDTS. Частота семплирования данных задается в регистре CNTRL в поле FDTS.

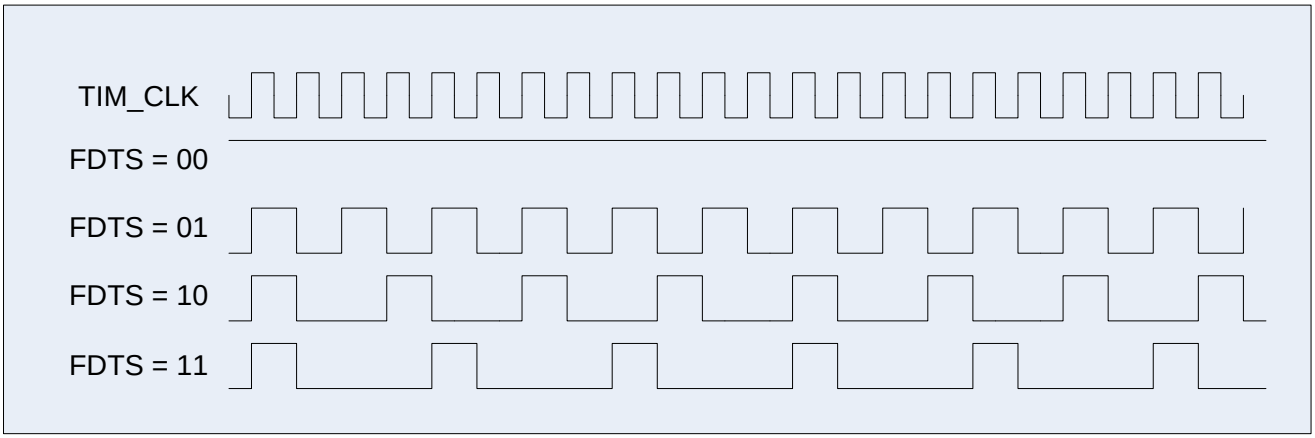


Рисунок 74. Диаграммы возможных частот семплирования данных (FDTS)

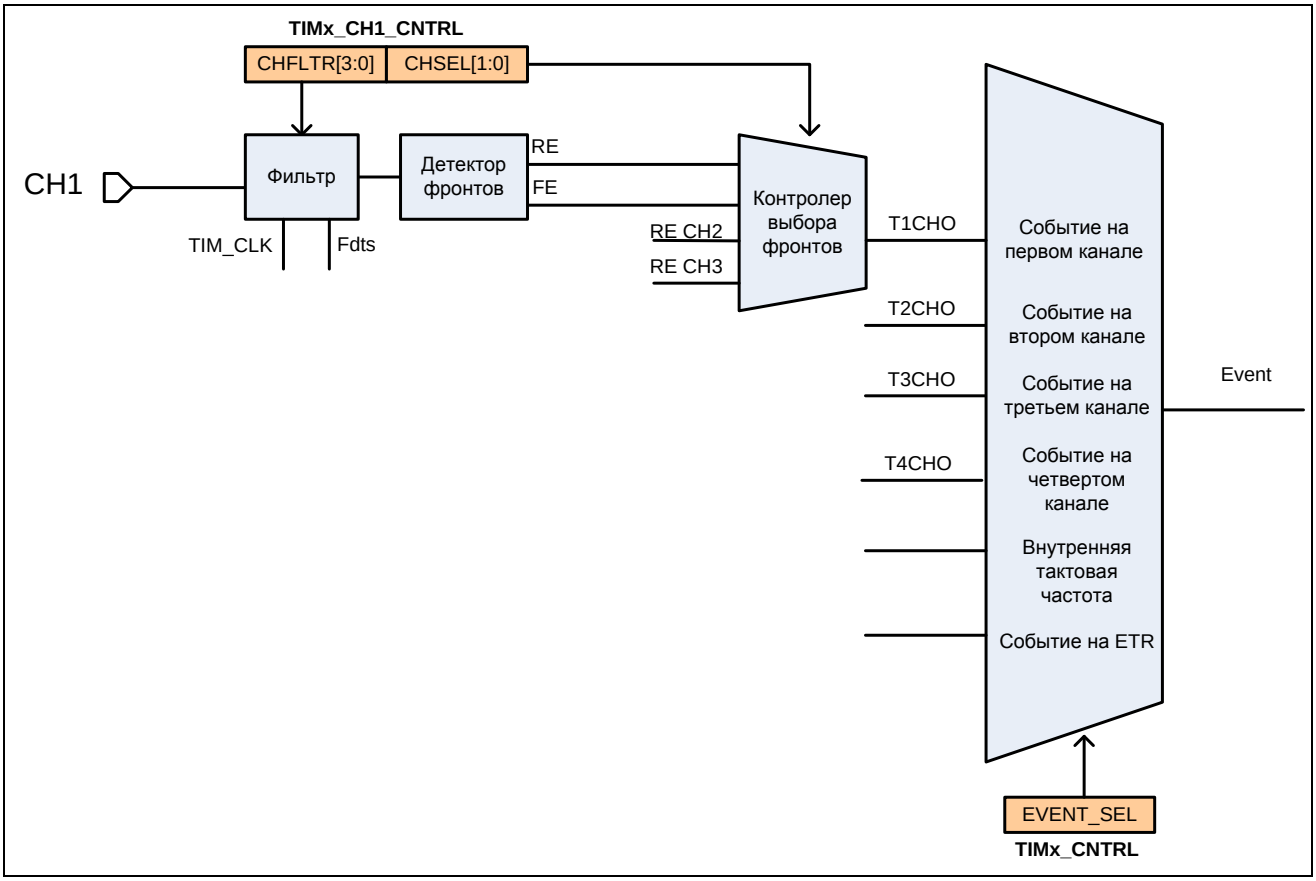


Рисунок 75. Тактирование с входа первого канала

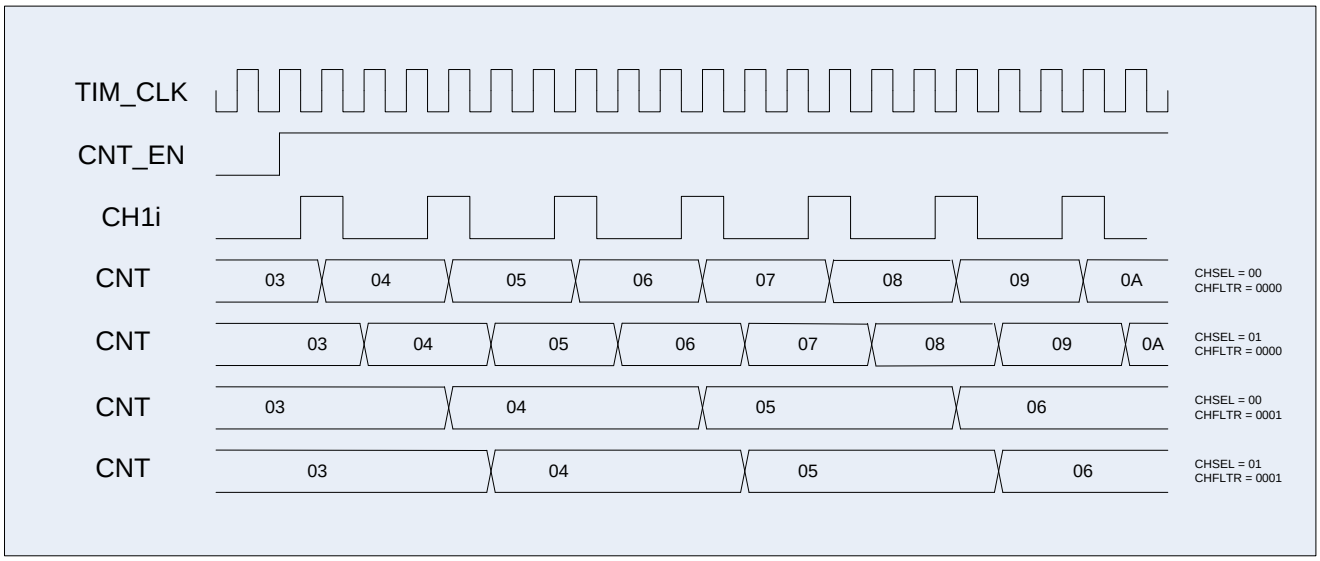


Рисунок 76. Диаграмма внешнего тактирования с разными вариантами фильтра

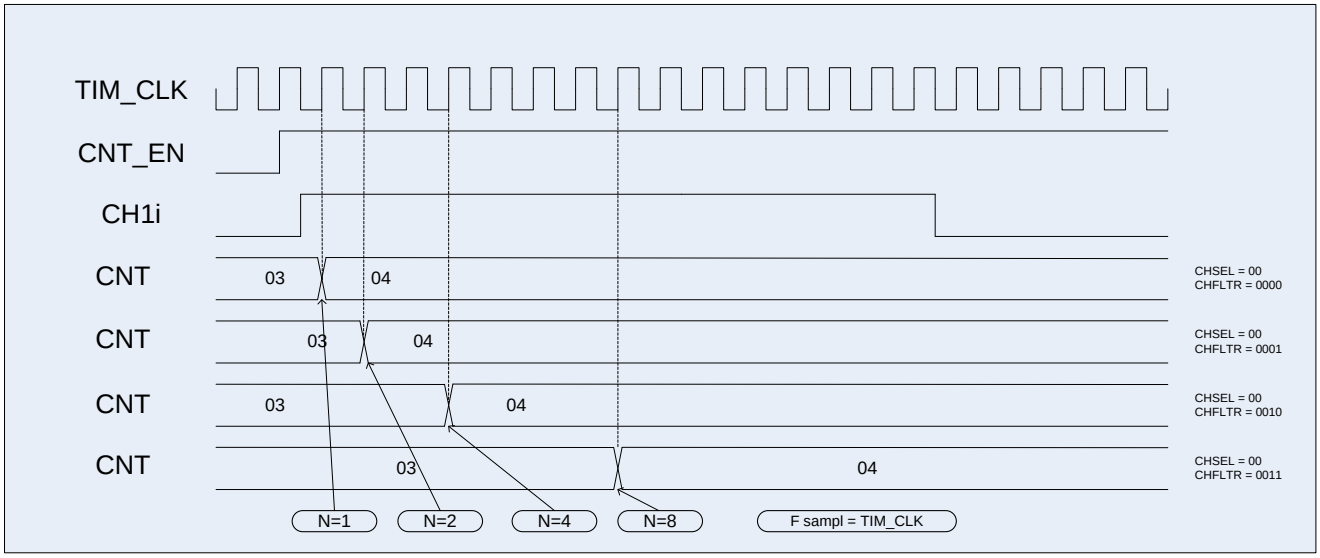


Рисунок 77. Диаграмма внешнего тактирования с разными вариантами фильтра

22.3.4 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.

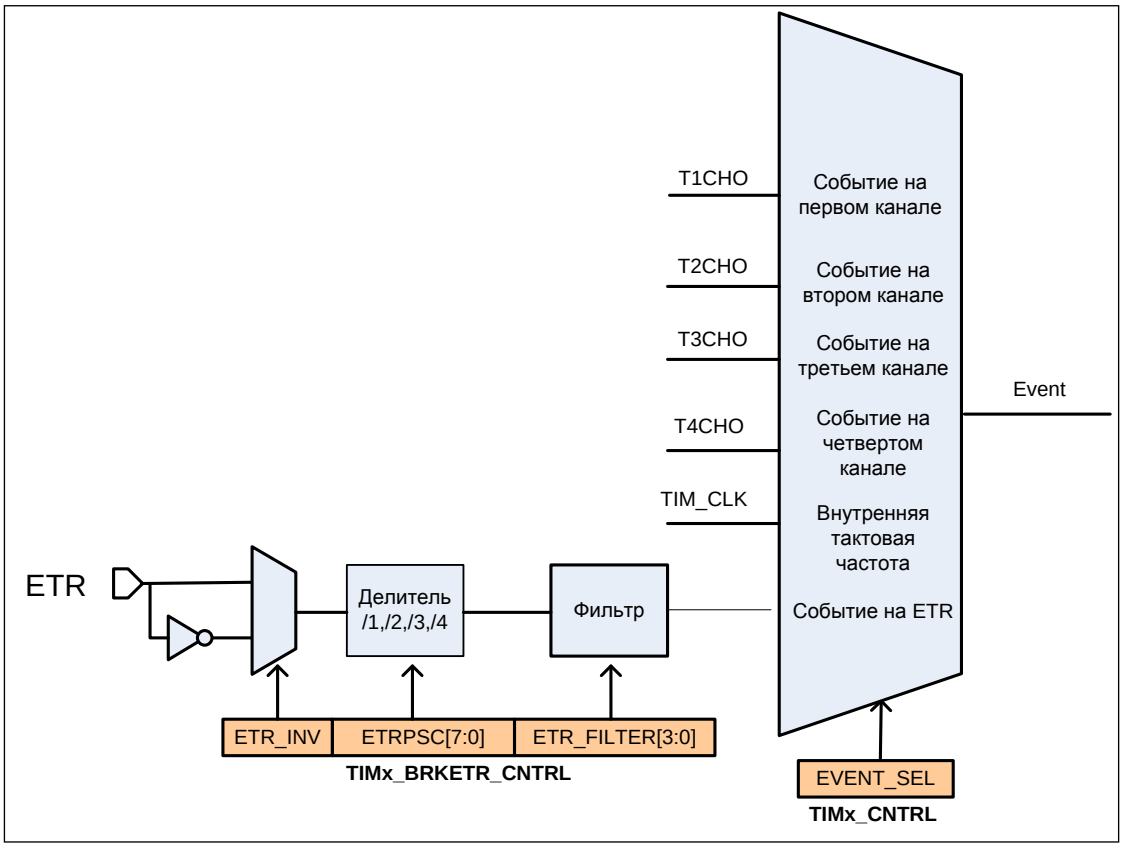


Рисунок 78. Схема тактирования сигналом со входа ETR

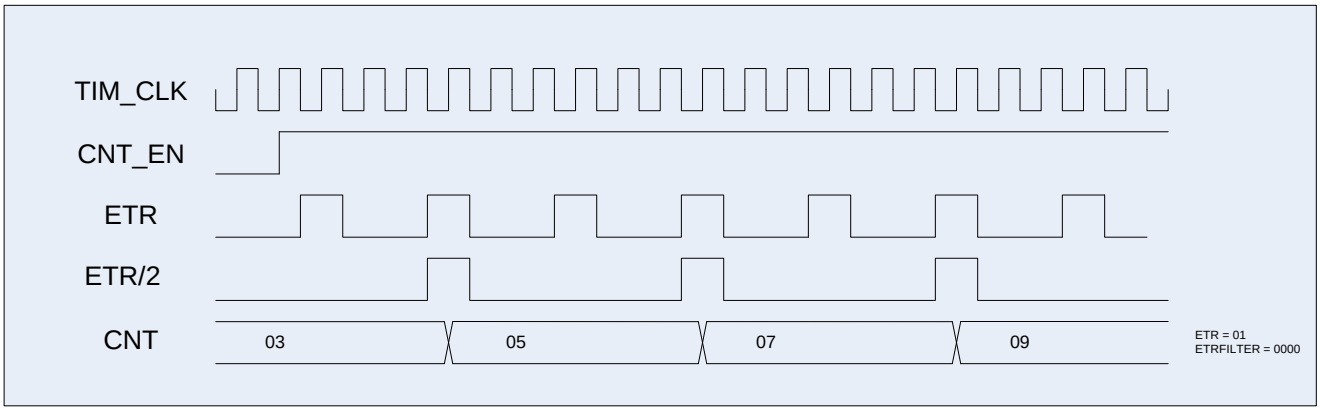


Рисунок 79. Диаграмма тактирования сигналом со входа ETR

22.4 Режим захвата

Структурная схема блока захвата представлена на Рисунок 80.

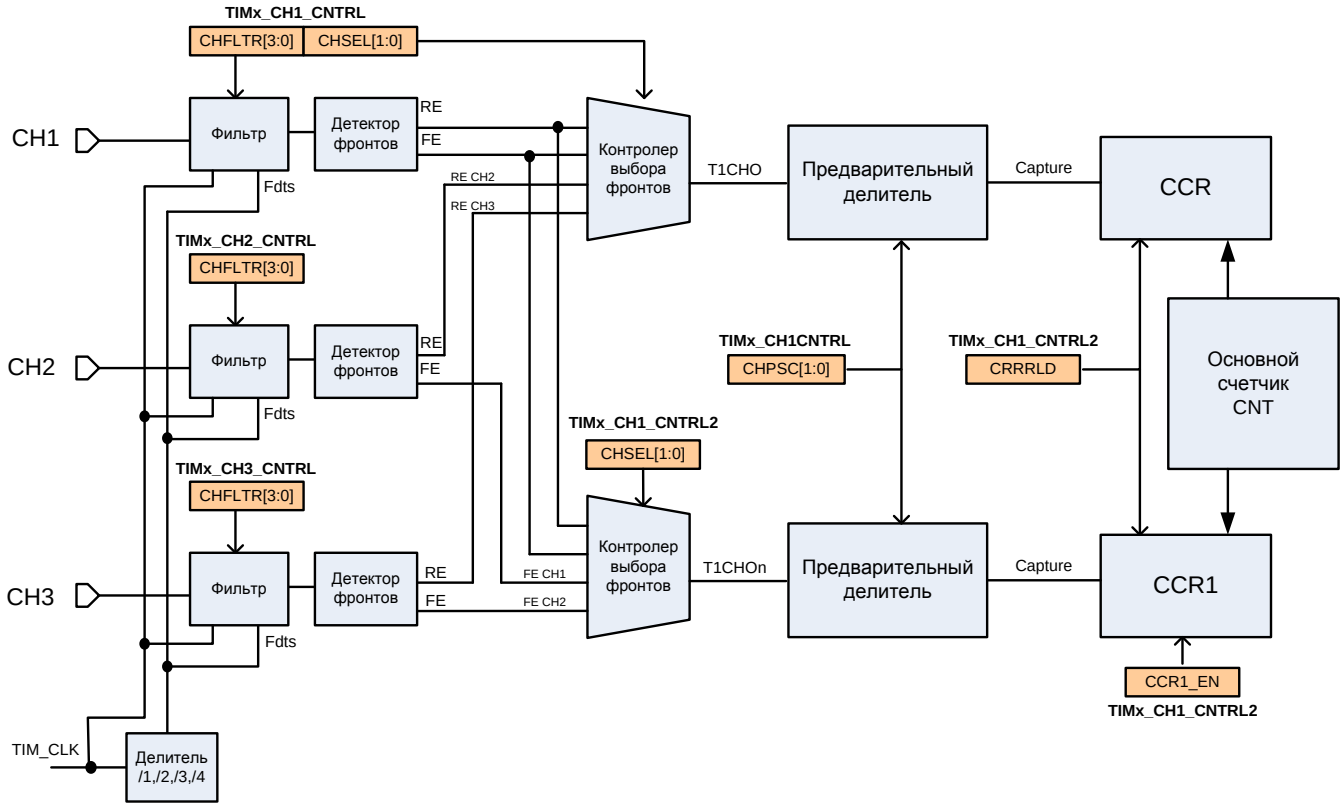


Рисунок 80. Структурная схема блока захвата на примере канала 1

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.

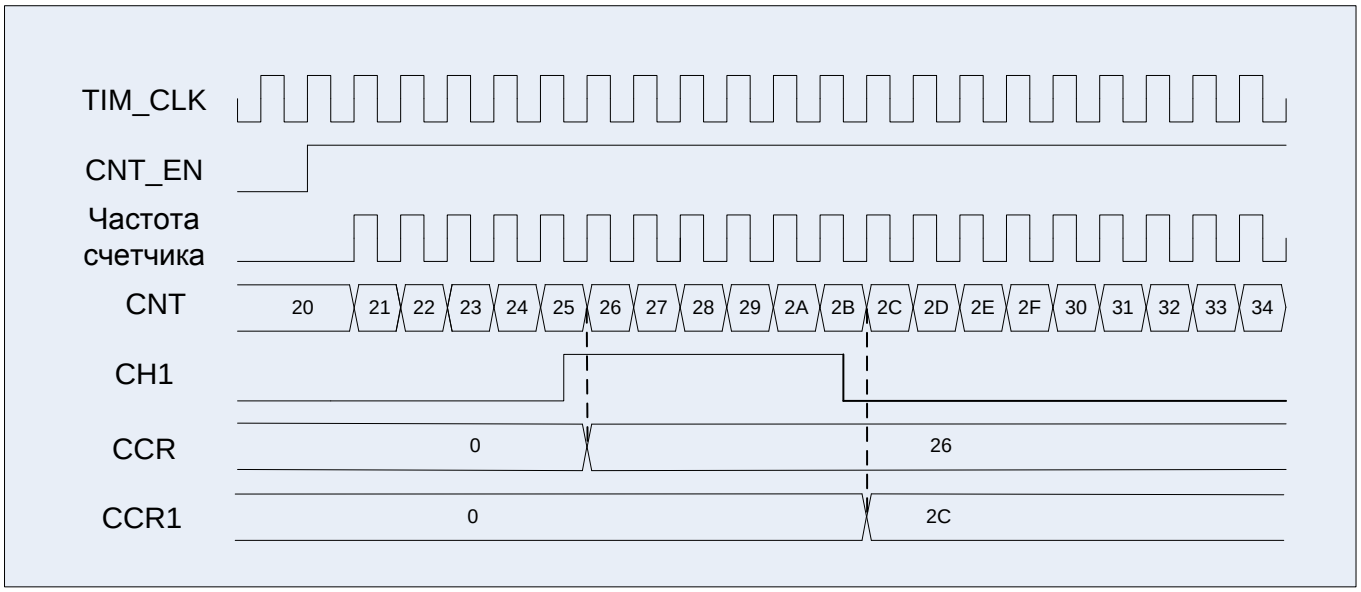


Рисунок 81. Диаграмма захвата события со входа первого канала

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.

22.5 Режим ШИМ

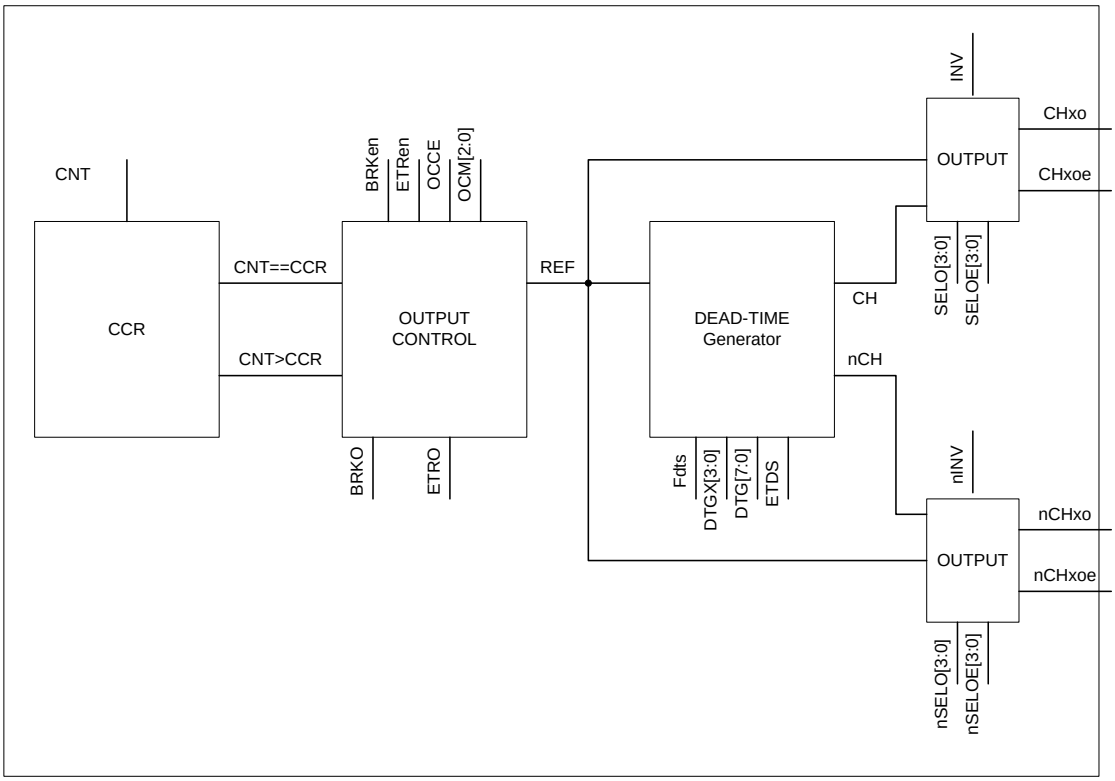


Рисунок 82. Структурная схема блока формирования ШИМ

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.

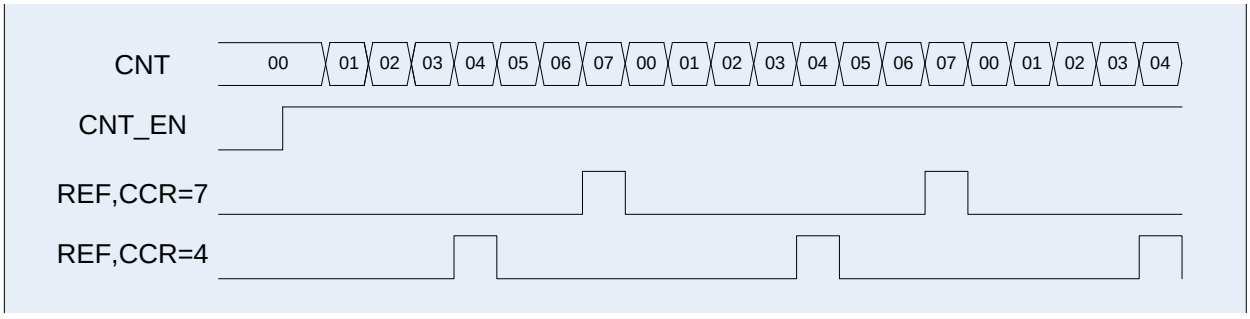


Рисунок 83. Диаграмма работы схемы в режиме ШИМ, CCR1_EN=0

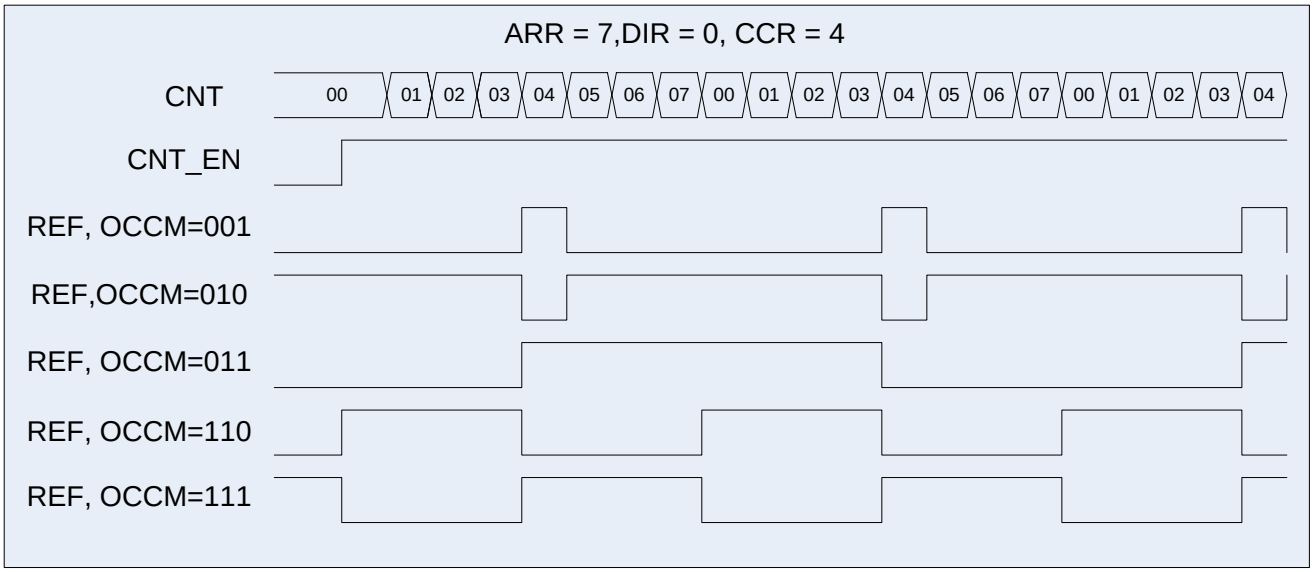


Рисунок 84. Диаграмма работы схемы в режиме ШИМ, CCR1_EN = 0

Сигнал REF может быть очищен с использованием внешнего сигнала со входа ETR, или внешнего, синхронизированного по PCLK сигналу со входа BRK.

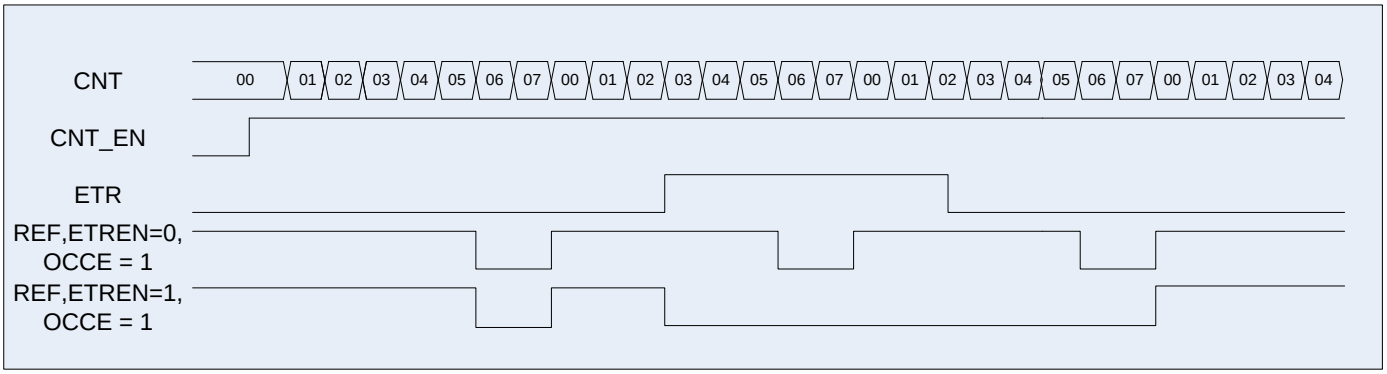


Рисунок 85. Диаграмма работы схемы в режиме ШИМ, CCR1_EN = 0

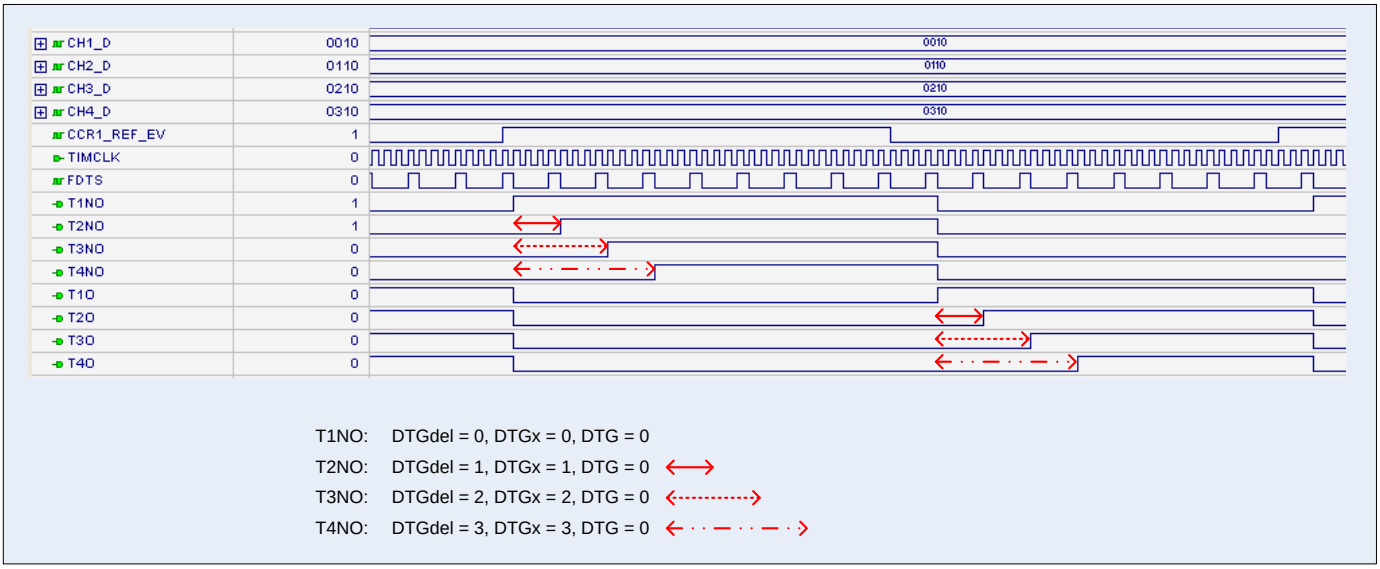


Рисунок 86. Диаграмма работы схемы DTG

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.

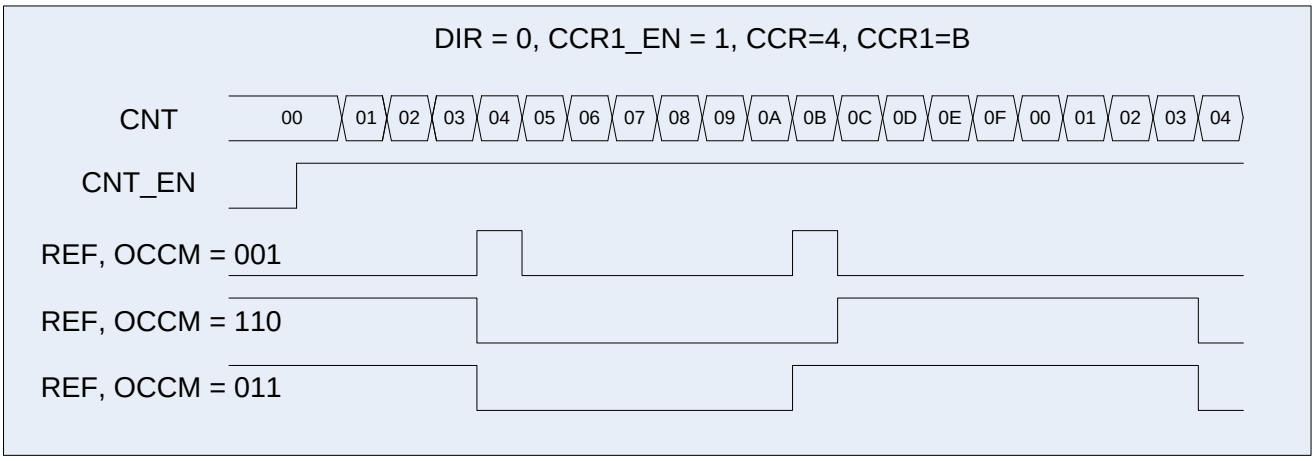


Рисунок 87. Диаграмма работы схемы в режиме ШИМ, CCR1_EN = 1

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.

22.6 Примеры

22.6.1 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.
```

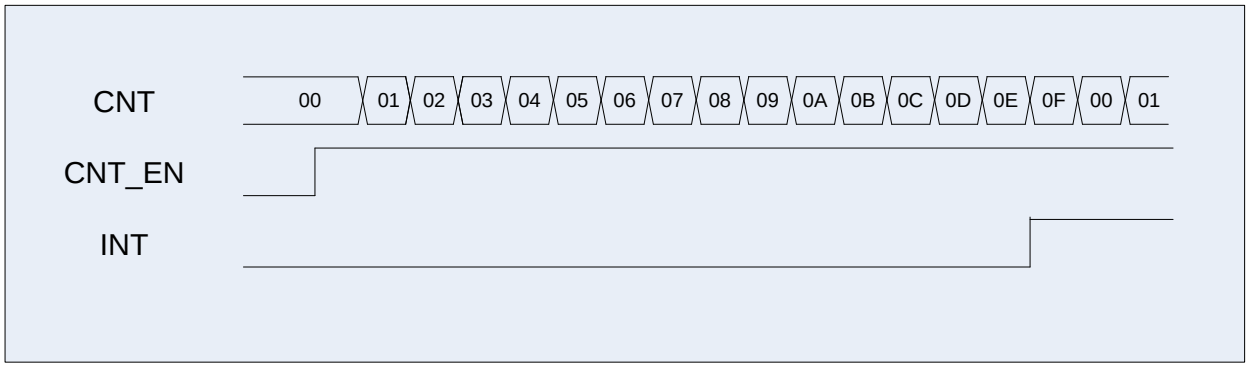


Рисунок 88. Режим обычного счетчика

22.6.2 Режим захвата

```
    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 = 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
```

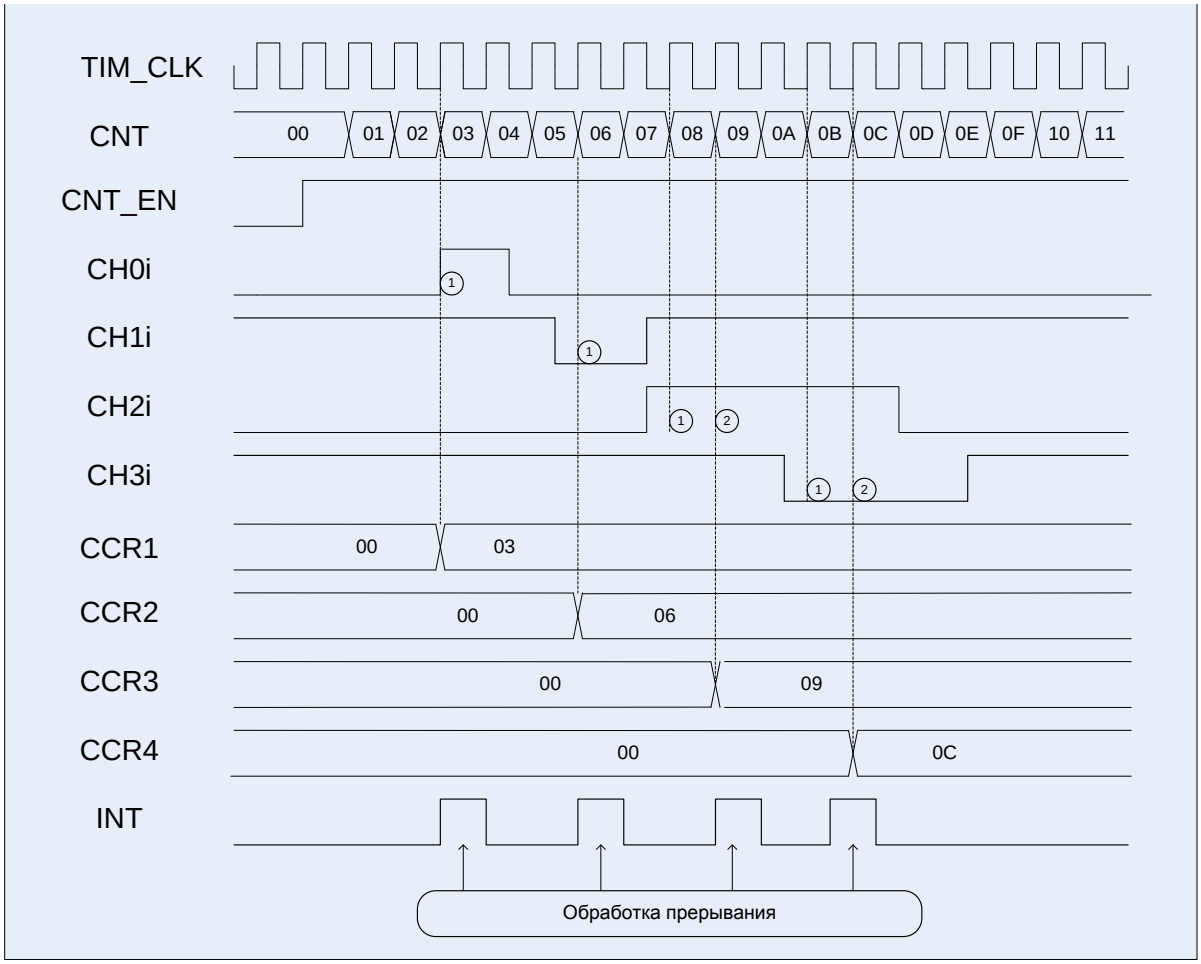


Рисунок 89. Диаграммы примера работы в режиме захвата

22.6.3 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.
```

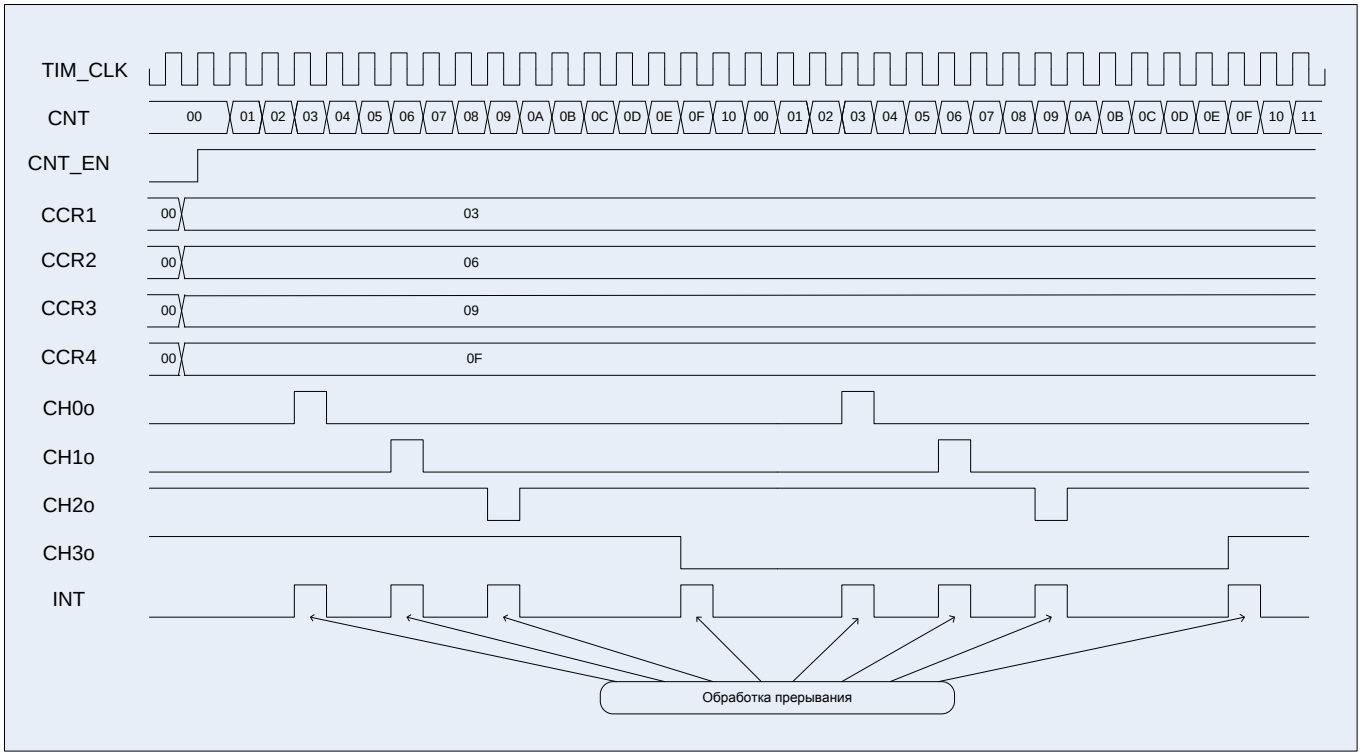


Рисунок 90. Диаграммы примера работы в режиме ШИМ

22.7 Описание регистров блока таймера

Таблица 264 – 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

Адрес	Название	Описание
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

22.7.1 MDR_TIMERx->CNT

Таблица 265 – Основной счетчик таймера CNT

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

Таблица 266 – Описание бит регистра CNT

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31…16	-	Зарезервировано
15…0	CNT[15:0]	Значение основного счетчика таймера

22.7.2 MDR_TIMERx->PSG

Таблица 267 – Делитель частоты при счете основного счетчика PSG

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

Таблица 268 – Описание бит регистра PSG

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31…16	-	Зарезервировано
15…0	PSG[15:0]	Значение предварительного делителя счетчика Основной счетчик считает на частоте: CLK = TIM_CLK/(PSG+1)

22.7.3 MDR_TIMERx->ARR

Таблица 269 – Основание счета основного счетчика ARR

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

Таблица 270 – Описание бит регистра ARR

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31…16	-	Зарезервировано
15…0	ARR[15:0]	Основание счета для основного счетчика: CNT = [0…ARR]

22.7.4 MDR_TIMERx->CNTRL

Таблица 271 – 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 272 – 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 – событие переднего фронта на TIM_CLK; 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 счетчик вниз при DIR=1; 11 – счетчик вверх/вниз с автоматическим изменением DIR Режимы CNT_MODE[1:0] = 00 и CNT_MODE[1:0] = 01 действуют, если EVENT_SEL = 0000; режим CNT_MODE[1:0] = 10 и CNT_MODE[1:0] = 11 действуют, если EVENT_SEL ! = 0000
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

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
0	CNT_EN	Разрешение работы таймера: 0 – таймер отключен; 1 – таймер включен

22.7.5 MDR_TIMERx->CCRy

Таблица 273 – Регистр сравнения/захвата для ‘y’ канала таймера CCRy

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

Таблица 274 – 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 ШИМ режиме. Значение CNT, при котором произошел факт захвата события, в режиме захвата

22.7.6 MDR_TIMERx->CCRy1

Таблица 275 – Регистр сравнения/захвата для ‘y’ канала таймера CCRy1

Номер	31...16	15...0
Доступ	U	R/W
Сброс	0	0
	-	**CCR1[15:0]**

Таблица 276 – 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 ШИМ режиме. Значение CNT, при котором произошел факт захвата события, в режиме захвата

22.7.7 MDR_TIMERx->CHy_CNTRL

Таблица 277 – 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	RO	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 278 – 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), CCR< CNT< CCR1, иначе 0; 0, если DIR = 1 (счет вниз), CCR < CNT < CCR1, иначе 1; 111 – 0, если DIR = 0 (счет вверх), CCR< CNT < CCR1, иначе 1; 1, если DIR = 1 (счет вниз), CCR< CNT< CCR1, иначе 0 При условии, что CCR < CCR1

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
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 CHxi для фиксации значения основного счетчика (регистр MDR_TIMERx->CNT) в регистр CCR: 00 – положительный фронт на входном канале CHxi; 01 – отрицательный фронт на входном канале CHxi; 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

22.7.8 MDR_TIMERx->CHy_CNTRL1

Таблица 279 – 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 280 – 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

22.7.9 MDR_TIMERx->CHy_CNTRL2

Таблица 281 – Регистр управления 2 для ‘y’ канала таймера CHy_CNTRL2

Номер	31... 5	4	3	2	1…0
Доступ	U	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	00
	-	**EV_DELAY**	**CCRRLD**	**CCR1_EN**	**CHSEL [1:0]**

Таблица 282 – Описание бит регистра CHy_CNTRL2

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31...4	-	Зарезервировано
4	EV_DELAY	Задержка события захвата до обновления регистров CCR и CCR1: 0 – событие захвата происходит в момент обнаружения события, асинхронно с обновлением регистров CCR и CCR1; 1 – событие захвата происходит синхронно с обновлением информации в регистрах CCR и CCR1
3	CCRRLD	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 CHxi для фиксации значения основного счетчика (регистр MDR_TIMERx->CNT) в регистр CCR1: 00 – положительный фронт на входном канале CHxi; 01 – отрицательный фронт на входном канале CHxi; 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

22.7.10 MDR_TIMERx->CHy_DTG

Таблица 283 – 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 284 – 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 DTG**x**

22.7.11 MDR_TIMERx->BRKETR_CNTRL

Таблица 285 – 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 286 – 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 предделитель 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

22.7.12 MDR_TIMERx->STATUS

Таблица 287 – 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 288 – 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 настроенного фронта на входе CHxi канала таймера: 0 – нет события; 1 – есть событие. Сбрасывается записью 0. Если запись происходит одновременно с новым событием, то 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 происходит одновременно с новым событием, то приоритет у нового события. Бит 0 – первый канал. Бит 3 – четвертый канал
8…5	CCR CAP EVENT[3:0]	Событие настроенного фронта на входе CHxi канала таймера: 0 – нет события; 1 – есть событие. Сбрасывается записью 0. Если запись происходит одновременно с новым событием, то 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 происходит одновременно с новым событием, то приоритет у нового события.
2	ETR RE EVENT	Событие переднего фронта на входе ETR: 0 – нет события; 1 – есть событие. Сбрасывается записью 0. Если запись происходит одновременно с новым событием, то приоритет у нового события.

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
1	CNT ARR EVENT	Событие совпадения CNT с ARR: 0 – нет события; 1 – есть событие. Сбрасывается записью 0. Если запись происходит одновременно с новым событием, то 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 происходит одновременно с новым событием, то приоритет у нового события. Если с момента совпадения до момента программного сброса CNT не изменил состояния, то флаг повторно не взводится

22.7.13 MDR_TIMERx->IE

Таблица 289 – 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 290 – 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 настроенного фронта на входе CHxi канала таймера (фиксация значения основного счетчика таймера в регистре CCR1): 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 настроенного фронта на входе CHxi канала таймера (фиксация значения основного счетчика таймера в регистре CCR): 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

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

22.7.14 MDR_TIMERx->DMA_RE

Таблица 291 – 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 292 – 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 настроенного фронта на входе CHxi канала таймера (фиксация значения основного счетчика таймера в регистре CCR1): 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 настроенного фронта на входе CHxi канала таймера (фиксация значения основного счетчика таймера в регистре CCR): 0 – нет запроса DMA; 1 – запрос DMA разрешен. Бит 0 – первый канал. Бит 3 – четвертый канал

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

23 Контроллер MDR_ADC

В микроконтроллере реализовано два 12-разрядных АЦП. С помощью АЦП можно оцифровать сигнал от 16 внешних аналоговых выводов порта D и от двух внутренних каналов, на которые выводятся датчик температуры и источник опорного напряжения. Скорость выборки составляет до 512 тысяч преобразований в секунду для каждого АЦП.

В качестве опорного напряжения преобразования могут выступать:

– питание АЦП с выводов AUCC и AGND;

– внешние сигналы с выводов ADC0_REF+ и ADC_REF-.

Контроллер АЦП позволяет:

- оцифровать один из 16 внешних каналов;
- оцифровать значение встроенного датчика температуры;
- оцифровать значение внутреннего 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 28 тактов синхронизации CLK. В качестве синхросигнала может выступать частота процессора CPU_CLK либо частота ADC_CLK, формируемая в блоке «Сигналы тактовой частоты». Выбор частоты осуществляется с помощью бита Cfg_REG_CLKS. В контроллере АЦП частота CPU_CLK может быть поделена с помощью битов Cfg_REG_DIVCLK[3:0].

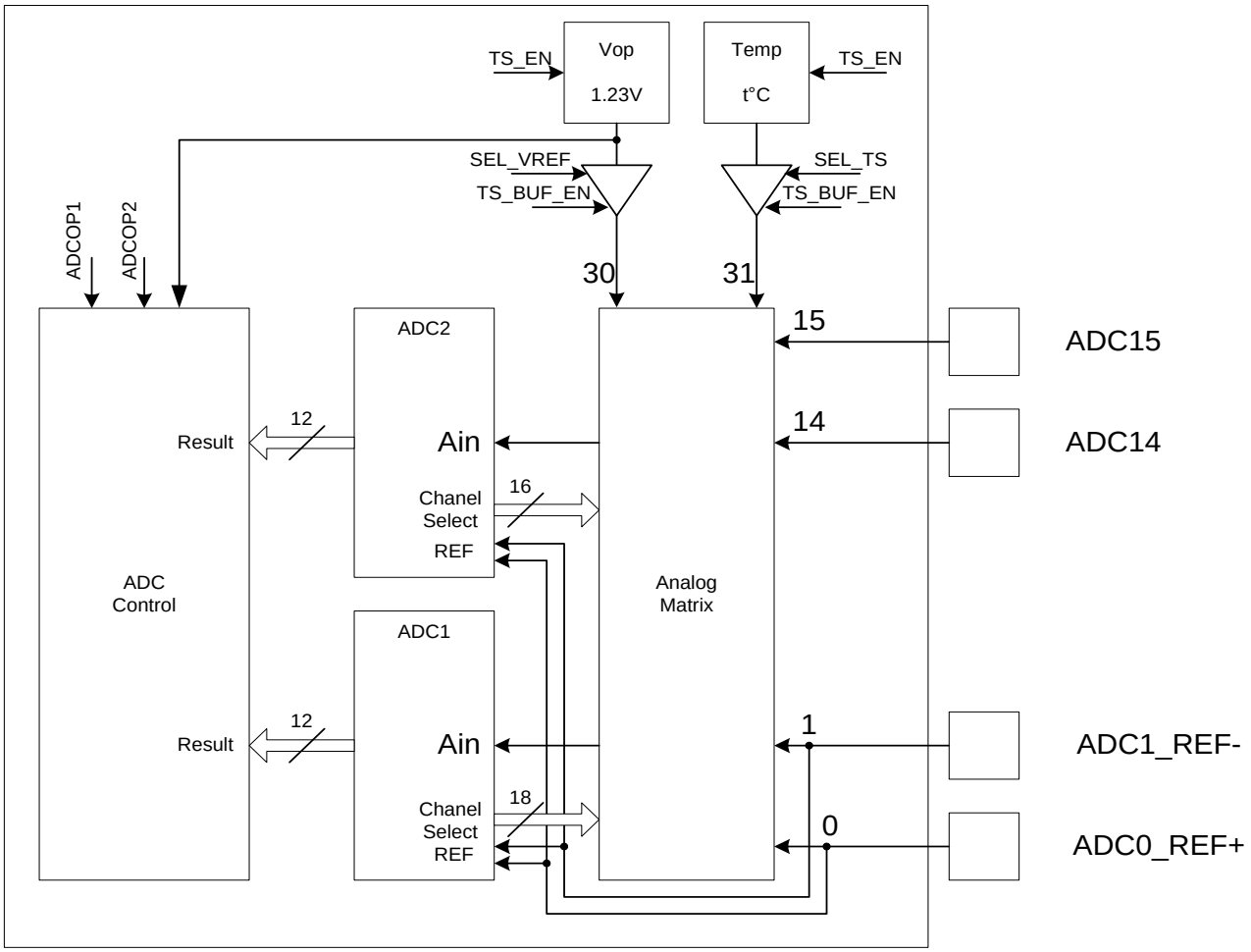


Рисунок 91. Структурная схема контроллера АЦП

Для включения АЦП необходимо установить бит Cfg_REG_ADON. Для снижения тока потребления вместо собственного источника опорного напряжения в АЦП может использоваться источник опорного напряжения 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.

23.1 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 случая опора берется с выводов ADC0_REF+ и ADC1_REF-. Биты Cfg_REG_CHCH, Cfg_REG_RNGC, Cfg_REG_SAMPLE, TS_BUF_EN, SEL_VREF, SEL_TS и Cfg_Sync_Conver должны быть сброшены.

Для начала преобразования необходимо записать 1 в бит Cfg_REG_GO.

После завершения преобразования будет взведен бит Flg_REG_EOCIF в регистре ADCx_STATUS, а в регистре ADCx_RESULT будет результат преобразования.

После считывания результата бит Flg_REG_EOCIF сбросится.

Если после первого преобразования результат не был считан, и было выполнено второе преобразование, то в регистре результата ADCx_RESULT будет значение от последнего преобразования, и помимо бита Flg_REG_EOCIF будет взведен бит Flg_REG_OVERWRITE. Флаг Flg_REG_OVERWRITE может быть сброшен только записью в регистр ADCx_STATUS.

23.2 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 Cfg_REG_SAMPLE и Cfg_REG_CHCH должны быть установлены. С помощью бит Delay_GO можно задать паузу между преобразованиями при переборе каналов. Эта определяется в тактах CPU_CLK, независимо от того на какой частоте ADC_CLK или CPU_CLK идет само преобразование. Для начала преобразования необходимо записать 1 в бит Cfg_REG_GO.

После завершения преобразования будет взведен бит Flg_REG_EOCIF в регистре ADCx_STATUS, а в регистре ADCx_RESULT будет результат преобразования.

После считывания результата бит Flg_REG_EOCIF сбросится.

Если после первого преобразования результат не был считан, и было выполнено второе преобразование, то в регистре результата ADCx_RESULT будет значение от последнего преобразования, и помимо бита Flg_REG_EOCIF будет взведен бит Flg_REG_OVERWRITE. Флаг Flg_REG_OVERWRITE может быть сброшен только записью в регистр ADCx_STATUS.

Для последовательного преобразования одного и того же канала можно в регистре ADCx_CHSEL выбрать только один канал и установить бит Cfg_REG_CHCH в 1, либо установить номер канала в битах Cfg_REG_CHS[4:0] и сбросить бит Cfg_REG_CHCH в 0. В этом случае процесс последовательного преобразования будет выполняться только для данного канала. Последовательное преобразование значения датчика температуры и источника опорного напряжения могут выполняться только в режиме последовательного преобразования одного канала.

23.3 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.

23.4 Внутренний источник опорного напряжения

С помощью первого АЦП можно осуществить преобразования внутреннего источника опорного напряжения (блок V_{OP} на рисунке 91). Для этого необходимо включить блок датчика температуры и источник опорного напряжения, установив бит TS_EN в 1. После включения можно использовать источник опорного напряжения для первого и второго АЦП вместо их собственных, что позволяет снизить ток потребления. Для этого необходимо установить биты ADCx_OP в единицу. Для выбора источника опорного напряжения в качестве источника для преобразования необходимо в битах Cfg_REG_CHS установить значение 30 канала, установить

биты TS_BUF_EN и SEL_VREF, после чего можно запустить процесс преобразования. Для запуска преобразования необходимо записать 1 в бит Cfg_REG_GO.

После завершения преобразования будет взведен бит Flg_REG_EOCIF в регистре ADC1_STATUS, а в регистре ADC1_RESULT будет результат преобразования.

После считывания результата бит Flg_REG_EOCIF сбросится.

Если после первого преобразования результат не был считан, и было выполнено второе преобразование, то в регистре результата ADC1_RESULT будет значение от последнего преобразования, а помимо бита Flg_REG_EOCIF будет взведен бит Flg_REG_OVERWRITE. Флаг Flg_REG_OVERWRITE может быть сброшен только записью в регистр ADC1_STATUS.

Для последовательного преобразования только источника опорного напряжения можно в регистре ADC1_CHSEL выбрать только 30 канал и установить бит Cfg_REG_CHCH в 1, либо установить номер 30-го канала в битах Cfg_REG_CHS[4:0] и сбросить бит Cfg_REG_CHCH в 0. В этом случае процесс последовательного преобразования будет выполняться только для данного канала. При этом должны быть также установлены биты TS_BUF_EN и SEL_VREF.

Источник опорного напряжения может быть выбран для более точного результата преобразования АЦП и не может быть использован для задания опорного напряжения преобразования.

23.5 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.

23.6 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.

23.7 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 R_{AIN} для обеспечения качественного преобразования:

$$R_{AIN} < (T_S/(f_{C\ ADC}*C_{ADC}*\ln(2^N)))-R_{ADC}$$

где:

T_S - время заряда внутренней емкости в тактах;
$f_{C\ ADC}$ - рабочая частота АЦП;
C_{ADC} - внутренняя емкость АЦП (~ 15 – 20 пФ);
N - требуемая точность, в разрядах;
R_{ADC} - входное сопротивление тракта АЦП (~500 Ом).

Если необходимо обеспечить преобразование с точностью 12 разрядов ± 1/4 LSB, то N = 14. Если необходимо обеспечить преобразование с точностью 10 разрядов ± 1 LSB, то N=10. Время заряда T_S = определяется битами DelayGo[2:0] и схемой самого АЦП и представлена в Таблица 293. 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 293 – Время заряда внутренней емкости АЦП и время преобразования

DelayGo[2:0]	Дополнительная задержка перед началом преобразования	Общее время Ts 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 E_{DLADC}, E_{ILADC} и E_{OFFADC}.

Задание режимов работы АЦП в регистре ADCx_CFG необходимо производить до задания бита Cfg_REG_GO, иначе новая конфигурация будет действовать со следующего преобразования.

23.8 Описание регистров блока контроллера АЦП

Таблица 294 – 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

23.8.1 MDR_ADC->ADC1_CFG

Таблица 295 – 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 REG 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 296 – 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. См. диаграмму на рисунке 92
20	SEL VREF	Выбор для оцифровки внутреннего источника опорного напряжения 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

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
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 течение 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; 0001 – CPU_CLK/2; 0010 – CPU_CLK/4; 0011 – CPU_CLK/8; … 1011 – CPU_CLK/2048 Остальные комбинации – CPU_CLK
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

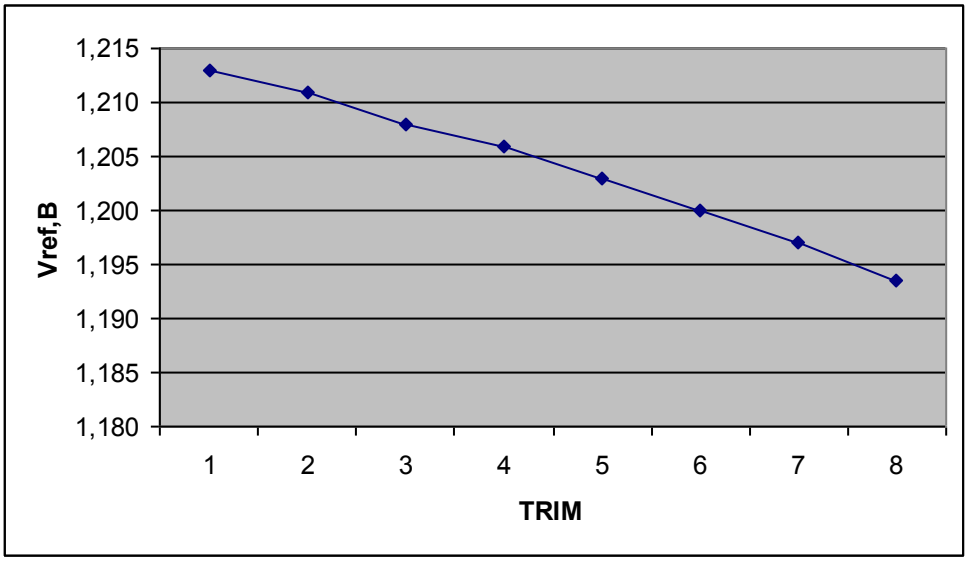


Рисунок 92. Зависимость источника опорного напряжения от подстройки

23.8.2 MDR_ADC->ADC2_CFG

Таблица 297 – 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 298 – 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 – 1 тактов CPU_CLK 001 – 2 такт CPU_CLK … 111 – 8 тактов CPU_CLK
24...19	-	Зарезервировано

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
18	ADC2 OP	Выбор источника опорного напряжения: 0 – внутренний (неточный); 1 – от датчика температуры (точный). Необходимо чтобы бит TS EN был равен 1
17	ADC1 OP	Выбор источника опорного напряжения: 0 – внутренний (неточный); 1 – от датчика температуры (точный). Необходимо чтобы бит TS EN был равен 1
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

23.8.3 MDR_ADC->ADCx_H_LEVEL

Таблица 299 – Регистр ADCx_H_LEVEL

Номер	31...12	11…0
Доступ	U	R/W
Сброс	0	0
	-	**REG H LEVEL [**11:0]

Таблица 300 – 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.

23.8.4 MDR_ADC->ADCx_L_LEVEL

Таблица 301 – Регистр ADCx_L_LEVEL

Номер	31...12	11…0
Доступ	U	R/W
Сброс	0	0
	-	**REG L LEVEL [11:0]**

Таблица 302 – 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

23.8.5 MDR_ADC->ADCx_RESULT

Таблица 303 – Регистр ADCx_RESULT

Номер	31...21	20…16	15…12	11…0
Доступ	U	RO	U	RO
Сброс	0	0	0	0
	-	**CHANNEL [4:0]**	-	**RESULT [11:0]**

Таблица 304 – 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

23.8.6 MDR_ADC->ADCx_STATUS

Таблица 305 – 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
	-	**EOCIF IE**	**AWOIF IE**	**Flg REG EOCIF**	**Flg REG AWOIFEN**	**Flg REG OVERWRITE**

Таблица 306 – Описание бит регистра ADCx_STATUS

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31…5	-	Зарезервировано
4	EOCIF_IE	Флаг разрешения генерирования прерывания по событию Flg_REG_EOCIF: 0 – прерывания не генерируется; 1 – прерывание генерируется
3	AWOIF_IE	Флаг разрешения генерирования прерывания по событию Flg_REG_AWOIFEN: 0 – прерывания не генерируется; 1 – прерывание генерируется
2	Flg REG EOCIF	Флаг выставляется, когда закончено преобразования и данные еще не считаны. Очищается считыванием результата из регистра ADCx_RESULT: 1 – есть готовый результат преобразования; 0 – нет результата
1	Flg REG AWOIFEN	Флаг выставляется, когда результат преобразования выше верхней или ниже нижней границы автоматического контролирования уровней. Сбрасывается только при записи нуля в данный бит регистр флагов: 0 – результат в допустимой зоне; 1 – вне допустимой зоны
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

23.8.7 MDR_ADC->ADCx_CHSEL

Таблица 307 – Регистр ADCx_CHSEL

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

Таблица 308 – 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

24 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.

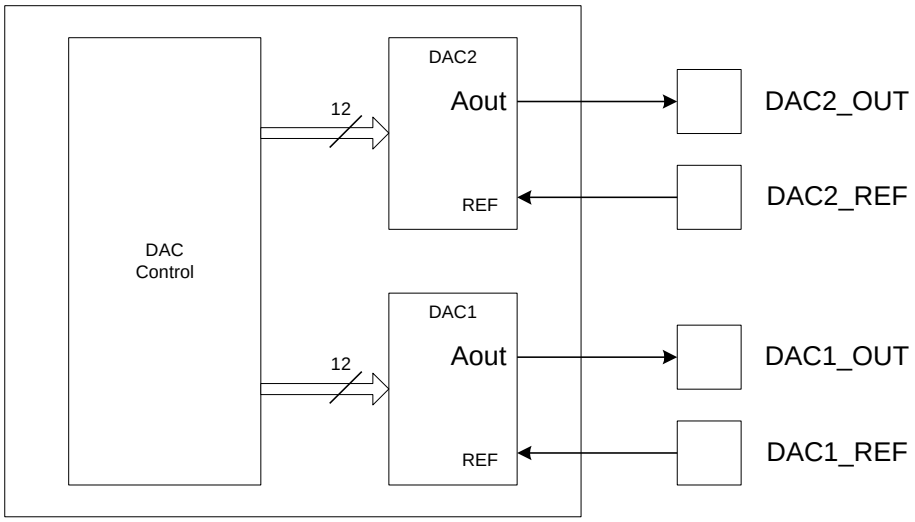


Рисунок 93. Структурная схема контроллера ЦАП

24.1 Описание регистров блока контроллера ЦАП

Таблица 309 – 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

24.1.1 MDR_DAC->CFG

Таблица 310 – 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 311 – 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 AU_{CC}; 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 AU_{CC}; 1 – в качестве опорного напряжения используется напряжение на входе опорного напряжения DAC1_REF

24.1.2 MDR_DAC->DAC1_DATA

Таблица 312 – 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 313 – 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

24.1.3 MDR_DAC->DAC2_DATA

Таблица 314 – 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 315 – 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.

25 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.

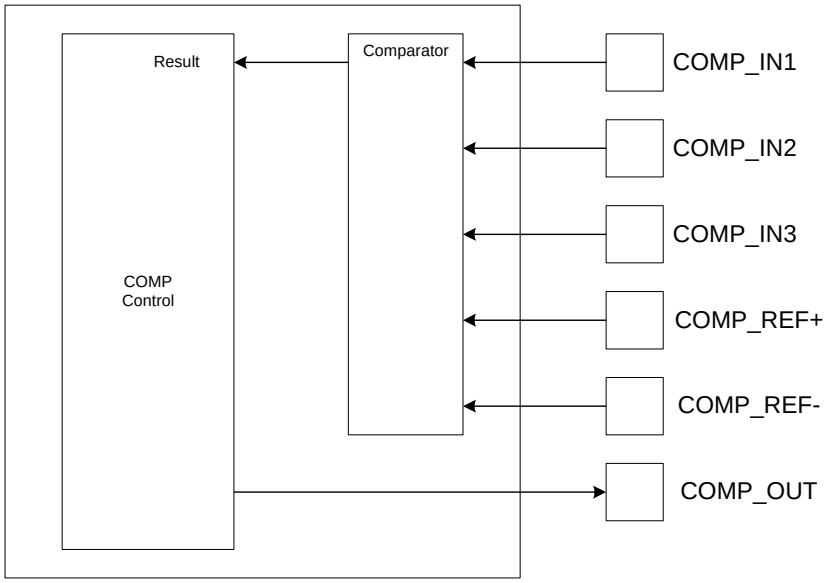
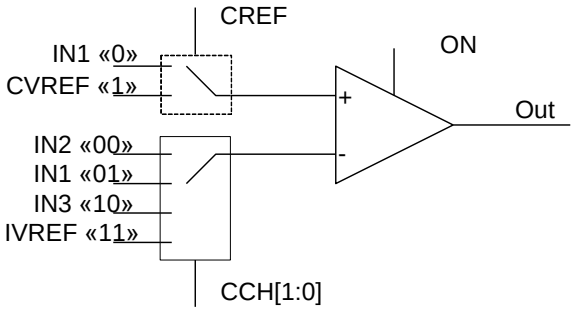


Рисунок 94. Структура блока компаратора



*IVREF – выход внутреннего источника опорного напряжения 1,2 В.

Рисунок 95. Структура мультиплексирования входов компаратора

25.1 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 IN3 (CCH=10), при этом, если уровень на входе «+» будет больше уровня на входе «–», то выход Out установится в 1.

25.2 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.

25.3 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 IN3 (CCH=10), при этом, если уровень на входе «+» будет больше уровня на входе «–», то выход Out установится в 1.

25.4 Формирование внутренней шкалы напряжений

Внутренняя шкала напряжений формируется на резистивном делителе (см Рисунок 96), который включается битом CVREN=1.

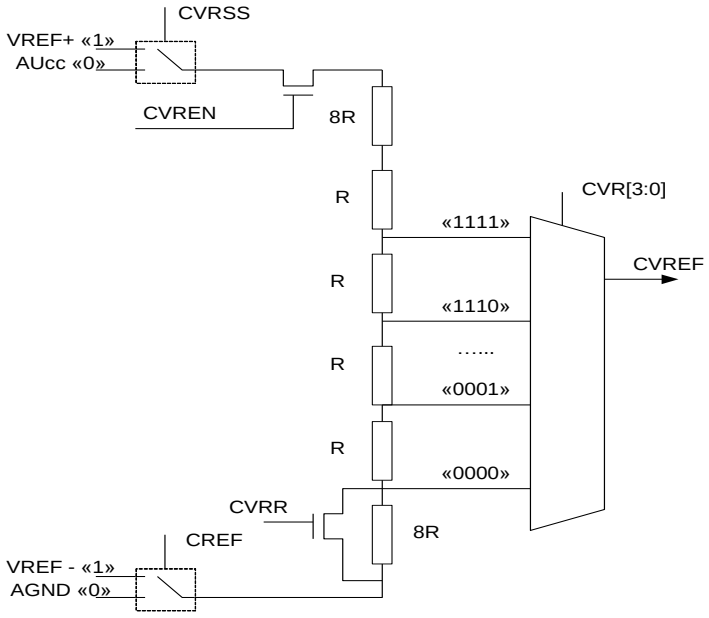


Рисунок 96. Структура блока формирования CVREF

При этом в качестве опорного напряжения делителя может выступать питание микросхемы AU_{CC} (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 316, как справочные данные. Реальные значения в конкретном кристалле могут отличаться за счет технологического разброса параметров.

Таблица 316 – Формирование внутренней шкалы напряжений CVREF

CVRR	CVR[3:0]	Отношение резисторов	Напряжение CREF при U_{CC}=3.3 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_lch фиксирует событие появления положительного сигнала сравнения и устанавливается в 1 до тех пор, пока не будет считан регистр COMP_RESULT_LATCH.

25.5 Описание регистров блока контроллера компаратора

Таблица 317- 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

25.5.1 MDR_COMP->CFG

Таблица 318 – Регистр CFG

Номер	31…14	13	12	11	10…9	8	7…4	3	2	1	0
Доступ	U	R/W	RO	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0	0	0	0	0	0
	-	**CMP IE**	**Ready**	**INV**	**CCH [1:0]**	**CREF**	**CVR [3:0]**	**CVR EN**	**CVRSS**	**CVRR**	**ON**

Таблица 319 – 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. См. таблицу 316
3	CVREN	Бит разрешения работы источника CVREF: 0 – не разрешен; 1 – разрешен

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
2	CVRSS	Бит выбора опоры CVREF: 0 – источник CVREF работает в границах 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

25.5.2 MDR_COMP->RESULT

Таблица 320 – Регистр RESULT

Номер	31…3	2	1	0
Доступ	U	R/W	R/W	R/W
Сброс	0	0	0	0
	-	**Rst_lch**	**Rslt_As**	**Rslt_Sy**

Таблица 321 – Описание бит регистра RESULT

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31…3	-	Зарезервировано
2	Rst_lch	Значение компарирования хранится до момента считывания из регистра COMP_RESULT_LATCH, после чего сбрасывается. Взводится по переднему фронту сигнала с компаратора
1	Rslt_As	Значение компарирования непосредственно из компаратора
0	Rslt_Sy	Значение результата компарирования, синхронизированное с частотой HCLK

25.5.3 MDR_COMP->RESULT_LATCH

Таблица 322 – Регистр RESULT_LATCH

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

Таблица 323 – Описание бит регистра RESULT_LATCH

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31…1	-	Зарезервировано
0	Rst_lch	Значение компарирования хранится до момента считывания из регистра COMP_RESULT_LATCH, после чего сбрасывается. Взводится по переднему фронту сигнала с компаратора

26 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. Контроллер интерфейса I2C в микроконтроллерах серии 1986BE9x работает только в режиме Master.

Интерфейс работает на 3 скоростях:
- нормальная: 100 Kbps (DIV=150 при HCLK=80МГц);
- быстрая: 400 Kbps (DIV=25 при HCLK=80МГц);
- очень быстрая: 1 Mbps (DIV=1 при HCLK=80МГц).

Скорость обмена данными блоком I2C рассчитывается по формуле:

$$Fscl = HCLK/(5*(DIV+1)).$$

26.1 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»).

26.2 Протокол I2C

Нормальная передача по интерфейсу I2C содержит 4 фазы:
- сигнал START;
- передача адреса;
- передача данных;
- сигнал STOP.

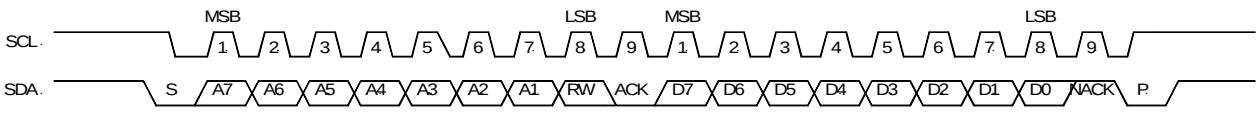


Рисунок 97. Передача по I2C

26.3 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.

26.4 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.

26.5 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.

26.6 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.

26.7 Описание регистров контроллера I2C

Таблица 324 – 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

26.7.1 MDR_I2C->PRL

Таблица 325 – Регистр PRL

Номер	31…8	7… 0
Доступ	U	R/W
Сброс	0	0
	-	**PR[7:0]**

Таблица 326 – 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

26.7.2 MDR_I2C->PRH

Таблица 327 – Регистр PRH

Номер	31…8	7… 0
Доступ	U	R/W
Сброс	0	0
	-	**PR[15:8]**

Таблица 328 – 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

26.7.3 MDR_I2C->CTR

Таблица 329 – 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 330 – 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

26.7.4 MDR_I2C->RXD

Таблица 331 – Регистр RXD

Номер	31…8	7… 0
Доступ	U	R/W
Сброс	0	0
	-	**RXD[7:0]**

Таблица 332 – 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

26.7.5 MDR_I2C->STA

Таблица 333 – 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 334 – 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

26.7.6 MDR_I2C->TXD

Таблица 335 – Регистр TXD

Номер	31…8	7… 0
Доступ	U	R/W
Сброс	0	0
	-	**TXD[7:0]**

Таблица 336 – 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

26.7.7 MDR_I2C->CMD

Таблица 337 – 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 338 – 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

27 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.

27.1 Основные характеристики модуля SSP

- может функционировать 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 98.

27.2 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.

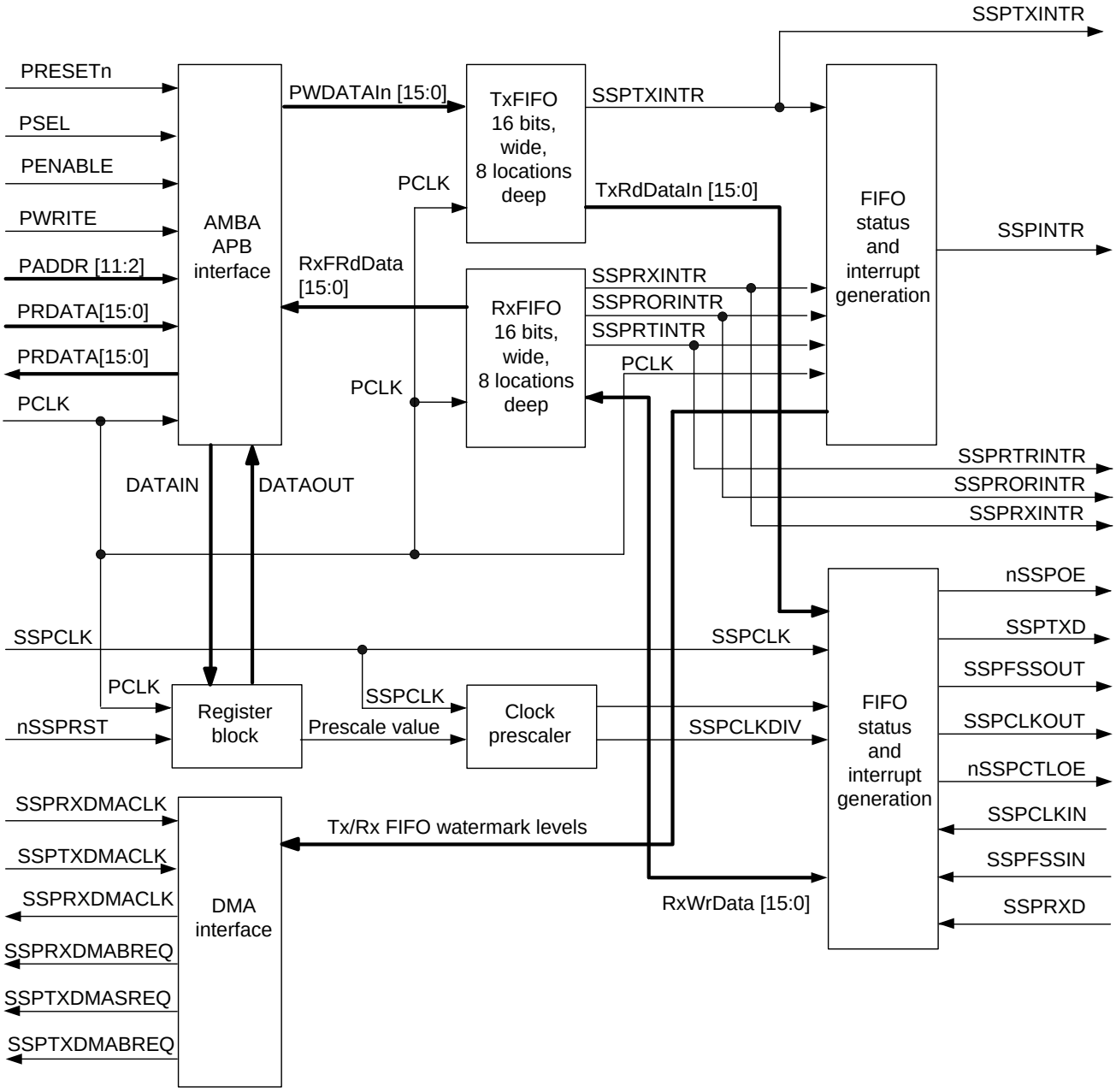


Рисунок 98. Структурная схема модуля SSP

27.3 Характеристики интерфейса SPI

Последовательный синхронный интерфейс SPI фирмы Motorola обеспечивает:
- полнодуплексный обмен данными по четырехпроводной линии;
- программное задание фазы и полярности тактового сигнала.

27.4 Характеристики интерфейса Microwire

Интерфейс Microwire фирмы National Semiconductor обеспечивает:
- полудуплексный обмен данными с использованием 8-битных управляющих последовательностей.

27.5 Характеристики интерфейса SSI

Интерфейс SSI фирмы Texas Instruments обеспечивает:
- полнодуплексный обмен данными по четырехпроводной линии;
- возможность перевода линии передачи данных в третье (высокоимпедансное) состояние.

27.6 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 в 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).

27.6.1 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.

27.6.2 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.

27.6.3 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.

27.6.4 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.

27.6.5 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.

27.6.6 Интерфейс прямого доступа к памяти

Модуль обеспечивает интерфейс с контроллером DMA согласно схеме взаимодействия приемопередатчика и контроллера DMA.

27.6.7 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.

27.6.8 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.

27.6.9 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].
```

27.6.10 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).

27.6.11 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).

27.6.12 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 программирования 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.

27.6.13 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. Можно 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 8-битная 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.

27.6.14 Формат синхронного обмена SSI фирмы Texas Instruments

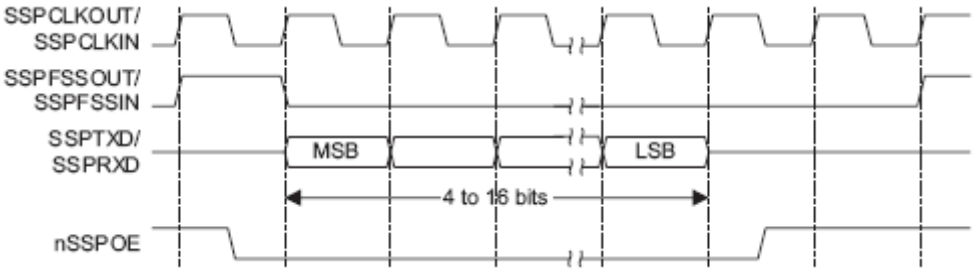


Рисунок 99. Формат синхронного обмена протокола SSI (единичный обмен)

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. При этом значение из буфера FIFO 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 99 – передача единичного информационного кадра Рисунок 100 – передача последовательности кадров.

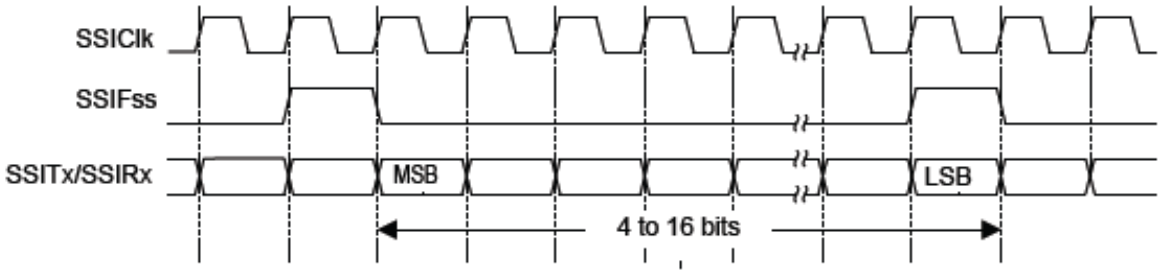


Рисунок 100. Формат синхронного обмена протокола SSI (непрерывный обмен)

27.6.15 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.

27.6.16 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 101 - одиночный обмен Рисунок 102 - непрерывный обмен.

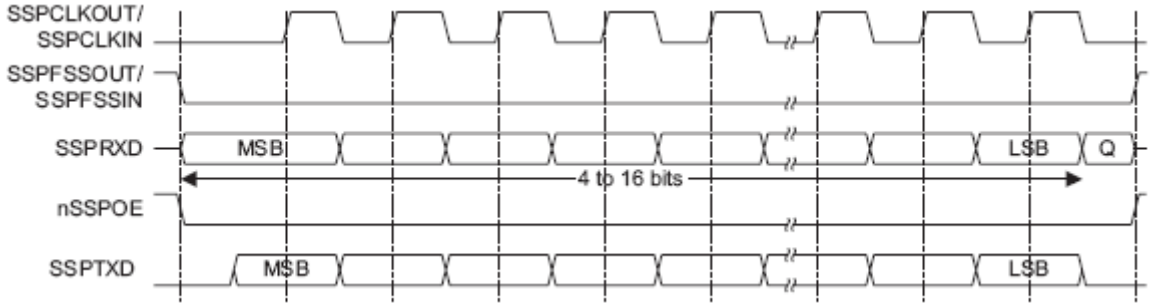


Рисунок 101. Формат синхронного обмена протокола SPI, SPO=0, SPH=0 (одиночный обмен)

Примечание – На рисунке буквой Q обозначен сигнал с неопределенным уровнем.

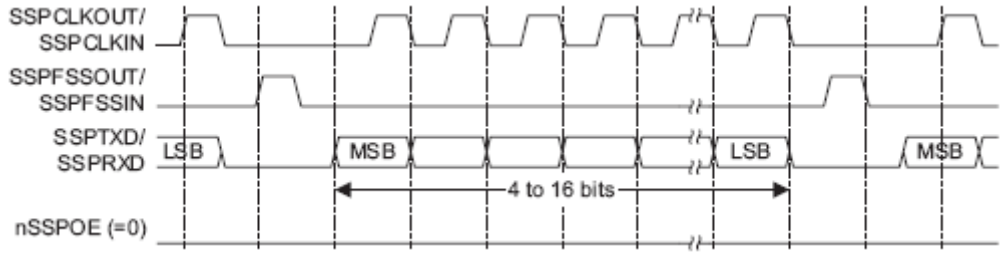


Рисунок 102. Формат синхронного обмена протокола SPI, SPO=0, SPH=0 (непрерывный обмен)

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.

27.6.17 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 103 - одиночный и непрерывный обмен.

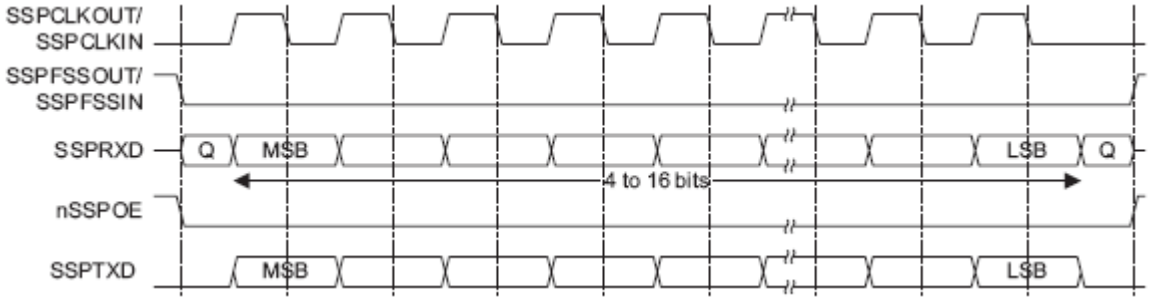


Рисунок 103. Формат синхронного обмена протокола SPI, SPO=0, SPH=1

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.

27.6.18 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 104 - одиночный обмен и Рисунок 105 - непрерывный обмен.

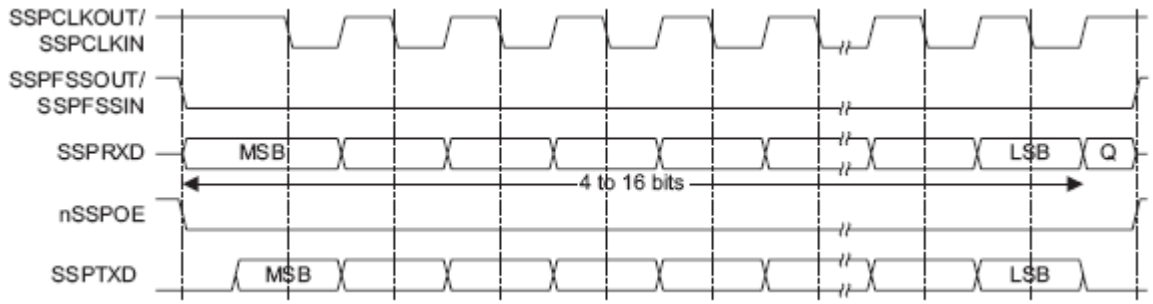


Рисунок 104. Формат синхронного обмена протокола SPI, SPO=1, SPH=0 (одиночный обмен)

Примечание – На рисунке буквой Q обозначен сигнал с неопределенным уровнем.

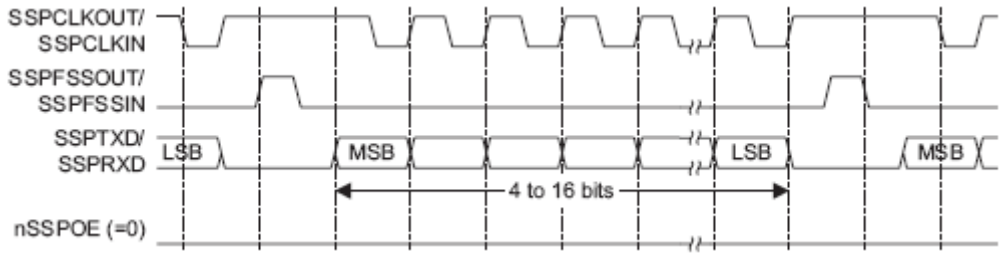


Рисунок 105. Формат синхронного обмена протокола SPI, SPO=1, SPH=0 (непрерывный обмен)

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.

27.6.19 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 106 - одиночный и непрерывный обмен.

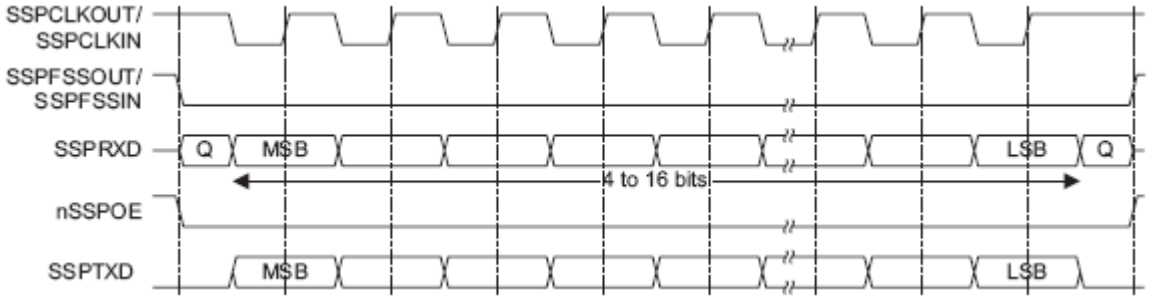


Рисунок 106. Формат синхронного обмена протокола SPI, SPO=1, SPH=1

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.

27.6.20 Формат синхронного обмена Microwire фирмы National Semiconductor

Временные диаграммы последовательного синхронного обмена в режиме Microwire показаны на Рисунок 107 - одиночный обмен и на Рисунок 108 - непрерывный обмен.

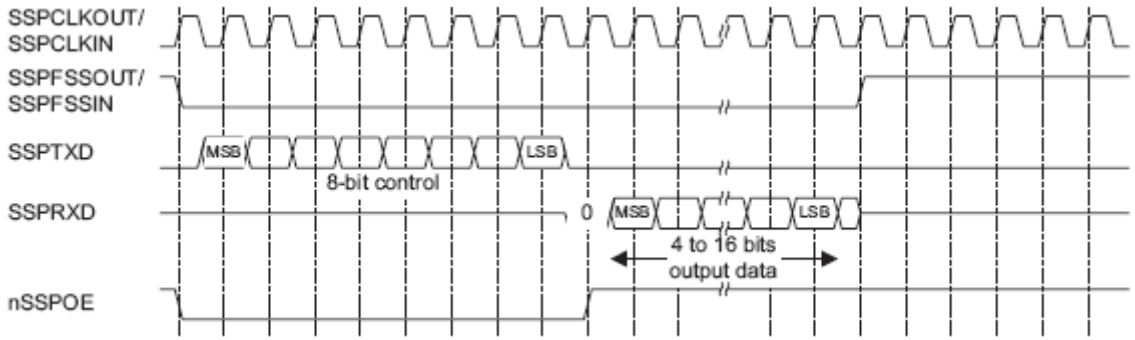


Рисунок 107. Формат синхронного обмена протокола Microwire (одиночный обмен)

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 8-битной 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.

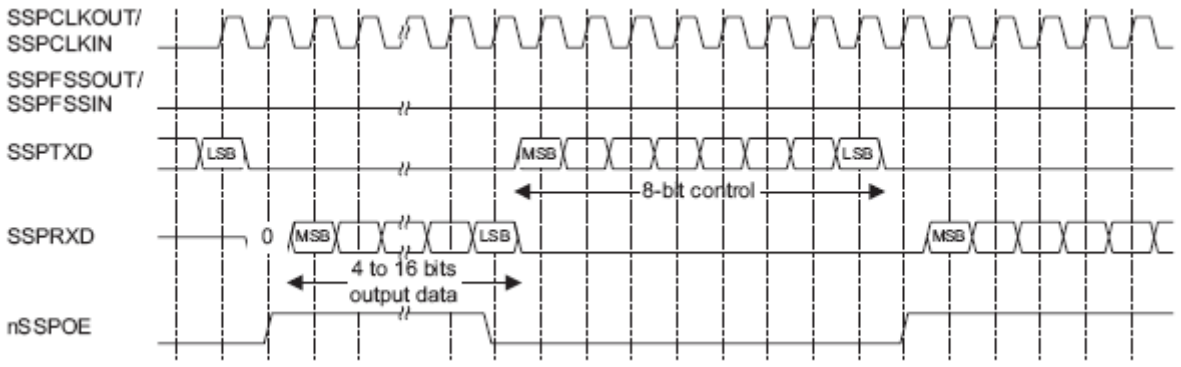


Рисунок 108. Формат синхронного обмена протокола Microwire (непрерывный обмен)

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 109. 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.

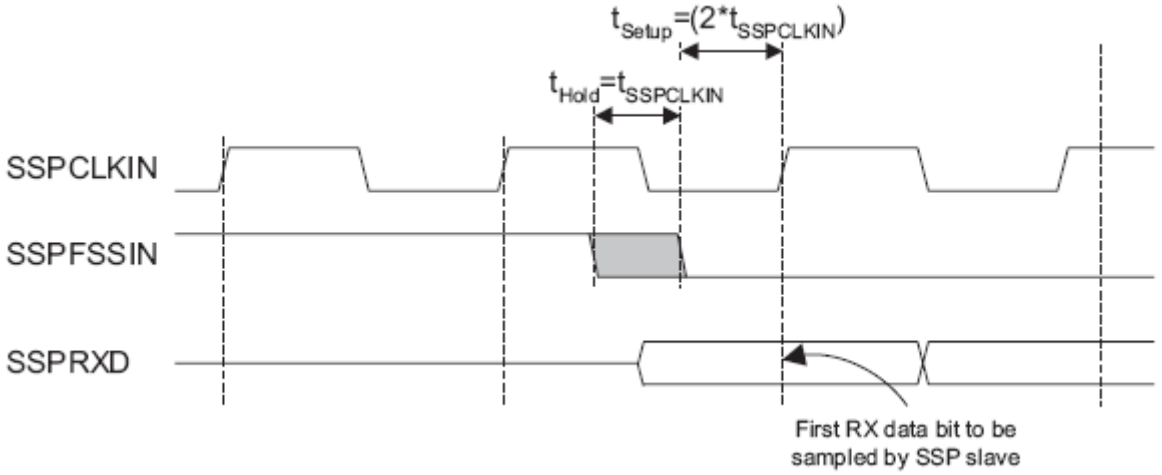


Рисунок 109. Формат Microwire, требования к времени установки и удержания сигнала

27.6.21 Примеры конфигурации модуля в ведущем и ведомом режимах

На рисунках Рисунок 110, Рисунок 111 и Рисунок 112 показаны варианты подключения модуля SSP к периферийным устройствам, работающим в ведущем или ведомом режиме.

Примечание – Модуль SSP не поддерживает динамическое изменение режима «ведущий – ведомый». Каждый приемопередатчик должен быть изначально сконфигурирован в одном из этих режимов.

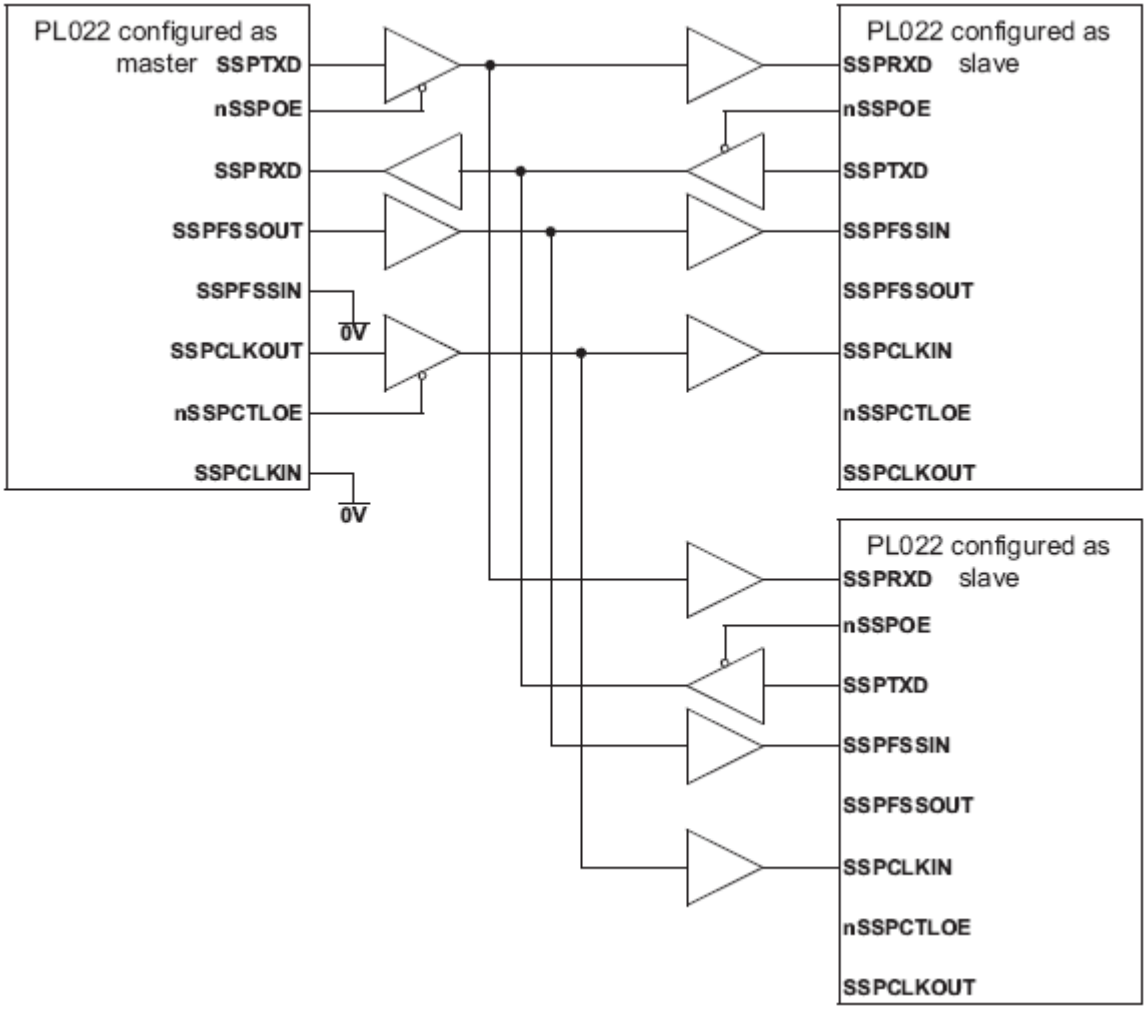


Рисунок 110. Ведущее устройство SSP подключено к двум ведомым

Рисунок 110 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.

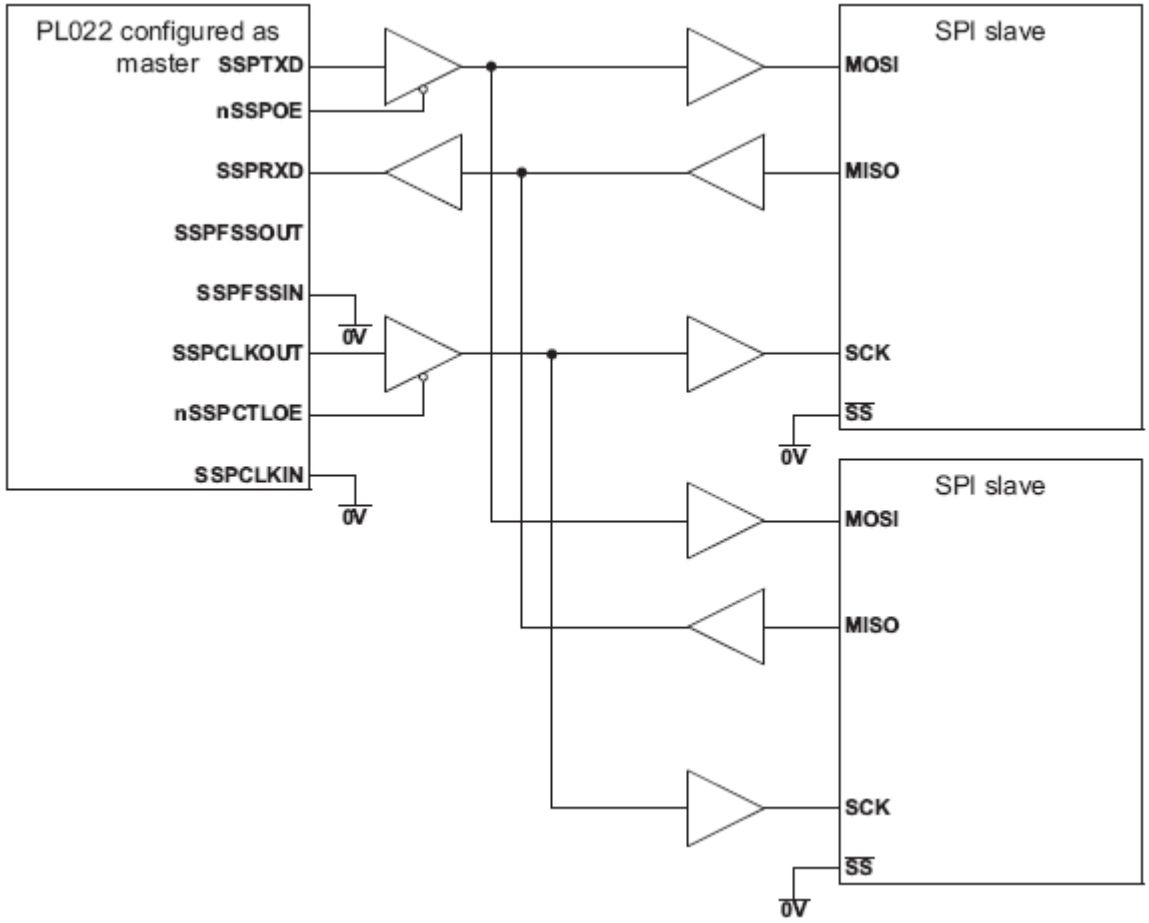


Рисунок 111. Ведущее устройство SSP подключено к двум ведомым, поддерживающим SPI

Рисунок 111 показывает подключение модуля SSP, сконфигурированного как ведущее устройство, к двум ведомым устройствам, поддерживающим протокол SPI фирмы Motorola. Внешние устройства сконфигурированы как ведомые путем установки в низкий логический уровень сигнала выбора ведомого устройства Slave Select (SS). Как и в предыдущем примере, ведущее устройство способно передавать данные в адрес ведомых циркулярно по линии SSP_TXD. Ответная передача данных на входную линию SSP_RXD ведущего устройства одновременно осуществляется только одним из ведомых по соответствующей линии MISO.

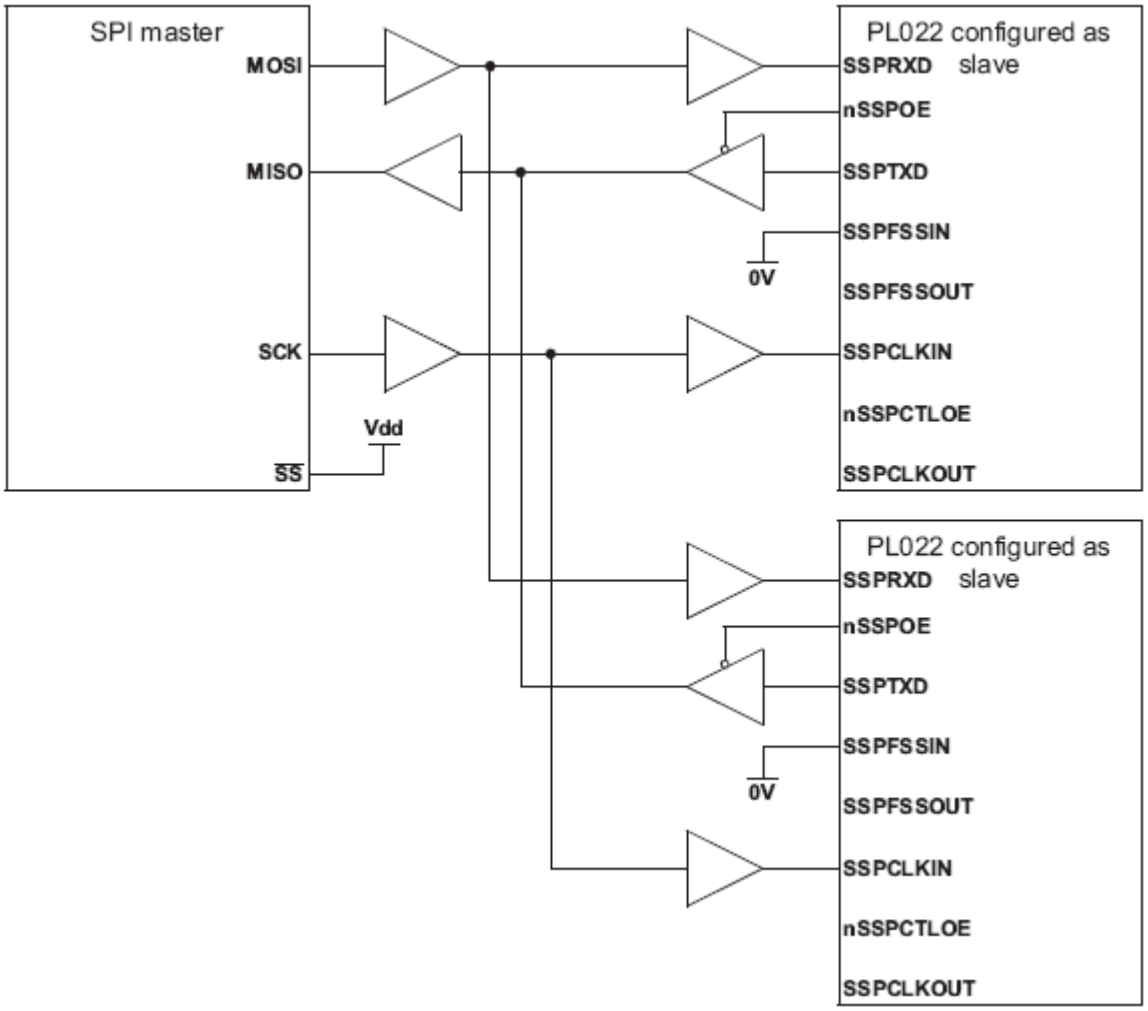


Рисунок 112. Ведущее устройство, протокол SPI, подключено к двум ведомым модулям SSP

Рисунок 112 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.

27.6.22 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 339 приведены значения порогов заполнения буферов приемника и передатчика, необходимых для срабатывания запросов блочного обмена DMABREQ.

Таблица 339 – 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 113 показывает временные диаграммы одноэлементного и блочного запросов DMA, в том числе действие сигнала DMACLR. Все сигналы должны быть синхронизированы с PCLK.

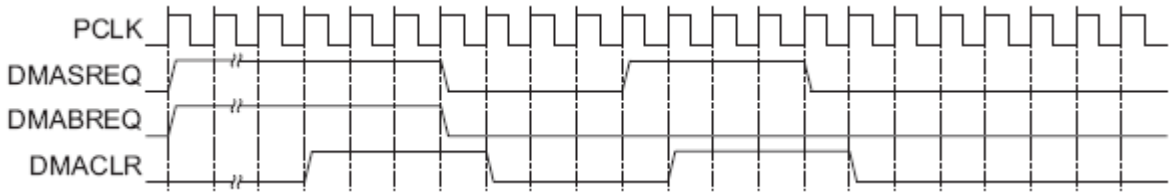


Рисунок 113. Временные диаграммы обмена в режиме DMA

27.7 Программное управление модулем

27.7.1 Общая информация

В микроконтроллере реализовано два модуля SSP, базовые адреса каждого модуля указаны далее в Таблица 340. Смещение каждого регистра относительно базового адреса постоянно. Следующие адреса являются резервными и не должны использоваться в нормальном режиме функционирования:

- адреса со смещениями в диапазоне +0x028 ... +0x07C и +0xFD0 ... +0xFDC зарезервированы для перспективных расширений возможностей модуля;
- адреса со смещениями в диапазоне +0x080 ... +0x088 зарезервированы для тестирования.

27.7.2 Описание регистров контроллера SSP

Данные о регистрах модуля SSP приведены в Таблица 340.

Таблица 340 – 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.

27.7.3 MDR_SSPx->CR0

Регистр управления 0

Регистр CR0 содержит пять битовых полей, предназначенных для управления блоками модуля SSP. Назначение разрядов регистра представлено в Таблица 341.

Таблица 341 – 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

27.7.4 MDR_SSPx->CR1

Регистр управления 1

Регистр CR1 содержит четыре битовых поля, предназначенных для управления блоками модуля SSP. Назначение разрядов регистра представлено в Таблица 342.

Таблица 342 – 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 разрешена. 1 – управление линией SSP_TXD в ведомом режиме запрещена
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 со входом регистра сдвига приемника

27.7.5 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 следующей Таблица 343.

Таблица 343 – 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

27.7.6 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 Таблица 344.

Таблица 344 – 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

27.7.7 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 345 отображает назначение бит регистра SSPSR.

Таблица 345 – 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

27.7.8 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 346 – 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 более буфера FIFO приемника. 1 – не маскирована. 0 – маскирована
1	RTIM	Маска прерывания по таймауту приемника (буфер FIFO приемника не пуст и не было попыток его чтения в течение времени таймаута). 1 – не маскирована. 0 – маскирована
0	RORIM	Маска прерывания по переполнению буфера приемника. 1 – не маскирована. 0 – маскирована

27.7.9 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 347 отображает назначение бит в регистре RIS.

Таблица 347 – 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

27.7.10 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 348 – 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

27.7.11 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 349 – 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

27.7.12 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 350 – 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

27.8 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.

SSPRXINTR

Прерывание по заполнению буфера FIFO приемника формируется в случае, если буфер приемника содержит четыре или более несчитанных слов данных.

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.

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.

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.

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.

28 Контроллер MDR_UART

Модуль универсального асинхронного приемопередатчика (UART – Universal 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).

28.1 Основные сведения

Основные сведения о модуле представлены в следующих разделах:
- основные характеристики;
- программируемые параметры;
- отличия от приемопередатчика 16C650.

28.1.1 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 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.

28.1.2 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 – 16 элементов 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.

28.1.3 Отличия от контроллера UART 16C650

Контроллер отличается от промышленного стандарта асинхронного приемопередатчика 16C650 следующими характеристиками:
– пороги срабатывания прерывания по заполнению буфера FIFO приемника – 1/8, 1/4, 1/2, 3/4 и 7/8;
– пороги срабатывания прерывания по заполнению буфера FIFO передатчика – 1/8, 1/4, 1/2, 3/4 и 7/8;
– отличается распределение адресов внутренних регистров и назначение бит в регистрах;
– недоступны изменения сигналов состояния модема.

Следующие возможности контроллера 16C650 не поддерживаются:
– полуторная длительность стопового бита (поддерживается только 1 или 2 стоповых бита);
– независимое задание тактовой частоты приемника и передатчика.

28.2 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 в 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.

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) и «индикатор вызова» (Ring Indicator, RI), а также выходные линии: «запрос на передачу» (Request to Send, RTS) и «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.

28.3 Описание функционирования блока UART

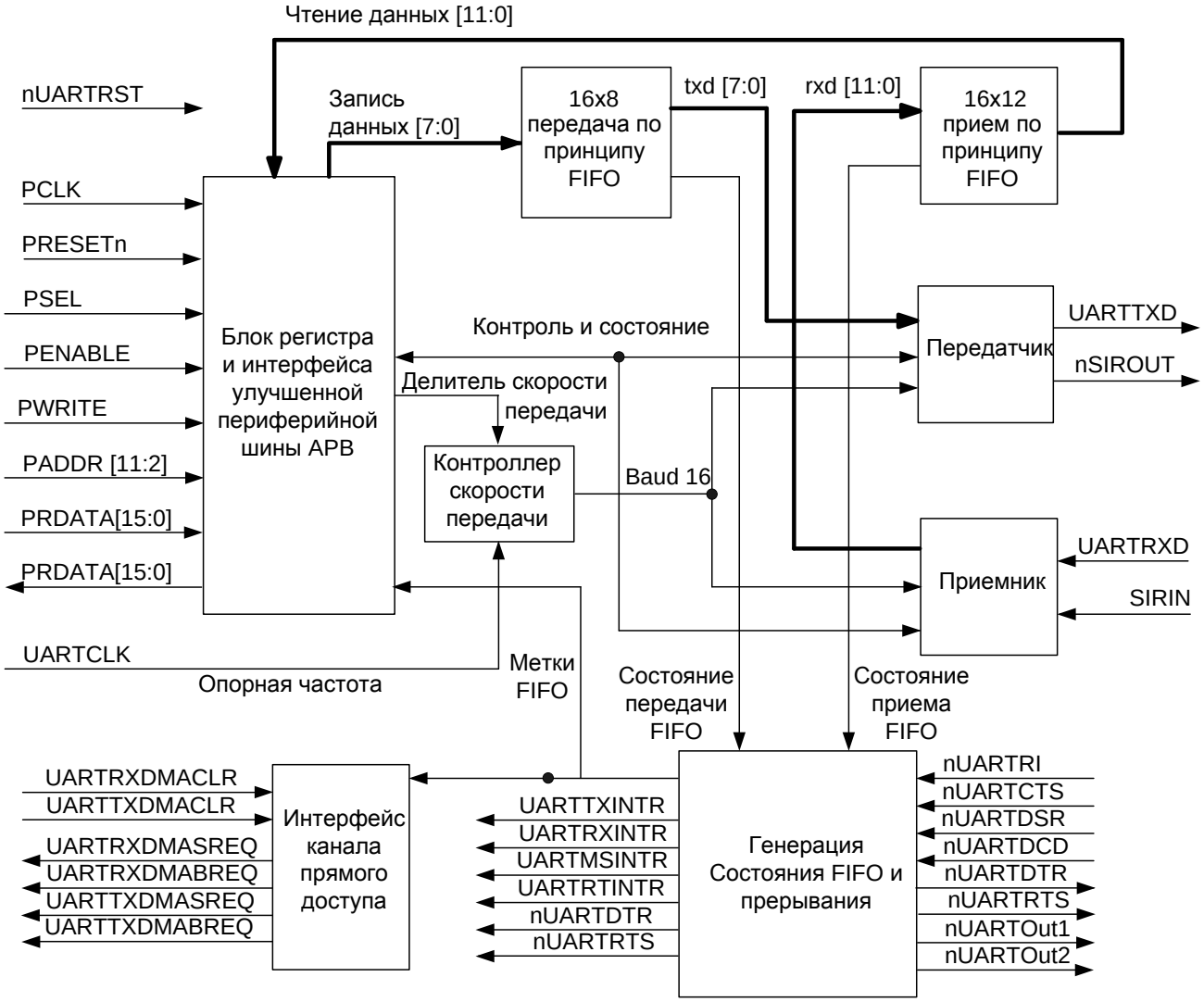


Рисунок 114. Блок-схема универсального асинхронного приёмопередатчика (UART)

28.3.1 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.

28.3.2 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.

28.3.3 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.

28.3.4 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.

28.3.5 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.

28.3.6 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.

28.3.7 Интерфейс прямого доступа к памяти

Модуль обеспечивает интерфейс с контроллером DMA согласно схеме взаимодействия приемопередатчика и контроллера DMA.

28.3.8 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.

28.4 Описание функционирования ИК кодека IrDA SIR

Структурная схема кодека представлена ниже (Рисунок 115).

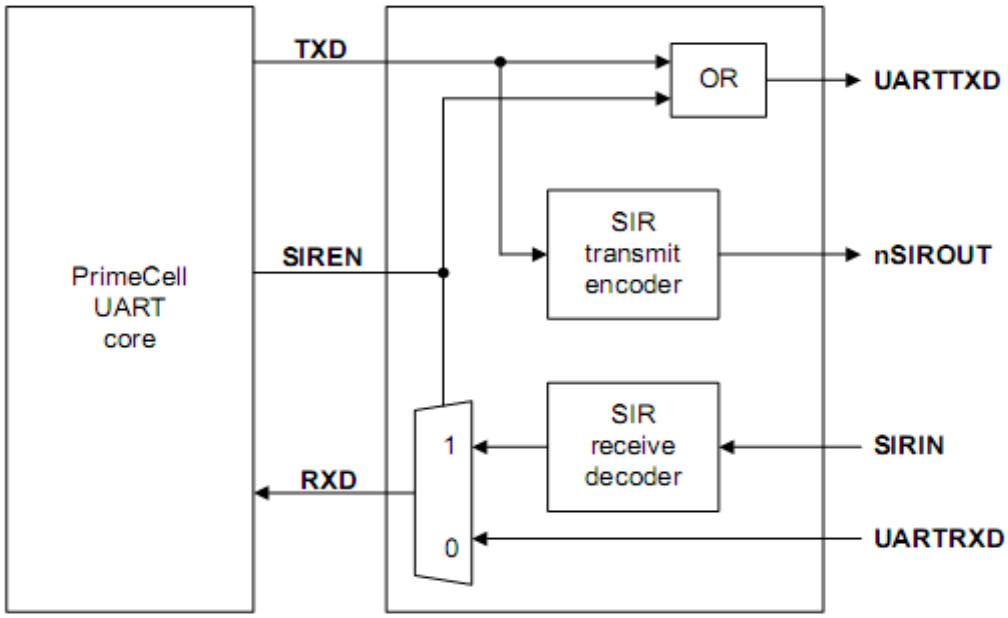


Рисунок 115. Структурная схема кодека IrDA

28.4.1 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.

28.4.2 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.

28.5 Описание работы UART

28.5.1 Сброс модуля

Приемопередатчик и кодек могут быть сброшены общим сигналом сброса процессора. Значения регистров после сброса описаны в разделе «Программное управление модулем».

28.5.2 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.

28.5.3 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.

28.5.4 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.

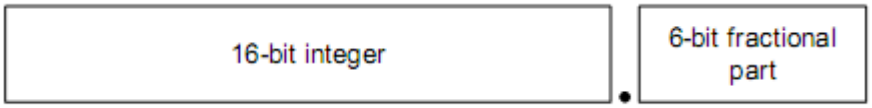


Рисунок 116. Коэффициент деления

Шестибитное значение, записываемое в регистр UARTFBRD, вычисляется путем выделения дробной части требуемого коэффициента деления, умножения ее на 64 (то есть на 2^n, где n – ширина регистра UARTFBRD) и округления до ближайшего целого числа:

$$M = integer\ (BRD\_F * 2^n + 0.5),$$

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.

28.5.5 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. таблицу 351).

28.5.6 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 351 описывает назначение всех бит слова данных в FIFO-буфере приемника.

Таблица 351 – 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

28.5.7 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.

28.5.8 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.

Проверка по шлейфу

Проверка по шлейфу (замыкание выхода передатчика на вход приемника) выполняется путем установки в 1 бита LBE в регистре управления контроллером UARTCR.

28.5.9 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.
```

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.

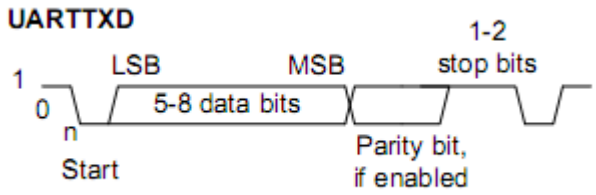


Рисунок 117. Кадр передачи данных

28.5.10 Модуляция данных IrDA

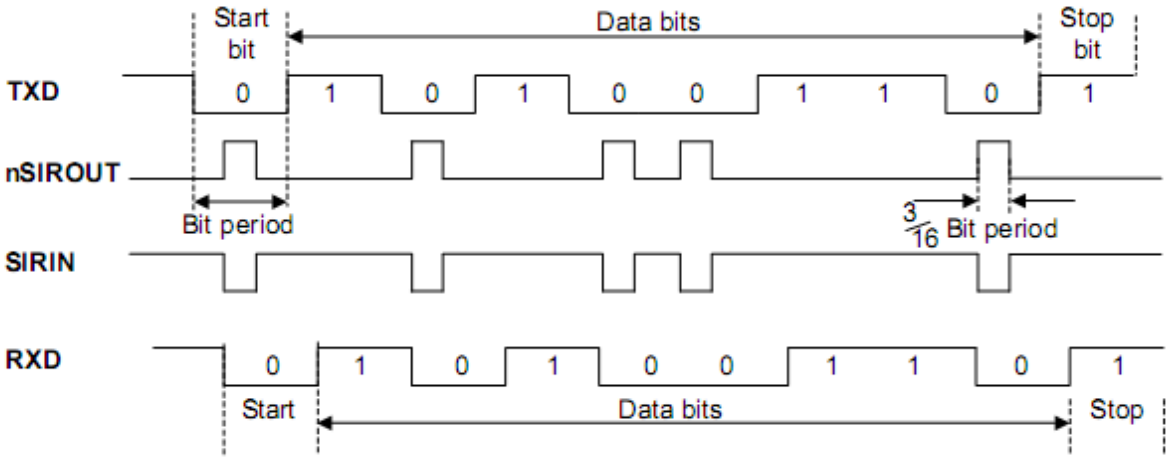


Рисунок 118. Модуляция данных IrDA

28.6 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 114).

Назначение сигналов в режимах DTE и DCE представлено в таблице ниже.

Таблица 352 – 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	-
nUARTRTS	Запрос передачи данных	Готов к передаче данных
nUARTDTR	Приемник данных готов	Источник данных готов
nUARTOUT1	-	Обнаружен информационный сигнал
nUARTOUT2	-	Индикатор вызова

28.6.1 Аппаратное управление потоком данных

Программно активизируемый режим аппаратного управления потоком данных позволяет контролировать (приостанавливать и возобновлять) информационный обмен с помощью сигналов nUARTRTS и nUARTCTS. Иллюстрация взаимодействия двух устройств последовательной связи с аппаратным управлением потоком данных представлена на рисунке ниже (Рисунок 119).

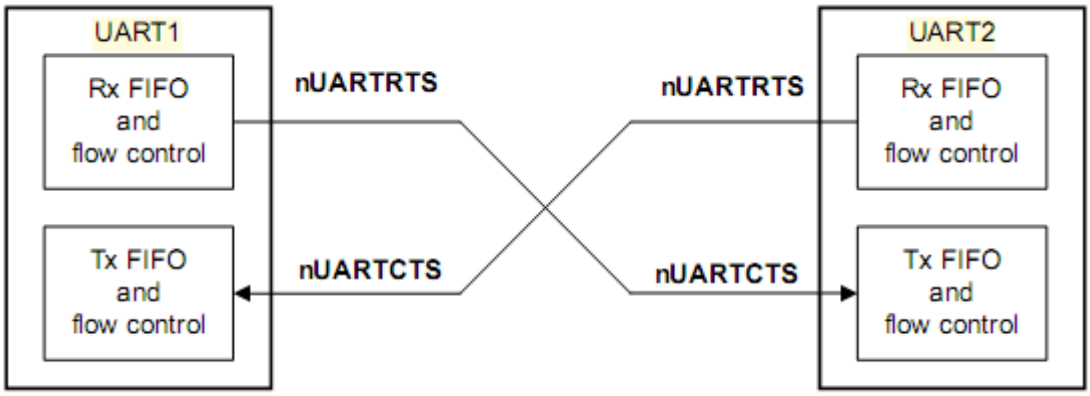


Рисунок 119. Взаимодействие двух устройств последовательной связи с аппаратным управлением потоком данных

Если разрешено управление потоком данных по сигналу RTS, линия nUARTRTS переводится в активное состояние 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 353 показывает необходимые установки для различных режимов управления потоком данных.

Таблица 353 – 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.

28.6.2 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.

28.6.3 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.

28.7 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 одно-элементном 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 354 показывает значения параметров срабатывания запросов блочного обмена UARTRXDMABREQ и UARTTXDMABREQ в зависимости от порога заполнения буфера.

Таблица 354 – 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	14	2
1/4	12	4
1/2	8	8
3/4	4	12
7/8	2	14

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.

На Рисунок 120 показаны временные диаграммы одноэлементного и блочного запросов DMA, в том числе действие сигнала DMACLR. Все сигналы должны быть синхронизированы с CPU_CLK. В интересах ясности изложения предполагается, что синхронизация сигналов запроса DMA в контроллере DMA не производится.

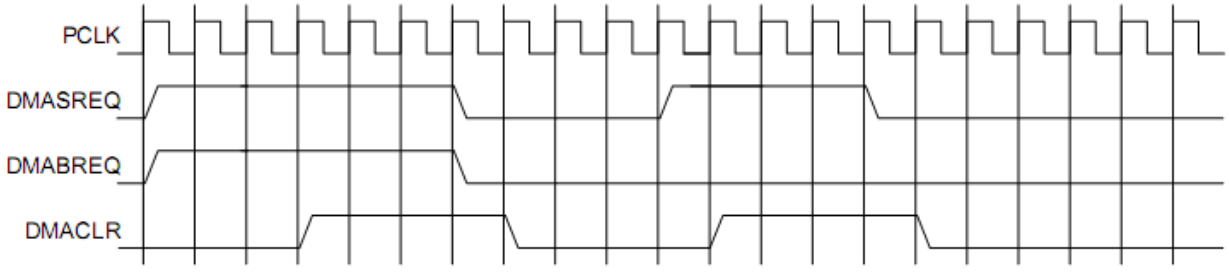


Рисунок 120. Временные диаграммы одноэлементного и блочного запросов DMA

28.8 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.

28.8.1 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.

28.8.2 UARTRXINTR

Состояние прерывания от приемника может измениться в случае возникновения одного из следующих событий:

- буфер FIFO разрешен и его заполнение достигло заданного порогового значения. В этом случае линия прерывания переходит в высокое состояние. Сигнал прерывания переходит в низкое состояние после чтения данных из буфера приемника до тех пор, пока его заполнение не станет меньше порога, либо после сброса прерывания;
- буфер FIFO запрещен (имеет размер один символ), принят один символ данных. При этом линия прерывания переходит в высокое состояние. Сигнал прерывания переходит в низкое состояние после чтения одного байта данных, либо после сброса прерывания.

28.8.3 UARTTXINTR

Состояние прерывания от передатчика может измениться в случае возникновения одного из следующих событий:

- буфер FIFO разрешен и его заполнение меньше или равно заданному пороговому значению. В этом случае линия прерывания переходит в высокое состояние. Сигнал прерывания переходит в низкое состояние после записи данных в буфера передатчика до тех пор, пока его заполнение не станет больше порога, либо после сброса прерывания;
- буфер FIFO запрещен (имеет размер один символ), данные в буферном регистре передатчика отсутствуют. При этом линия прерывания переходит в высокое состояние. Сигнал прерывания переходит в низкое состояние после записи одного байта данных, либо после сброса прерывания.

Для занесения данных в буфер FIFO передатчика необходимо записать данные в буфер либо перед разрешением работы приемопередатчика и прерываний, либо после разрешения работы приемопередатчика и прерываний.

Примечание – Прерывание передатчика работает по фронту, а не по уровню сигнала. В случае, если модуль и прерывания от него разрешены до осуществления записи данных в буфер FIFO передатчика, прерывание не формируется. Прерывание возникает только при опустошении буфера FIFO.

28.8.4 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.

28.8.5 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.

28.8.6 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.

28.9 Программное управление модулем

28.9.1 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 355 определяет режим доступа к регистру в соответствии с обозначениями:
  - RW – чтение и запись;
  - RO – только чтение;
  - WO – только запись.

28.9.2 Обобщенные данные о регистрах устройства

Данные о регистрах модуля универсального асинхронного приемопередатчика приведены в таблице ниже.

Таблица 355 – 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					Зарезервировано
0x018	FR	RO	0b-10010---	9	**MDR_UARTx->FR** Регистр флагов
0x01C					Зарезервировано
0x020	ILPR	RW	0x00	8	**MDR_UARTx**->ILPR Регистр управления ИК обменом в режиме пониженного 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

28.9.3 MDR_UARTx->DR

Регистр данных

В ходе передаче данных:

Если буфер FIFO передатчика разрешен, то слово данных, записанное в рассматриваемый регистр, направляется в буфер FIFO передатчика.

В противном случае, записанное слово фиксируется в буферный регистр передатчика (последний элемент буфера FIFO).

Операция записи в регистр инициирует передачу данных. Слово данных предваряется стартовым битом, дополняется битом контроля четности (если режим контроля четности включен) и стоповым битом. Сформированное слово отправляется в линию передачи данных.

В ходе приема данных:

Если буфер FIFO приемника разрешен, байт данных и четыре бита состояния (разрыв, ошибка формирования кадра, четность, переполнение) сохраняются в 12-битном буфере.

В противном случае байт данных и биты состояния записываются в буферный регистр (последний элемент буфера FIFO).

Полученные из линии связи байты данных 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 356).

Таблица 356 – 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.

28.9.4 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 357 показывает назначение бит регистра UARTRSR/UARTECR.

Таблица 357 – 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.

28.9.5 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 358 – 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

28.9.6 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 359 – Регистр UARTILPR.

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 359 – 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.

28.9.7 MDR_UARTx->IBRD

Регистр целой части делителя скорости передачи данных

Назначение бит регистра UARTBIRD показано ниже.

Таблица 360 – 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

28.9.8 MDR_UARTx->FBRD

Регистр дробной части делителя скорости передачи данных,

Таблица 361 показывает назначение бит регистра представлено в таблице ниже.

Таблица 361 – 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:

- изменения содержимого регистров UARTIBRD и UARTFBRD вступают в силу только после завершения передачи и приема текущего символа данных;
- минимальный допустимый коэффициент деления – 1, максимальный 65535 (2^{16} - 1). Таким образом, значение UARTIBRD, равное 0, является недопустимым, при этом значение регистра UARTFBRD игнорируется;
- 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 = $(4*10^6)/(16*230400) = 1.085$.

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 = $(4*10^6)/(16*1.078)$ = 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 362) 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 362 – 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 363 – 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

28.9.9 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 364 показывает назначение разрядов регистра UARTLCR_H.

Таблица 364 – 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:

Изменение значений трех регистров можно осуществить корректно двумя способами:

– запись UARTIBRD, запись UARTFBRD, запись UARTLCR_H;

– запись UARTFBRD, запись UARTIBRD, запись UARTLCR_H.

Для того, чтобы изменить значение лишь одного из регистров (UARTIBRD или UARTFBRD) необходимо выполнить следующий шаг:

– запись UARTIBRD (или UARTFBRD), запись UARTLCR_H.

Таблица 365 показывает таблицу истинности для бит управления контролем четности PEN, EPS и SPS регистра управления линией UARTLCR_H.

Таблица 365 – 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:

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.

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.

28.9.10 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 таблице ниже.

Таблица 366 – 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

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

28.9.11 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 367 – 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 Также стоит помнить, что в случае, если сдвиговый регистр передатчика пуст, то слово, записанное в FIFO, будет сразу же переписано в сдвиговый регистр. Следовательно, для генерирования события прерывания от передатчика блока UART необходимо произвести запись в FIFO такого количества слов, которое превысит установленный порог хотя бы на одно слово с учетом описанного случая.

28.9.12 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 таблице ниже.

Таблица 368 – 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

28.9.13 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 таблице ниже.

Таблица 369 – 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 Бит RTRIS может быть установлен только при установленной маске прерывания по таймауту приема данных UARTRTINTR в регистре UARTIMSC. Это вызвано тем, что сигнал маски прерывания по таймауту используется в качестве разрешения перехода в режим пониженного энергопотребления. Чтение состояния прерывания по таймауту из регистров UARTMIS и UARTRIS приводит к одинаковым результатам.
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

28.9.14 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 370 – 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

28.9.15 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 ниже.

Таблица 371 – 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

28.9.16 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 ниже.

Таблица 372 – 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

29 Контроллер прямого доступа в память MDR_DMA

29.1 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.

29.2 Термины и определения

При описании контроллера используются следующие термины:

Таблица 373 – 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 10
Исполнение с изменением конфигурации	Режим работы для выбранного канала, при котором контроллер получает запрос от периферии и выполняет 4 DMA передачи, используя первичную структуру управляющих данных, которые настраивают альтернативную структуру управляющих данных. После чего контроллер начинает цикл DMA, используя альтернативную структуру данных. После того, как цикл закончится 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

29.3 Функциональное описание

На рисунке ниже (Рисунок 121) показана упрощенная структурная схема контроллера.

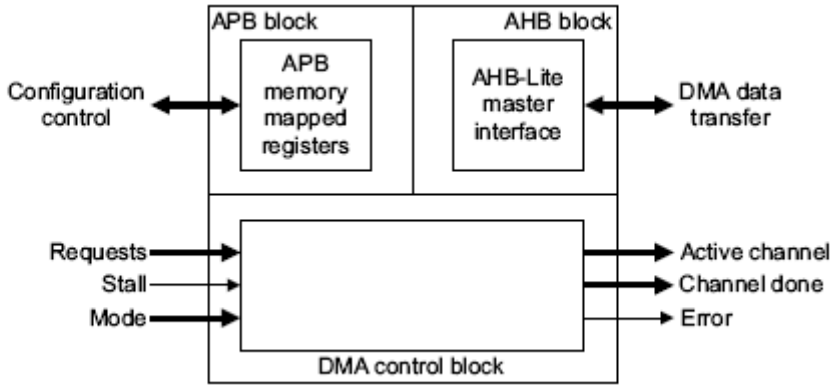


Рисунок 121. Структурная схема контроллера

Контроллер состоит из следующих основных функциональных блоков:

- блок, подключенный к шине APB;
- блок, подключенный к шине AHB;
- управляющий блок DMA.

29.3.1 Распределение каналов DMA

Таблица 374 – Распределение каналов DMA

Номер канала	Источник sreg	Источник reg	Тип	Описание
0	UART1 TX	UART1 TX		Запрос DMA от UART1 по передаче
1	UART1 RX	UART1 RX		Запрос DMA от UART1 по приему
2	UART2 TX	UART2 TX		Запрос DMA от UART2 по передаче
3	UART2 RX	UART2 RX		Запрос DMA от UART2 по приему
4	SSP1 TX	SSP1 TX		Запрос DMA от SSP1 по передаче
5	SSP1 RX	SSP1 RX		Запрос DMA от SSP1 по приему
6	SSP2 TX	SSP2 TX		Запрос DMA от SSP2 по передаче
7	SSP2 RX	SSP2 RX		Запрос DMA от SSP2 по приему
8	ADC0_EC	-		Запрос DMA от АЦП1
9	ADC1_EC	-		Запрос DMA от АЦП2
10	TIMER1	-		Запрос DMA от Timer1
11	TIMER2	-		Запрос DMA от Timer2
12	TIMER3	-		Запрос DMA от Timer3
13	-	-		Программный
14	-	-		Программный
15	-	-		Программный
16	-	-		Программный
17	-	-		Программный
18	-	-		Программный
19	-	-		Программный
20	-	-		Программный
21	-	-		Программный
22	-	-		Программный

Номер канала	Источник sreg	Источник reg	Тип	Описание
23	-	-		Программный
24	-	-		Программный
25	-	-		Программный
26	-	-		Программный
27	-	-		Программный
28	-	-		Программный
29	-	-		Программный
30	-	-		Программный
31	-	-		Программный

29.3.2 Блок, подключенный к шине APB

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

29.3.3 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.

29.3.4 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.

29.3.5 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.

29.3.6 Разрядность передач данных

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

Таблица 375 – 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.

29.3.7 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 376 перечисляет значения комбинаций шины HPROT.

Таблица 376 – 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.

29.3.8 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 377 – 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»).

29.4 Управление DMA

29.4.1 Правила обмена данными

Контроллер использует правила обмена данными, перечисленные далее в Таблица 378, 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 378 – 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

Правило	Описание
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

Правило	Описание
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 таблице 379.

Таблица 379 – 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

29.4.2 Диаграммы работы контроллера DMA

Данный раздел описывает примеры функционирования контроллера с использованием правил обмена данными, представленных ранее (Таблица 378):

- импульсный запрос на обработку;
- запрос по уровню на обработку;
- флаги завершения;
- флаги ожидания запроса на обработку.

Примечание – Все диаграммы, показанные далее в этом подразделе на рисунках ниже (Рисунок 122 – Рисунок 126), подразумевают следующее:

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

29.4.2.1 Импульсный запрос на обработку

Рисунок 122 показывает временную диаграмму работы контроллера DMA при получении импульсного запроса от периферии.

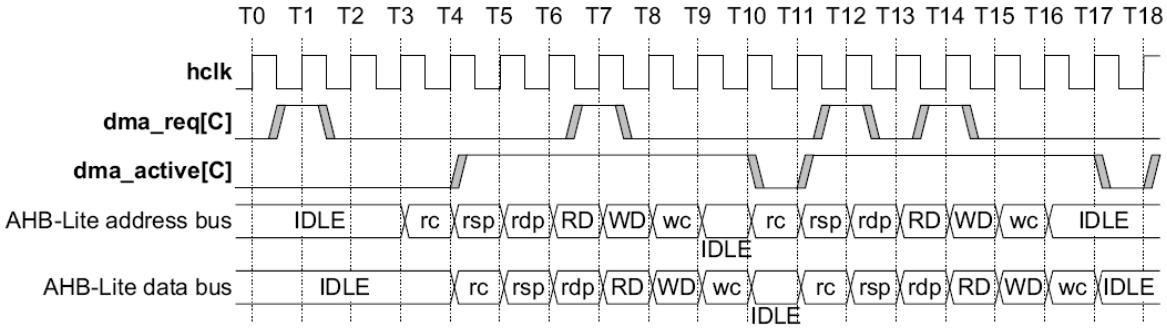


Рисунок 122. Диаграмма работы при получении импульсного запроса

Пояснения к диаграмме (Рисунок 122) приведены ниже.

Таблица 380 – 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] как минимум на 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)

29.4.2.2 Запрос на обработку по уровню

Рисунок 123 показывает временную диаграмму работы контроллера DMA при получении от периферии запроса на обработку по уровню.

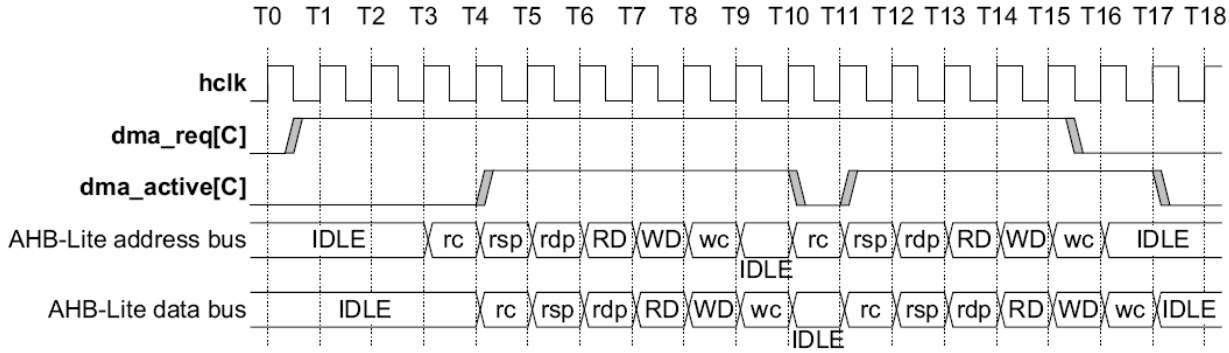


Рисунок 123. Диаграмма работы при получении запроса на обработку по уровню.

Пояснения к диаграмме на Рисунок 123 даны в Таблица 381.

Таблица 381 – Пояснения к диаграмме работы при получении запроса на обработку по уровню

T1	Контроллер обнаружил запрос на обработку по каналу C (Таблица 378, 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.

29.4.2.3 Флаги завершения.

Рисунок 124 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.

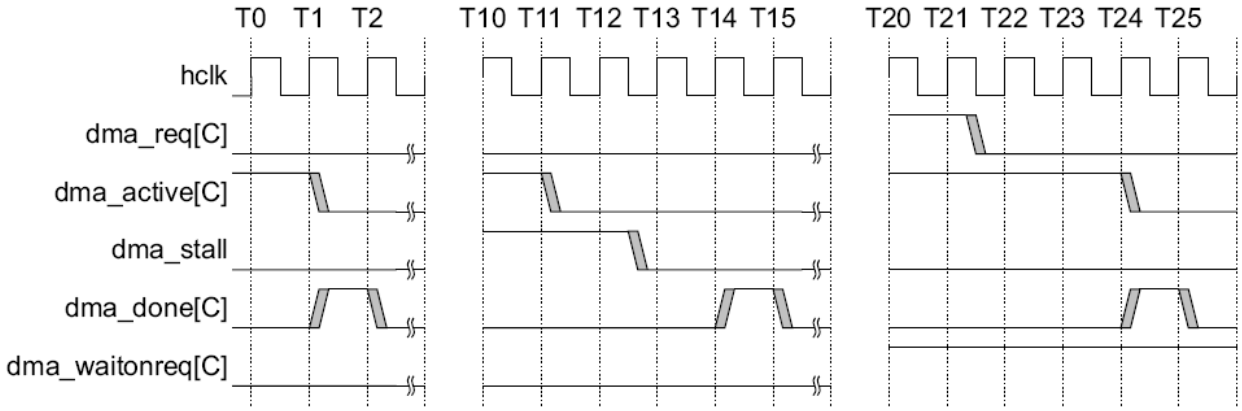


Рисунок 124. Диаграммы функционирования dma_done

Пояснения к диаграмме на Рисунок 124, такты от T0 до T2, приведены в Таблица 382.

Таблица 382 – Пояснения функционирования dma_done, такты от T0 до T2

T1	Контроллер сбрасывает сигнал dma_active[C], что указывает на окончание передачи DMA (см. Таблица 378, 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 на Рисунок 124, такты от T10 до T15, приведены в Таблица 383.

Таблица 383 – 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 на Рисунок 124, такты от T20 до T25, приведены в таблице ниже.

Таблица 384 – 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)

29.4.2.4 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.

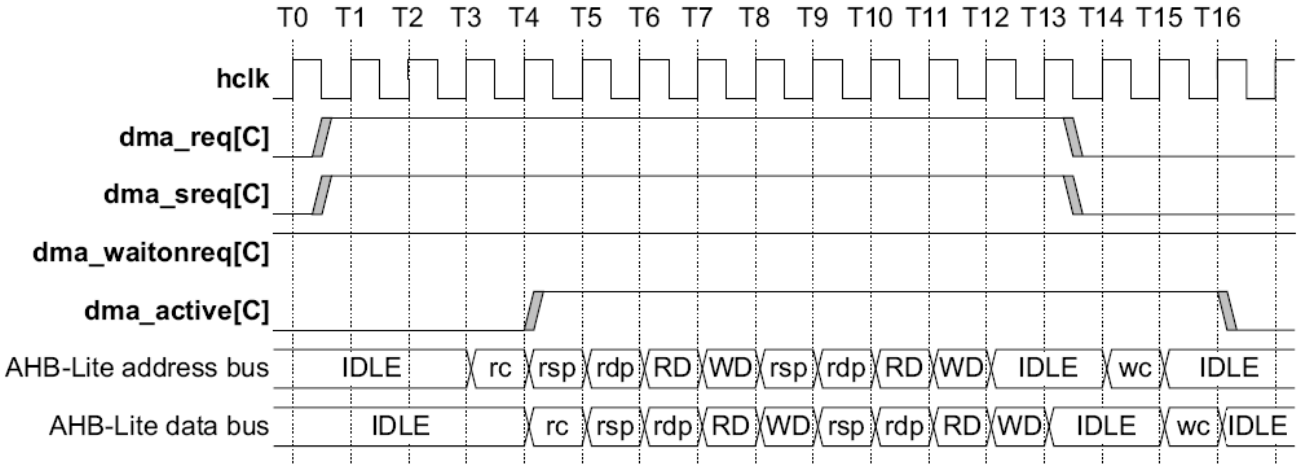


Рисунок 125. Диаграмма работы контроллера DMA при использовании dma_waitonreq

Пояснения к диаграмме (Рисунок 125) приведены в Таблица 385.

Таблица 385 – 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 126 показывает работу контроллера DMA при установке dma_waitonreq в 1 и выполнении одиночной DMA передачи.

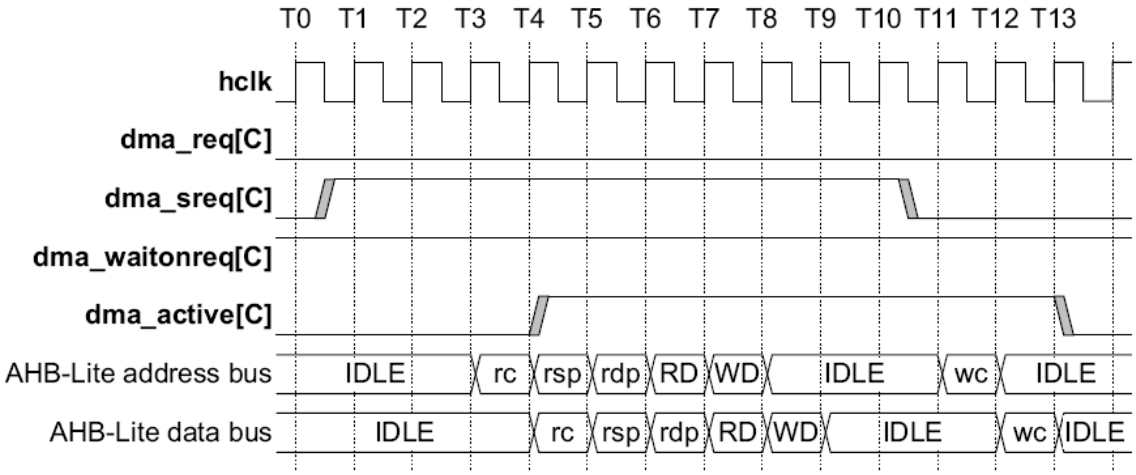


Рисунок 126. Работа DMA при использовании dma_waitonreq совместно с dma_sreq

Пояснения к диаграмме на Рисунок 126 приведены в Таблица 386.

Таблица 386 – 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)

29.4.3 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 387 показывает возможную периодичность арбитража.

Таблица 387 – 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».

29.4.4 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 388** показывает уровень приоритета каналов DMA в порядке его уменьшения.

Таблица 388 – Уровень приоритета каналов DMA

Уровень приоритета в порядке его уменьшения	**Номер канала**	**Уровень приоритета установленный битом chnl_priority_set**
Наивысший уровень приоритета	0	Высокий
-	1	Высокий
-	2	Высокий
…….	…….	…….
-	30	Высокий
-	31	Высокий
-	0	По умолчанию (низкий)
-	1	По умолчанию (низкий)
-	2	По умолчанию (низкий)
…….	…….	…….
-	30	По умолчанию (низкий)
Низший уровень приоритета	31	По умолчанию (низкий)

После окончания цикла DMA контроллер выбирает следующий для обслуживания канал из всех включенных каналов DMA. Рисунок 127 иллюстрирует процесс выбора следующего канала для обслуживания.

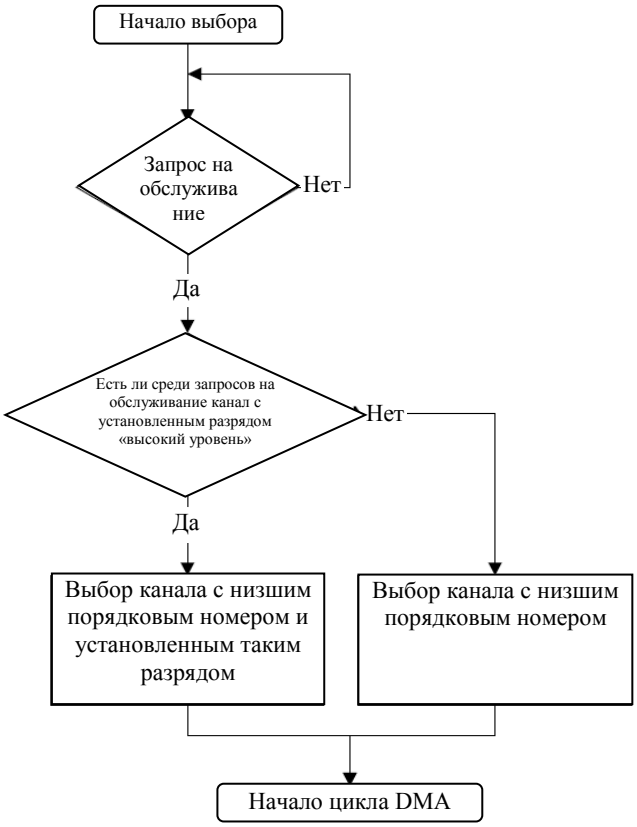


Рисунок 127. Алгоритм выбора следующего канала для обслуживания

29.4.5 Типы циклов DMA

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

Таблица 389 – 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 2^R передач DMA. Если установить длинный период арбитража на низкоприоритетном канале, то это заблокирует все запросы на обработку от других каналов до тех пор, пока не будут выполнены 2^R передач DMA по данному каналу. Поэтому, устанавливая значение R, необходимо учитывать, что это может привести к повышенному времени отклика на запрос на обработку от высокоприоритетных каналов.

Данный раздел описывает следующие типы циклов DMA:
- недействительный;
- основной;
- авто-запрос;
- «пинг-понг»;
- работа с памятью в режиме «исполнение с изменением конфигурации»;
- работа с периферией в режиме «исполнение с изменением конфигурации».

Недействительный

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

Основной

В этом режиме контроллер работает только с основными или альтернативными управляющими данными канала. После того, как разрешена работа канала, и контроллер получил запрос на обработку, цикл DMA выглядит следующим образом:

1. Контроллер выполняет 2^R передач. Если число оставшихся передач 0, контроллер переходит к шагу 3.
2. Осуществление арбитража:
   - если высокоприоритетный канал выдает запрос на обработку, то контроллер начинает обслуживание этого канала;
   - если периферийный блок или программное обеспечение выдает запрос на обработку (повторный запрос на обработку по каналу), то контроллер переходит к шагу 1.
3. Контроллер устанавливает dma_done[C] в состояние 1 на один такт сигнала hclk. Это указывает центральному процессору на завершение цикла DMA.

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.

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 128 демонстрирует пример функционирования контроллера в режиме «пинг-понг».

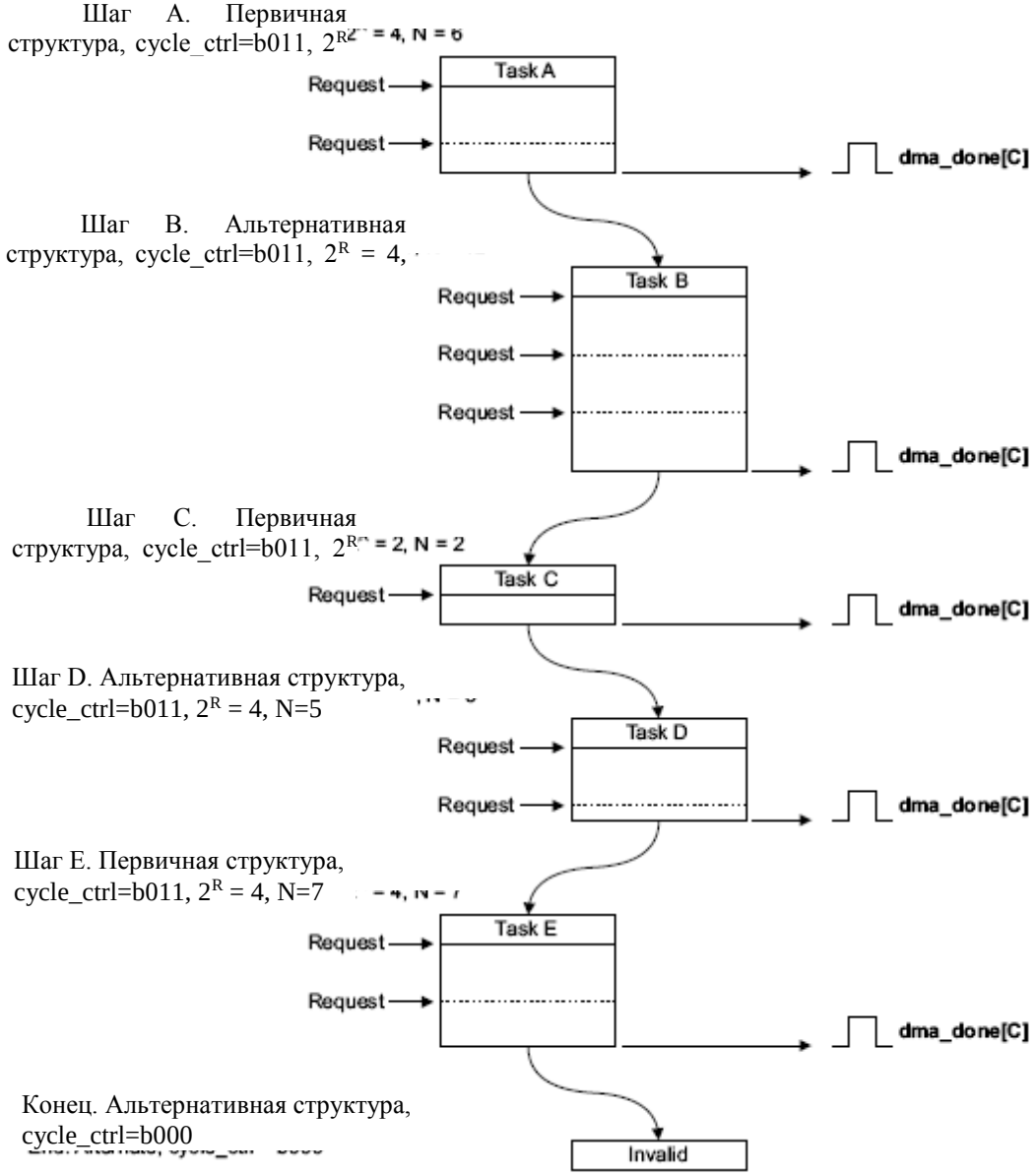


Рисунок 128. Пример функционирования контроллера в режиме «пинг-понг»

Пояснения к схеме на рисунке 128:

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.

29.4.5.1 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 390 перечисляет области памяти channel_cfg, как те, которые должны быть определены константами, так и те, значения которых определяются пользователем.

Таблица 390 – 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

№ бита	Обозначение	Значение	Описание
Области с константными значениями			
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 129 демонстрирует пример функционирования в режиме работы с памятью «Исполнение с изменением конфигурации».

	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

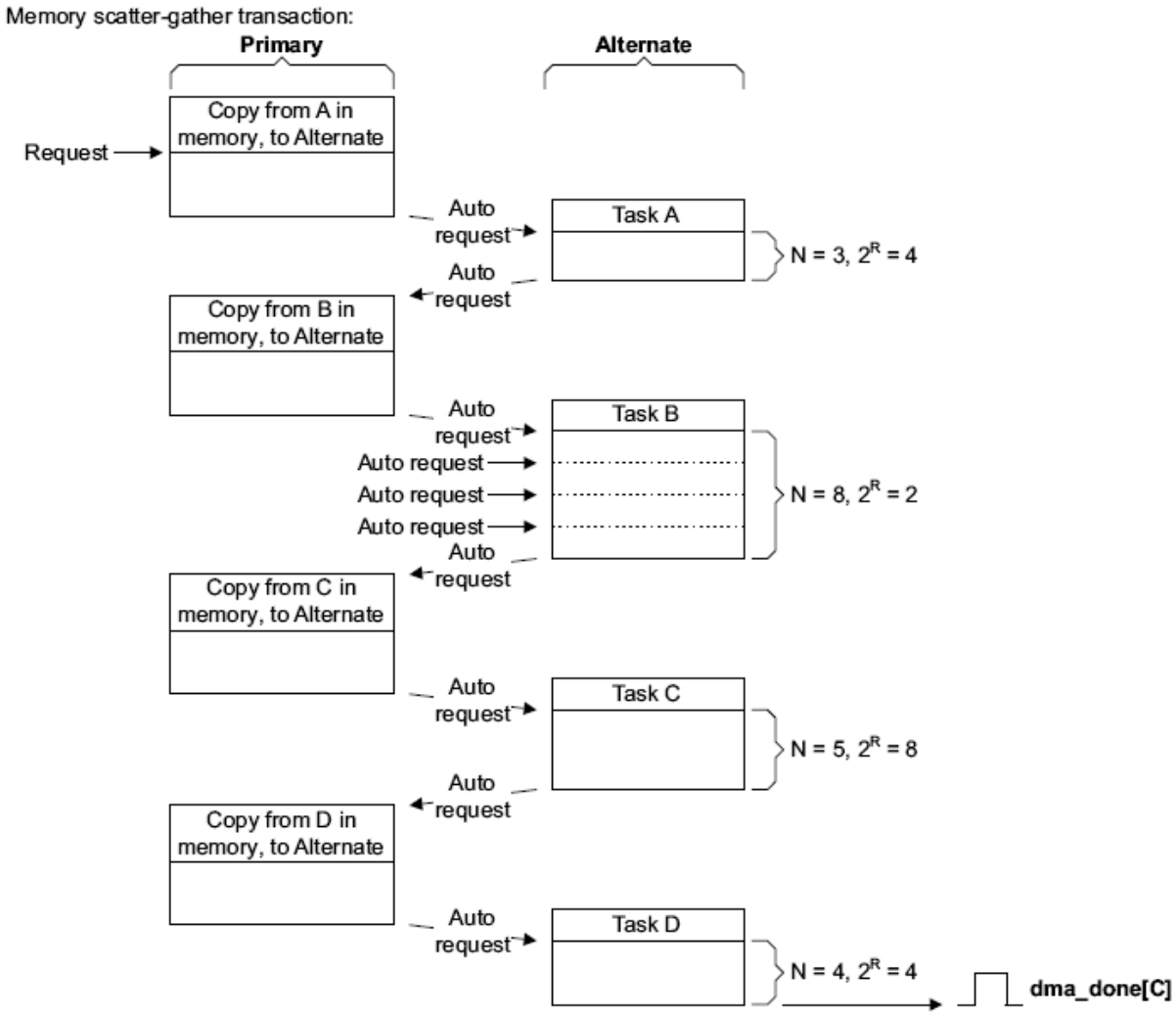


Рисунок 129. Пример работы DMA в режиме с «Исполнением с изменением конфигурации»

Пояснения к схеме на рисунке:

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.

29.4.5.2 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 391 перечисляет области памяти channel_cfg, как те, которые должны быть определены константами, так и те, значения которых определяются пользователем.

Таблица 391 – 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 (Рисунок 130) 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

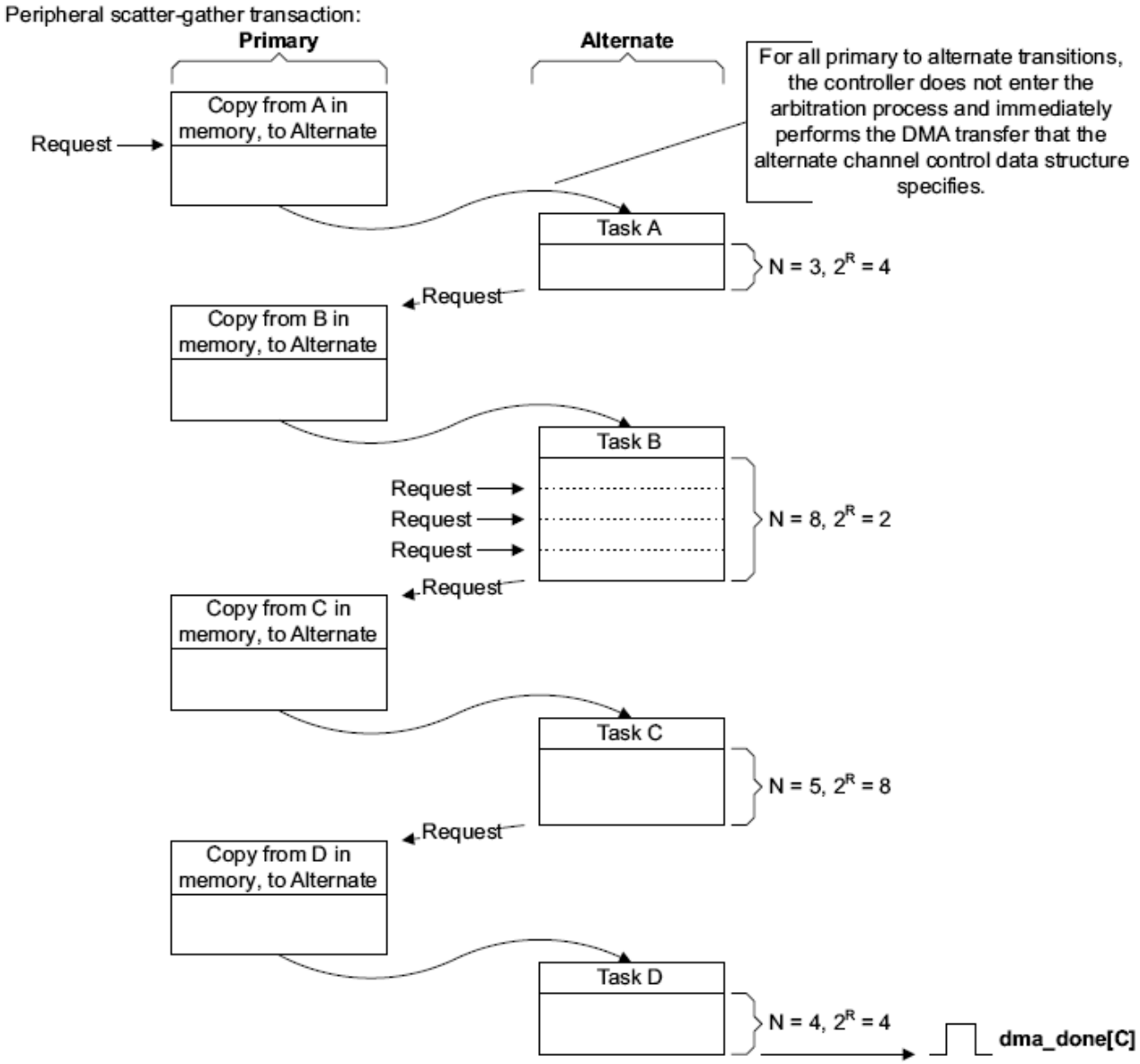


Рисунок 130 – Пример работы DMA в режиме с «Исполнением с изменением конфигурации»

Пояснения к схеме на рисунке:

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.

29.4.6 Индикация ошибок

При получении контроллером по шине AHB ответа об ошибке, он выполняет следующие действия:

- отключает канал, связанный с ошибкой;
- 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.

29.5 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 131 показывает область памяти, необходимую контроллеру для структур управляющих данных канала, при использовании всех 32 каналов и опциональной альтернативной структуры управляющих данных.

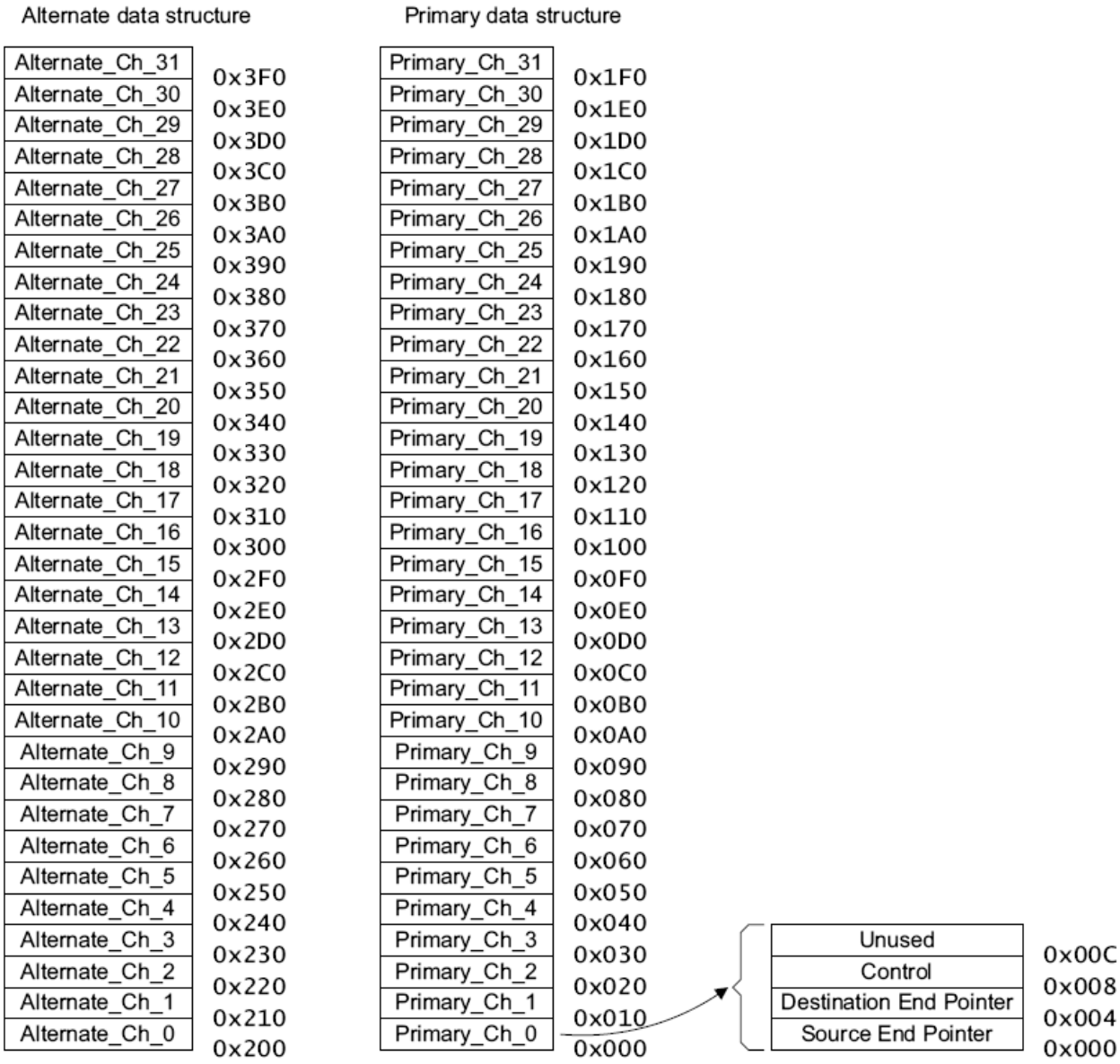


Рисунок 131. Карта памяти для 32-х каналов, включая альтернативную структуру

Пример, показанный на Рисунок 131, 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 392 перечисляет разряды адреса, обеспечивающие контроллеру доступ к различным элементам структуры управляющих данных, в зависимости от количества каналов, используемых в контроллере.

Таблица 392 – Разряды адреса, соответствующие элементам структуры управляющих данных

Количество каналов, используемых в контроллере	Разряды адреса						
	[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 132 демонстрирует пример реализации контроллера с использованием 3 каналов DMA и с альтернативной структурой управляющих данных.

Перевод текста рисунка
Destination end pointer – указатель конца данных приемника;
Source end pointer – указатель конца данных источника;
Control – управление.

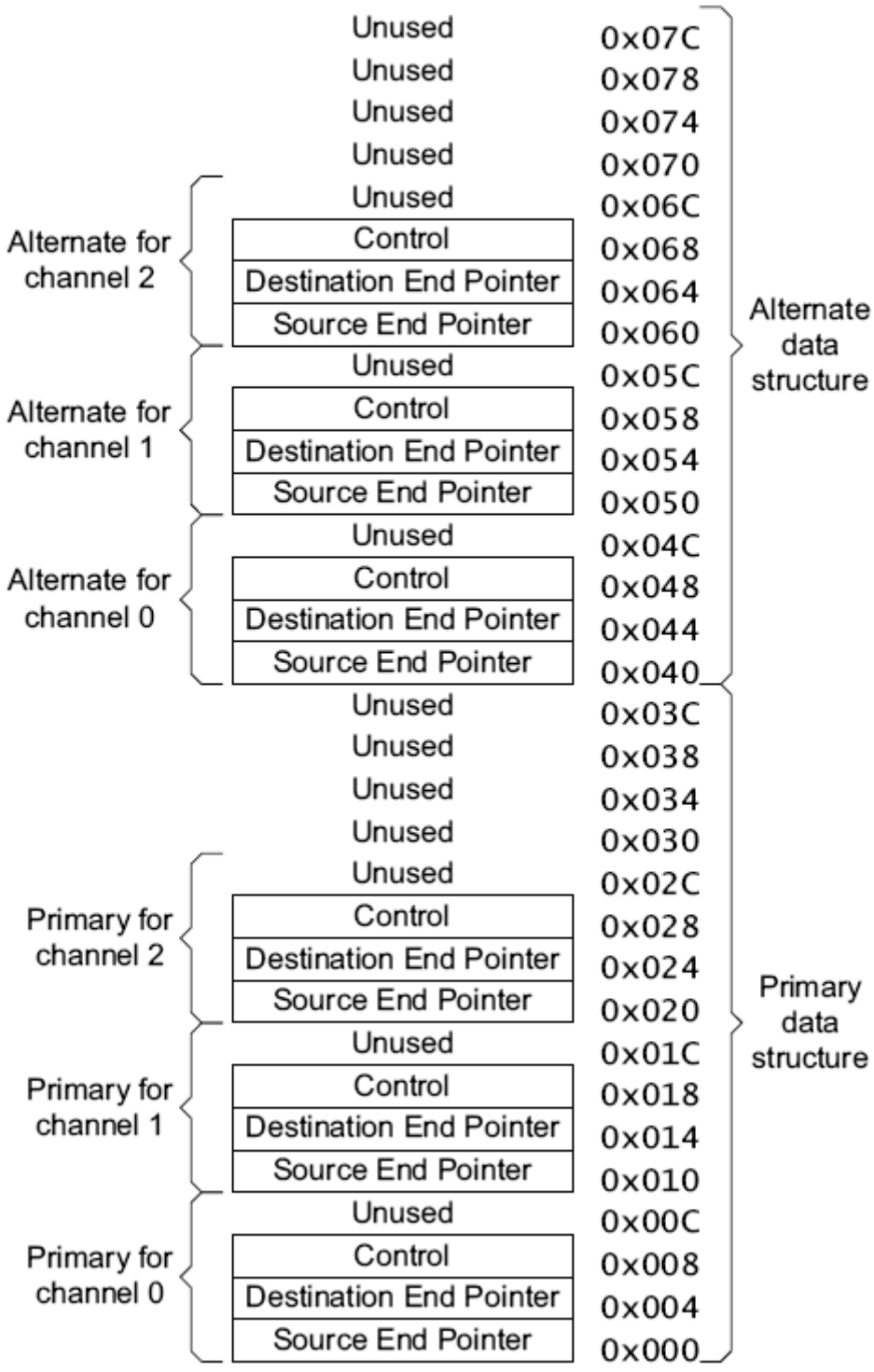


Рисунок 132. Карта памяти для трех каналов DMA, включая альтернативную структуру

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

Таблица 393 – 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-х 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 394 – 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 2^R передачи DMA.

Примечание – Контроллер не имеет доступа по записи в эту область памяти.

Указатель конца данных приемника

Область памяти под названием dst_data_end_ptr содержит указатель на последний адрес месторасположения данных приемника.

Таблица 395 – 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 2^R передачи DMA.

Примечание – Контроллер не имеет доступа по записи в эту область памяти.

Разряды управления

Область памяти под названием channel_cfg обеспечивает управление каждой передачей DMA.

Таблица 396 – 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 397 объясняет назначение разрядов этой области памяти.

Таблица 397 – Назначение разрядов channel_cfg

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31, 30	dst_inc	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

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

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

№ бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
		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 398 перечисляет адреса приемника цикла DMA для 6 слов.

Таблица 398 – 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 399 перечисляет адреса приемника для передач DMA 12 байт с использованием «полусловного» инкремента.

Таблица 399 – Цикла DMA для 12 байт с «полусловным» инкрементом

Начальные значения channel_cfg перед циклом DMA				
src_size=b00, dst_inc=b01, n_minus_1=b1011, cycle_ctrl=1, R_power=b11				
DMA передачи	**Указатель конца данных**	**Счетчик**	**Отличие*)**	**Адрес**
	0x5E7	11	0x16	0x5D1
	0x5E7	10	0x14	0x5D3
	0x5E7	9	0x12	0x5D5
	0x5E7	8	0x10	0x5D7
	0x5E7	7	0xE	0x5D9
	0x5E7	6	0xC	0x5DB
	0x5E7	5	0xA	0x5DD
	0x5E7	4	0x8	0x5DF
Значения channel_cfg после 2^R передач DMA				
src_size=b00, dst_inc=b01, n_minus_1=b011, cycle_ctrl=1, R_power=b11				
DMA передачи	0x5E7	3	0x6	0x5E1
	0x5E7	2	0x4	0x5E3
	0x5E7	1	0x2	0x5E5
	0x5E7	0	0x0	0x5E7
Конечные значения channel_cfg после цикла DMA				
src_size=b00, dst_inc=b01, n_minus_1=0, cycle_ctrl=0**), R_power=b11				

***** это значение, полученное после сдвига влево значения счетчика на количество разрядов, соответствующее dst_inc.

****** после окончания цикла DMA контроллер делает channel_cfg «неправильным», сбрасывая в 0 поле cycle_ctrl.

29.6 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 400 – 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.

29.6.1 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 402 перечисляет назначение разрядов регистра.

Таблица 401 – 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 402 – 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

29.6.2 MDR_DMA->CFG

Регистр конфигурации DMA

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

Таблица 404 перечисляет назначение разрядов регистра.

Таблица 403 – 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 404 – 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

29.6.3 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 406 перечисляет назначение разрядов регистра ctrl_base_ptr.

Таблица 405 – Регистр базового адреса управляющих данных каналов

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

Таблица 406 – 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

29.6.4 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 408 перечисляет назначение разрядов регистра.

Таблица 407 – Регистр базового адреса альтернативных управляющих данных каналов

Номер	31... 0
Доступ	RO
Сброс	0
	Alt_ctrl_base_ptr

Таблица 408 – 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

29.6.5 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 410 перечисляет назначение разрядов регистра.

Таблица 409 – 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 410 – 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

29.6.6 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 412 перечисляет назначение разрядов регистра.

Таблица 411 – 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 412 – 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

29.6.7 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 414 перечисляет назначение разрядов регистра.

Таблица 413 – 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 414 – 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[] и dma_req[]. Контроллер выполняет одиночные передачи или 2^R передач. Разряд [C] = 1 означает, что канал DMA C выполняет циклы DMA в ответ на запросы, полученные только от dma_req[]. Контроллер выполняет 2^R передач. **При записи:** Разряд [C] = 0 не дает эффекта. Необходимо использовать chnl_useburst_clr регистр и установить соответствующий разряд C в 0; Разряд [C] = 1 отключает возможность обрабатывать запросы на выполнение циклов DMA, полученные от dma_sreq[]. Контроллер выполняет 2^R передач. Запись разряда, соответствующего нереализованному каналу, не дает никакого эффекта

После выполнения предпоследней передачи из 2^R передач, в том случае, если число оставшихся передач (N) меньше чем 2^R, контроллер сбрасывает разряд chnl_useburst_set в 0. Это позволяет выполнять оставшиеся передачи, используя dma_sreq[] и dma_req[].

Примечание – При программировании channel_cfg значением N меньшим, чем 2^R, запрещена установка соответствующего разряда chnl_useburst_set в случае, если периферийный блок не поддерживает сигнал dma_req[].

В режиме работы с периферией «исполнение с изменением конфигурации», если разряд next_useburst установлен в channel_cfg, то контроллер устанавливает chnl_useburst_set [C] в 1 после окончания цикла DMA, использующего альтернативные управляющие данные.

29.6.8 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 416 перечисляет назначение разрядов регистра chnl_useburst_clr.

Таблица 415 – 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 416 – 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

29.6.9 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 418 перечисляет назначение разрядов регистра chnl_req_mask_set.

Таблица 417 – 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 418 – 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

29.6.10 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 450 перечисляет назначение разрядов регистра chnl_req_mask_clr.

Таблица 419 – Регистр очистки маскирования 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 420 – 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

29.6.11 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 422 перечисляет назначение разрядов регистра chnl_enable_set.

Таблица 421 – 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 422 – 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

29.6.12 MDR_DMA->CHNL_ENABLE_CLR

Регистр сброса разрешения каналов

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

Таблица 424 перечисляет назначение разрядов регистра chnl_enable_clr.

Таблица 423 – 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 424 – 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

29.6.13 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 426 перечисляет назначение разрядов регистра chnl_pri_alt_set.

Таблица 425 – 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 426 – 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;

29.6.14 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 428 перечисляет назначение разрядов регистра chnl_pri_alt_clr.

Таблица 427 – 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 428 – 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»

29.6.15 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 430 перечисляет назначение разрядов регистра chnl_priority_set.

Таблица 429 – 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 430 – 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 уровень приоритета. Запись разряда, соответствующего нереализованному каналу, не дает никакого эффекта

29.6.16 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 432 перечисляет назначение разрядов регистра chnl_priority_clr.

Таблица 431 – 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 432 – 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

29.6.17 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 434 перечисляет назначение разрядов регистра err_clr.

Таблица 433 – Регистр сброса флага ошибки

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

Таблица 434 – 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	err_clr	Установка сигнала в состояние 0, чтение возвращает состояние сигнала (флага) dma_err. **При чтении**: Разряд [C] = 0 означает, что dma_err находится в состоянии 0; Разряд [C] = 1 означает, что dma_err находится в состоянии 1. **При записи:** Разряд [C] =0 не дает эффекта. Состояние dma_err останется неизменным; Разряд [C] =1 сбрасывает сигнал (флаг) dma_err в состояние 0. *Примечание* – При сбросе сигнала dma_err одновременно с появлением ошибки на шине AHB-Lite, приоритет отдается ошибке, и, следовательно, значение регистра (и dma_err) остается неизменным (несброшенным)

30 Прерывания и исключения

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.

30.1 Типы исключений

Исключения бывают следующих типов:

30.1.1 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.

30.1.2 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.

30.1.3 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.

30.1.4 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.

30.1.5 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.

30.1.6 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.

30.1.7 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.

30.1.8 PendSV

Исключение PendSV является прерыванием запросом сервисов системного уровня. В приложениях с использованием ОС PendSV используется для переключения контекстов, когда нет других активных исключений.

30.1.9 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

30.2 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 435 – Различные типы исключений

Номер исключения	Номер IRQ	Тип	Приоритет	Адрес вектора обработчика (смещение)	Активация
1	-	RESET	-3, наивысший	0x0000_0004	Асинхронный

Номер исключения	Номер IRQ	Тип	Приоритет	Адрес вектора обработчика (смещение)	Активация
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.

30.3 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.

30.4 Таблица векторов

Таблица векторов содержит указатель стека, вектор входа по RESET и стартовые адреса обработчиков, также называемых векторами. Рисунок 133 представляет последовательность векторов в таблице. Младший бит всех векторов должен быть равен 1, указывая на то, что обработчик выполняется в Thumb режиме.

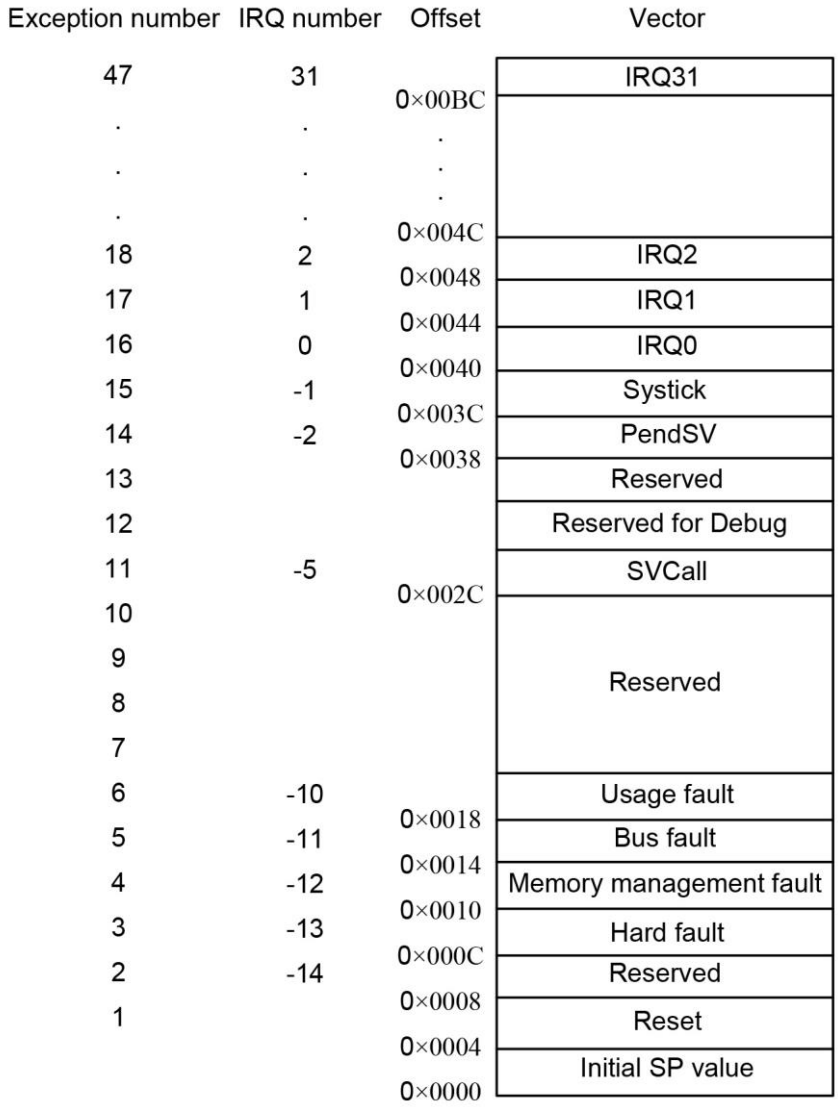


Рисунок 133. Таблица векторов исключений и прерываний

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.

30.5 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 7. 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.

30.5.1 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.

30.6 Вход в обработчик и выход из обработчика

При описании используются следующие термины:

30.6.1 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».

30.6.2 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»

30.6.3 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.

30.6.4 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.

30.6.5 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)».

30.6.6 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 таблице 436.

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 436 – 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

30.7 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.

30.7.1 Типы отказов

В таблице 437 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 437 – 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	
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.

30.7.2 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.

30.7.3 Регистры состояния и адреса отказа

Регистры состояния отказа содержат информацию о причине отказа. Для обработки отказов шины и доступа к памяти предусмотрены регистры адреса отказа, содержащие адрес, по которому произошло обращение, вызвавшее отказ. Подробная информация приведена в таблице ниже.

Таблица 438 – 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”

30.7.4 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.

30.8 Управление электропитанием

В процессоре Cortex-M3 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.

30.8.1 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.

30.8.2 Ожидание прерывания

Инструкция ожидания прерывания WFI (wait for interrupt) после своего выполнения немедленно переводит процессор в режим пониженного энергопотребления.

30.8.3 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.

30.8.4 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.

30.8.5 Выход из состояния ожидания

Условия выхода процессора из режима ожидания зависят от причины, по которой он был переведен в этот режим.

30.8.5.1 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.

30.8.5.2 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”.

30.8.6 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 439 – Формирование прерывания с IRQ0 до IRQ31

Прерывания	Блок	Принцип формирования
IRQ0	CAN1	Сигнал прерывания от блока CAN. Возникает при установленном бите GLB_INT_EN и при сигналах RX_INT_EN[31:0] и RX_INT[31:0] или EX_INT_EN[31:0] и EX_INT[31:0] или ERR_INT_EN и (ACKERR или FRAMEERR или CRCERR или BSERR или BITERR) или ERR_OVER_INT_EN и REC > CAN_ERR_MAX или TEC > CAN_ERR_MAX
IRQ1	CAN2	Аналогично

Прерывания	Блок	Принцип формирования
IRQ2	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…IRQ4	Зарезервировано	
IRQ5	DMA	Прерывания от DMA DMA_ERR или DMA_DONE. Обработка прерываний от DMA в соответствии с разделом Error signaling технического описания DMA
IRQ6	UART1	Сигнал UARTINTR
IRQ7	UART2	Сигнал UARTINTR
IRQ8	SSP1	Сигнал SSPINTR
IRQ9	Зарезервировано	
IRQ10	I2C	Сигнал INT при EN_INT
IRQ11	POWER	Сигнал прерывания от POWER Detecor
IRQ12	WWDG	Сигнал прерывания от WWDG
IRQ13	Зарезервировано	
IRQ14	Timer 1	Сигнал прерывания от Таймера. TIM_STATUS и TIM_IE
IRQ15	Timer 2	Аналогично
IRQ16	Timer 3	Аналогично
IRQ17	ADC	Сигналы прерываний от АЦП. EOCIF_1 или AWOIF_1 или EOCIF_2 или AWOIF_2
IRQ18	Зарезервировано	
IRQ19	COMP	Сигнал Rslt_Sy1
IRQ20	SSP2	Сигнал SSPINTR
IRQ21 … IRQ26	Зарезервировано	
IRQ27	BACKUP	Прерывание от BKP и часов реального времени
IRQ28	Внешнее прерывание 1	Сигнал EXT_INT1. Прерывание по уровню 0 – нет прерывания 1 – есть прерывание
IRQ29	Внешнее прерывание 2	Сигнал EXT_INT2. Прерывание по уровню 0 – нет прерывания 1 – есть прерывание
IRQ30	Внешнее прерывание 3	Сигнал EXT_INT3. Прерывание по уровню 0 – нет прерывания 1 – есть прерывание
IRQ31	Внешнее прерывание 4	Сигнал EXT_INT4. Прерывание по уровню 0 – нет прерывания 1 – есть прерывание

31 Контроллер прерываний NVIC

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 7 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 440.

Таблица 440 – Обобщенная информация о регистрах контроллера NVIC

Адрес	Название	Тип	Доступ	Значение после сброса	Описание
0xE000E100	NVIC				Контроллер прерываний NVIC
0x000	ISER[0]	RW	Привилегированный	0x00000000	Регистр разрешения прерываний ISER
…					
0x01C	ISER[7]				
…					
0x080	ICER[0]	RW	Привилегированный	0x00000000	Регистр запрета прерываний ICER
…					
0x09C	ICER[7]				
…					
0x100	ISPR[0]	RW	Привилегированный	0x00000000	Регистр установки состояния ожидания для прерывания ISPR
…					
0x11C	ISPR[7]				
…					
0x180	ICPR[0]	RW	Привилегированный	0x00000000	Регистр сброса состояния ожидания для прерывания ICPR
…					
0x19C	ICPR[7]				
…					
0x200	IABR[0]	RO	Привилегированный	0x00000000	Регистр активных прерываний IABR
…					
0x21C	IABR[7]				
…					

Адрес	Название	Тип	Доступ	Значение после сброса	Описание
0x300	IP[3],IP[2], IP[1],IP[0]	RW	Привилегированный	0x00000000	Регистр приоритета прерываний IP
...					
0x3F0	IP[239], IP[238], IP[237], IP[236]				
...					
0xE00	STIR	WO	В зависимости от конфигурации*)	0x00000000	Регистр программного формирования прерываний STIR

***** – Более подробную информацию см. в описании регистра.

31.1 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;
- 3-битные поля регистра приоритета прерываний отображаются на массив 3-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».

В Таблица 441 показано отображение прерываний (номеров запросов IRQ) на регистры прерываний и соответствующие переменные CMSIS, для которых предусмотрено по одному биту на прерывание.

Таблица 441 – Распределение прерываний в переменных прерывания

Номер прерывания	Элементы массивов CMSIS*)				
	Разрешение	Запрет	Установка режима ожидания	Сброс режима ожидания	Признак активности
0 – 31	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

31.1.1 NVIC->ISER[x]

Регистр ISER0 предназначен для разрешения прерываний (запись) и определения, какие из прерываний разрешены (чтение).

Таблица 442 – 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.

31.1.2 NVIC->ICER[x]

Регистр запрета прерываний

Регистр ICER0 предназначен для запрета прерываний (запись) и определения, какие из прерываний разрешены (чтение).

Таблица 443 – 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.

31.1.3 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 444 – 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.

31.1.4 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 445 – 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.

31.1.5 NVIC->IABR[x]

Регистр активных прерываний

Регистр ICPR0 показывает, какие из прерываний находятся в активном состоянии. Этот регистр доступен только для чтения (Таблица 446).

Таблица 446 – 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.

31.1.6 NVIC->IP[x]

Регистры приоритета прерываний

Регистры IPR0-IPR7 представляют собой набор 3-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 447 – 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 7, 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].

31.1.7 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 448 – 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.

31.2 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.

31.3 Аппаратное и программное управление прерываниями

Процессор Cortex-M3 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. п. NVIC->ISPR[x]) или соответствующего значения в регистр STIR (см. п. 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.

31.4 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 449 – 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.

32 Блок управления системой

Блок управления системой SCB обеспечивает доступ к информации о конфигурации и управление работой системы. Регистры блока управления системой представлены в таблице 450.

Таблица 450 – 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 идентификации 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 приоритета 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 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

32.1 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].

32.1.1 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 таблице ниже.

Таблица 451 – 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 п. «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 Cortex-M3.

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.

32.1.2 SCB->CPUID

Регистр идентификации процессора

Регистр CPUID содержит информацию о модели процессора, версии и варианте его реализации. Подробная информация о регистре представлена в Таблица 452.

Таблица 452 – 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 = ARM.
Variant – значение r в номере версии rnpn изделия: 0x2 = r2p0;
Constant – постоянное значение 0xF;
PartNo – номер модели процессора: 0xC23 = Cortex-M3;
Revision – значение p в номере версии rnpn изделия: 0x0 = r2p0.

32.1.3 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 453 – 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.

32.1.4 SCB->VTOR

Регистр смещения таблицы векторов прерываний

Регистр VTOR содержит смещение базового адреса таблицы векторов прерываний относительно адреса 0x00000000.

Таблица 454 – 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.

32.1.5 SCB->AIRCR

32.1.5.1 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 455 – 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 таблице ниже.

Таблица 456 – 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**
0b100	bxxx.00000	[7:5]	None	8	1
0b101	bxx.y00000	[7:6]	5	4	2
0b110	bx.yy00000	[7]	[6:5]	2	4
0b111	b.yyy00000	None	[7:5]	1	8

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.

32.1.6 SCB->SCR

Регистр управления системой

Регистр SCR позволяет определить требования к переходу в режим и выходу из режима пониженного энергопотребления.

Таблица 457 – Регистр управления системой

Номер	31…5	4	3	2	1	0
Доступ	U	R/W	U	R/W	R/W	U
Сброс	0	0	0	0	0	0
	-	**SEVONPEND**	**-**	**SLEEPDEEP**	**SLEEONEXIT**	**-**

SEVONPEND – разрешает или запрещает формирование сигнала события при переводе исключения в состояние ожидания обработки. 0 – выход из режима пониженного энергопотребления по прерыванию могут инициировать только разрешенные прерывания или события; 1 – выход может инициироваться разрешенными событиями и любыми, в том числе запрещенными, прерываниями.

Перевод прерывания в состояние ожидания обслуживания формирует событие, что в свою очередь приводит к выходу процессора из режима пониженного потребления, инициированного инструкцией WFE, либо к регистрации факта события, если эта инструкция еще не выполнялась.

Кроме того, процессор может быть выведен из режима пониженного энергопотребления при поступлении внешнего события, а также после выполнения инструкции SEV.

SLEEPDEEP – определяет режим пониженного энергопотребления процессора:
0 – спящий режим (Sleep);
1 – режим глубокого сна (Deep Sleep).

SLEEPONEXIT – разрешает или запрещает перевод процессора в режим пониженного энергопотребления при выходе из обработчика события в режим выполнения прикладной программы: 0 – не переводить, 1 – переводить.

32.1.7 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 458 – 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”).

32.1.8 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 459 – 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 460 – 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 461 – 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 462 – 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

32.1.9 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 463 – 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:

- установка бита разрешения в 1 разрешает обработку исключения, установка в 0 – запрещает;
- 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;
- 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.

32.1.10 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 464 – 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.

32.1.10.1 Поле MMFSR

Регистр состояния отказов доступа к памяти

Регистр MMFSR содержит набор флагов, указывающих на различные причины отказа доступа к памяти.

Таблица 465 – 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 работы, так как значение регистра MMAR будет изменено.

MSTKERR признак отказа на этапе сохранения в стеке контекста при передаче управления на обработчик исключения: 0 = отсутствует, 1 = попытка сохранения в стеке контекста при вызове обработчика исключения вызвала одно или несколько нарушений доступа к памяти. В случае, если бит равен 1, значение указателя стека SP по-прежнему корректно, однако содержимое стека может быть неверным. Адрес отказа в регистр MMAR не записывается.

MUNSTKERR признак отказа на этапе восстановления контекста из стека при выходе из обработчика исключения: 0 = отсутствует, 1 = попытка восстановления контекста из стека вызвала одно или несколько нарушений доступа к памяти.
Передача управления на обработчик данного отказа осуществляется без сохранения контекста. Таким образом, в случае, если данный бит равен 1, состояние стека сохраняется, значение указателя стека не меняется, контекст не сохраняется.
Адрес отказа в регистр MMAR не записывается.

DACCVIOL признак нарушения доступа к памяти данных: 0 = отсутствует, 1 = процессор попытался прочесть или записать данные в области, для которой не разрешен такой тип доступа. Если бит равен 1, значение счетчика команд PC, сохраненное в стеке, указывает на инструкцию, вызвавшую отказ. В регистре MMAR содержится адрес, по которому была осуществлена попытка доступа к памяти.

IACCVIOL признак нарушения доступа к памяти команд: 0 = отсутствует, 1 = процессор попытался считать очередную команду из области памяти, для которой не разрешено выполнение. Этот отказ возникает всякий раз при доступе к области, помеченной как неразрешенная для выполнения (XN), даже в случае, если блок защиты памяти MPU не активен (disabled) или отсутствует. Если бит равен 1, значение счетчика команд PC, сохраненное в стеке, указывает на инструкцию, вызвавшую отказ. Адрес отказа в регистр MMAR не записывается.

32.1.10.2 Поле BFSR

Регистр состояния отказов доступа к шине

Регистр BFSR содержит набор флагов, указывающих на различные причины отказа доступа к шине:

Таблица 466 – 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.

32.1.10.3 Поле UFSR

Регистр состояния отказов, вызванных ошибками программирования

Регистр UFSR содержит набор флагов, указывающих на различные причины отказа.

Таблица 467 – 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. п. **“**SCB->CCR**”**).

UNALIGNED признак доступа к памяти по невыровненному адресу: 0 = не было, либо обработка данного типа ошибки запрещена, 1 = процессор попытался обратиться к памяти по невыровненному адресу. Разрешить либо запретить обработку этой ошибки можно путем установки в 1 бита UNALIGN_TRP регистра CCR (см. п. “SCB->CCR”).
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.

32.1.11 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 468 – 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.

32.1.12 SCB->MMFAR

Регистр адреса отказа доступа к памяти

Регистр MMFAR содержит адрес, при обращении по которому возникла ошибка управления памятью.

Таблица 469 – 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”.

32.1.13 SCB->BFAR

Регистр адреса отказа доступа к шине

Регистр BFAR содержит адрес, при обращении по которому возникла ошибка доступа к шине.

Таблица 470 – 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”.

32.1.14 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.

33 Сторожевые таймеры

33.1 Описание регистров блока сторожевых таймеров

Таблица 471 – Описание регистров блока сторожевых таймеров

Базовый Адрес	Название	Описание
0x4006_8000	MDR_IWDG	Сторожевой таймер IWDG
Смещение		
0x00	MDR_IWDG->KR[15:0]	Регистр Ключа
0x04	MDR_IWDG->PR[2:0]	Делитель частоты сторожевого таймера
0x08	MDR_IWDG->RLR[11:0]	Регистр основания счета сторожевого таймера
0x0C	MDR_IWDG->SR[1:0]	Регистр статуса сторожевого таймера

Таблица 472 – Оконный сторожевой таймер

Базовый Адрес	Название	Описание
0x4006_0000	MDR_WWDG	Сторожевой таймер WWDG
Смещение		
0x00	MDR_WWDG->CR[7:0]	Регистр управления
0x04	MDR_WWDG->CFR[9:0]	Регистр конфигурации
0x08	MDR_WWDG->SR[0]	Регистр статуса

33.1.1 MDR_IWDG->KR

Таблица 473 – Регистр KR

Номер	15...0
Доступ	WO
Сброс	0
	KEY[15:0]

Таблица 474 – Описание бит регистра KR

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31…16		Зарезервировано
15...0	KEY[15:0]	**Значение ключа (только запись, читается 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 PR и RLR. Запись ключевого значения CCCCh разрешает работу сторожевого таймера. Сторожевой таймер IWDT сбрасывается и останавливается только выключением питания.

33.1.2 MDR_IWDG->PR

Таблица 475 – Регистр PR

Номер	31…3	2	1	0
Доступ	U	R/W	R/W	R/W
Сброс	0	0	0	0
	-	**PR2**	**PR1**	**PR0**

Таблица 476 – Описание регистра PR

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31…3		Зарезервировано
2…0	PR[2:0]	**Делитель частоты сторожевого таймера:** 000 – делитель на 4 001 – 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 допустимы только, если бит PVU = 0 в регистре SR Сторожевой таймер IWDG тактируется частотой LSI генератора

33.1.3 MDR_IWDG->RLR

Таблица 477 – Регистр RLR

Номер	31...12	11…0
Доступ	U	R/W
Сброс	0	1
	-	**RLR[11:0]**

Таблица 478 – Описание регистра RLR

№ бита	**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 бит по доступу защищено с помощью регистра KR. Эти биты записываются программно и определяют значение, загружаемое в сторожевой таймер в момент записи значение AAAAh в регистр KR. Сторожевой таймер декрементируется, начиная с этого значения. Период таймаута сторожевого таймера функция от этого значения и делителя частоты. Чтение и запись этого регистра допустимы только, если бит RVU=0 в регистре SR

33.1.4 MDR_IWDG->SR

Таблица 479 – Регистр SR

Номер	31…2	1	0
Доступ	U	RO	RO
Сброс	0	0	0
	-	**RVU**	**PVU**

Таблица 480 – Описание регистра SR

№ бита	**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

33.1.5 MDR_WWDG->CR

Таблица 481 – Регистр CR

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	1	1	1	1	1	1	1
	-	**WDGA**	**T6**	**T5**	**T4**	**T3**	**T2**	**T1**	**T0**

Таблица 482 – Описание бит регистра CR

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31...8		Зарезервировано
7	WDGA	**Бит активации** Этот бит устанавливается программно и очищается только аппаратно при сбросе. Когда WDGA=1, сторожевой таймер может генерировать сброс: 0 – сторожевой таймер отключен; 1 – сторожевой таймер включен
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

33.1.6 MDR_WWDG->CFR

Таблица 483 – Регистр CFR

Номер	31...10	9	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	R/W	R/W
Сброс		0	0	0	1	1	1	1	1	1	1
	-	**EWI**	**WDGTB1**	**WDGTB0**	**W6**	**W5**	**W4**	**W3**	**W2**	**W1**	**W0**

Таблица 484 – Описание бит регистра CFR

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31…10		Зарезервировано
9	EWI	**Раннее предупреждающее прерывание.** Если бит установлен, то разрешается генерация прерывания при достижении сторожевым таймером 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 бит T[6:0] значением в пределах 40h-7Fh. Если происходит инициализация бит в момент T>W, то формируется сброс на выходе RESET. Если таймер достигнет значения T=3Fh, то также формируется сброс

33.1.7 MDR_WWDG->SR

Таблица 485 – Регистр SR

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

Таблица 486 – Описание бит регистра SR

№ бита	**Функциональное имя бита**	**Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений**
31…1		Зарезервировано
0	EWIF	**Флаг раннего предупреждающего прерывания.** Этот бит устанавливается аппаратно, когда сторожевой таймер достигает значения 40h. Бит очищается программно записью нуля. Запись единицы не влияет. Этот бит также устанавливается, если прерывание запрещено EWI=0

34 Предельно допустимые характеристики микросхемы

Таблица 487 – Предельно допустимые и предельные режимы эксплуатации микросхем

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}	2,2	3,6	–	4,0
Напряжение источника питания при использовании USB, В		3,0	3,6	–	4,0
Напряжение источника питания АЦП, ЦАП, В*	U_{CCA}	2,4	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)	U_{IL}				
на выводах: PA, PB, PC, PD, PE, PF, RESET, WAKEUP, SHDN, JTAG_EN		0	0,8	– 0,3	–
на выводах: DN, DP		0	0,8	– 0,3	–
на выводе: OSC_IN HSE BYPASS = 1		0	0,8	– 0,3	–
Входное напряжение высокого уровня, В,	U_{IH}				
на выводах: PD, PE (0-10)		2,0	3,6	–	4,0
на выводах: PA, PB, PC, PE (11-15), PF, RESET, WAKEUP, SHDN, JTAG_EN		2,0	5,25	–	5,3
на выводах: DN, DP		2,0	3,6	–	4,0
на выводах: OSC_IN HSE BYPASS = 1		2,0	3,6	–	4,0
Выходной ток высокого уровня, мА, (при работе в цифровом режиме)	I_{OH}				
на выводах: PA, PB, PC, PD, PE (0-5, 8-15), PF		– 6	–	– 10	–
на выводах PE (6, 7), STANDBY		– 3	–	– 10	–
на выводах: DN, DP		– 6	–	– 40	–
Выходной ток низкого уровня, мА,	I_{OL}				
на выводах: PA, PB, PC, PD, PE, PF, STANDBY		–	6	–	10
на выводах: DN, DP		–	6	–	40

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, МГц	f_C	–	80	–	–
Частота следования импульсов тактовых сигналов HSE, МГц при: BYPASS = 0	f_{C_HSE}	2	16	–	–
при: BYPASS = 1		0	80	–	–
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 опорного напряжения ЦАП, В, на выводе: REFD0, REFD1 при: Cfg_M_REFx = 1	$U_{REF(DAC)}$	2,4	U_{CCA}	–	–
Резистивная нагрузка ЦАП, кОм	R_{LOAD}	10	–	–	–
Емкостная нагрузка ЦАП, пФ	C_{LOAD}	–	100	–	–
Параметры АЦП					
Напряжение нижней границы внешнего опорного напряжения АЦП, В, при: ADC1_Cfg_M_REF = 1 или ADC2_Cfg_M_REF = 1	U_{ADC1_REF-}	0	U_{CCA}-2,4	– 0,3	–
Напряжение верхней границы внешнего опорного напряжения АЦП,В при: ADC1_Cfg_M_REF = 1 или ADC2_Cfg_M_REF = 1	U_{ADC0_REF+}	2,4	U_{CCA}	–	4,0
Диапазон напряжения внешнего опорного источника АЦП, В, $U_{REF(ADC)} = U_{ADC0\_REF+} - U_{ADC1\_REF-}$	$U_{REF(ADC)}$	2,4	U_{CCA}	–	–
Диапазон напряжения на входе АЦП, В**	U_{AIN}	U_{ADC1_REF-}	U_{ADC0_REF+}	– 0,3	4,0
Частота следования импульсов тактовых сигналов АЦП, МГц	f_{C_ADC}	–	14	–	–

Наименование параметра, единица измерения	Буквенное обозначение параметра	Норма параметра			
		Предельно допустимый режим		Предельный режим	
		не менее	не более	не менее	не более
Емкость нагрузки, пФ, на выводах: PA, PB, PC, PD, PE, PF, Standby	C_L	–	30	–	–
Число циклов записи/стирания данных, при: T = 125 °C	N_{PR}	10 000	–	–	–
Время хранения информации, лет, при: T = 25 °C	t_{GS}	25	–	–	–
при: T = 85 °C		10	–	–	–
при: T = 125 °C		1	–	–	–

* Допускается использование отдельного источника для питания АЦП и ЦАП, но при этом его выходное напряжение не должно отличаться от U_{CC} более чем на ± 0,2 В.

** При использовании внутреннего опорного напряжения для АЦП $U_{ADC1\_REF-} = GND_A$ и $U_{ADC0\_REF+} = U_{CCA}$, для ЦАП $U_{REF(DAC)} = U_{CCA}$.

Примечание – Не допускается одновременное воздействие двух и более предельных режимов

35 Электрические параметры микросхемы

Таблица 488 – 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, 125, – 60
Выходное напряжение высокого уровня, В, на выводах: PA, PB, PC, PD, PE, PF, STANDBY, DN, DP при: U_{CC} = 3,0 В	U_{OH}	2,4	–	
Выходное напряжение высокого уровня, В, на выводах: PA, PB, PC, PD, PE, PF, STANDBY при: U_{CC} = 2,2 В		1,6	–	
Уровень напряжения питания для срабатывания схемы генерации сброса, В	U_{BOR}	1,8	2,1	
Входной ток утечки высокого уровня, мкА, на выводах: PA, PB, PC, PD, PE, PF, RESET, WAKEUP, DN, DP	I_{ILH}	– 1,0	1,0	
Входной ток утечки высокого уровня, мкА, на выводе: OSC_IN при: BYPASS = 1		– 40,0	40,0	
Входной ток утечки низкого уровня, мкА, на выводах: PA, PB, PC, PD, PE, PF, RESET, WAKEUP, DN, DP	I_{ILL}	– 1,0	1,0	
Входной ток утечки низкого уровня, мкА, на выводе: OSC_IN при: BYPASS = 1		– 1,0	1,0	
Статический ток потребления в режиме покоя (регулятор напряжения выключен), мкА	I_{CCS}	–	10	
		–	20	
Динамический ток потребления, мА	I_{OCC1}	–	120	
Выходная частота LSI RC-генератора, кГц	f_{O_LSI}	10	60	
Выходная частота HSI RC-генератора, МГц	f_{O_HSI}	6	10	
Выходная частота PLL, МГц максимальная минимальная	f_{O_PLL}	 100 –	 – 6	
Параметры АЦП				
Разрядность АЦП	E_{NADC}	12	–	25, 125, – 60
Дифференциальная нелинейность АЦП, единица младшего разряда	E_{DLADC}	– 1	2	
Интегральная нелинейность АЦП, единица младшего разряда	E_{ILADC}	– 3	3	
Ошибка смещения АЦП, единица младшего разряда	E_{OFFADC}	– 6	6	

Наименование параметра, единица измерения, режим измерения	**Буквенное обозначение параметра**	**Норма параметра**		**Температура среды, °С**
		не менее	**не более**	
Ошибка усиления АЦП, %	$E_{GAINADC}$	– 1	1	25, 125, – 60
Параметры ЦАП				
Разрядность ЦАП	E_{NDAC}	12	–	25, 125, – 60
Дифференциальная нелинейность ЦАП, единица младшего разряда	E_{DLDAC}	– 1	2	
Интегральная нелинейность ЦАП, единица младшего разряда	E_{ILDAC}	– 6	6	
Ошибка смещения ЦАП, мВ	E_{OFFDAC}	– 40	40	
Ошибка усиления ЦАП, %	$E_{GAINDAC}$	– 2	2	
Минимальное выходное напряжение ЦАП, В	$U_{O\_DAC\ min}$	–	0,08	
Максимальное выходное напряжение ЦАП, В	$U_{O\_DAC\ max}$	$U_{REF(DAC)}$-0,08	–	
Компаратор				
Время включения компаратора, мкс	t_{ON_C}*	–	100	25, 125, – 60
Время задержки переключения компаратора, нс	t_{d_C}*	–	400	

* Значения временных параметров t_{ON_C}, t_{d_C} гарантируются в процессе проведения ФК на максимальной частоте.

Примечание – Режимы измерения параметров приведены в технических условиях АЕЯР.431290.711ТУ

Примечание – Значения электрических параметров микросхем в бескорпусном исполнении на общей пластине, неразделенные при приемке и поставке соответствуют нормам для нормальных климатических условий, приведенным выше (Таблица 488).

36 Справочные данные

Зависимости основных электрических параметров микросхем от режимов и условий эксплуатации приведены на рисунках 134 – 141.

Значение собственной резонансной частоты:

- 1986BE91T, 1986BE94T - не менее 3 700 Гц;
- 1986BE92У - не менее 4 300 Гц;
- 1986BE92У1 - не менее 5 800 Гц;
- 1986BE93У - не менее 4 500 Гц;
- 1986BE94Ф - не менее 8 300 Гц;
- 1986BE94Я - не менее 8 600 Гц.

Значение теплового сопротивления кристалл – окружающая среда:

- 1986BE91T, 1986BE94T - не более 39 °С/Вт;
- 1986BE92У - не более 36 °С/Вт;
- 1986BE92У1 - не более 6 °С/Вт;
- 1986BE93У - не более 29 °С/Вт;
- 1986BE94Ф - не более 10 °С/Вт;
- 1986BE94Я - не более 12 °С/Вт.

Значение емкости входа-выхода на частоте 1 МГц:

- 1986BE91T, 1986BE94T - не более 12,5 пФ;
- 1986BE92У - не более 11 пФ;
- 1986BE93У - не более 10 пФ.

Значения предельно допустимых ОИН приведены в таблице 489.

Таблица 489 – Предельно допустимые значения ОИН

Тип вывода	**Длительность ОИН, мкс**	
	Предельно допустимое напряжение ОИН, В	
	1,0	**10,0**
Входы	1 000	300
Выходы	1 750	500
Цепь питания	1 750	1 000

Справочные параметры микросхемы приведены в таблице 490.

Таблица 490 – Справочные данные

Наименование параметра, единица измерения, режим измерения	Буквенное обозначение параметра	Норма параметра		Температура среды, °С
		не менее	не более	
Статический ток потребления в режиме покоя (регулятор напряжения включен), мА, при U_{CC} = 3,6 В	I_{CCS1}	–	1,5	25, 125, – 60
Ток потребления батарейного домена, мкА, при U_{CC} = 0 В	I_{CC_B}	–	5	
Динамический ток потребления в режиме ожидания SLEEPDEEP, мА, при U_{CC} = 3,6 В, $f_C = f_{O\_LSI}$	I_{OCC2}	–	2	
Время установления сигналов PBD и PBVD, мкс	$t_{SU(PBD)}$ $t_{SU(PBVO)}$	–	2	
Выходное напряжение регулятора LDO, В при U_{CC} = 3,6 В, I_{OL} = 80 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 U_{CC} = 2,2 В, - 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 U_{CC} = 3,0 В, - Full Speed, C_l = 50 пФ - Low Speed, C_l = 600 пФ	$t_{W(DN, DP)}$	– –	15 300	
Сопротивление внутренних программируемых резисторов доопределения, кОм	R_{PULL}	20	100	
Компаратор				
Напряжение смещения компаратора, мВ, при U_{CC} = 3,6 В	U_{IO_C}	–	|±0,5|	25, 125, – 60
Гистерезис компаратора, мВ, при U_{CC} = 3,6 В	ΔU_{TH_C}	8	12	
Напряжение внутреннего опорного источника компаратора, В, при: U_{CC} = 3,6 В	U_{REF_C}	1,17	1,23	

Наименование параметра, единица измерения, режим измерения	Буквенное обозначение параметра	Норма параметра		Температура среды, °С
		не менее	не более	
Тактовые частоты и генераторы				
Время установления сигнала HSIRDY относительно HSION, мкс, при U_{CC} = 2,2 В	$t_{SU(HSI)}$	–	1	25, 125, – 60
Время установления сигнала LSIRDY относительно LSION, мс, при U_{CC} = 2,2 В	$t_{SU(LSI)}$	–	80	
Время установления сигнала HSERDY относительно HSEON, мкс, при U_{CC} = 2,2 В	$t_{SU(HSE)}$	–	$2048/f_{C\_HSE}$	
Время установления сигнала LSERDY относительно LSEON, мкс, при U_{CC} = 2,2 В	$t_{SU(LSE)}$	–	$4096/f_{C\_LSE}$	
Время установления сигнала PLLRDY относительно PLLON, мкс, при U_{CC} = 2,2 В	$t_{SU(PLL)}$	–	100	
Длительность сигнала сброса, мкс, при U_{CC} = 2,2 В	$t_{W(СБРОС)}$	20	–	
Время запуска после сброса по POR, мс	t_{POR}	–	6	
Минимальная длительность сигнала низкого уровня на входе nRESET для формирования сигнала «Сброс», нс	$t_{W(nRESET)min}$	200	–	
Максимальная длительность сигнала низкого уровня на входе nRESET, при котором не формируется сигнал «Сброс», нс	$t_{W(nRESET)max}$	–	10	
АЦП				
Время выборки заряда АЦП, нс, при Ucc = 3,6 В	t_{A_ADC}	–	$4/f_{C\_ADC}$	25, 125, – 60
Время преобразования АЦП, нс при U_{CC} = 3,6 В	t_{AO_ADC}	–	$28/f_{C\_ADC}$	
Ток потребления по входу внешней верхней границы опоры АЦП, мкА при - ADC1_Cfg_M_REF = 1 или - ADC2_Cfg_M_REF = 1	I_{ADC0_VREF+}	–	50	
Ток потребления по входу внешней нижней опоры АЦП, мкА при - ADC1_Cfg_M_REF = 1 или - ADC2_Cfg_M_REF = 1	I_{ADC0_VREF-}	– 50	–	
Ток потребления по питанию АЦП, мА при: U_{CCA} = 3,6 В, f_{C_ADC} = 14 МГц	I_{OCCADC}	–	3	
Минимальная частота преобразования АЦП, кГц	f_{C_ADCMIN}	10	–	

Наименование параметра, единица измерения, режим измерения	**Буквенное обозначение параметра**	**Норма параметра**		**Температура среды, °С**
		не менее	**не более**	
ЦАП				
Время установления сигнала ЦАП, мкс, при U_{CC} = 3,6 В, C_l = 50 пФ, R_l = 10 кОм	$t_{SU(DAC)}$	–	5,2	25, 125, – 60
Время включения ЦАП, мкс, при U_{CC} = 2,4 В	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 - D+PULLUP = 1	R_{DN-UCC}	1	2	25, 125, – 60
	R_{DP-UCC}	1	2	
Сопротивление резистора между линиями DN, DP и шиной «Общий», кОм при - D-PULLDOWN = 1 - D+PULLDOWN = 1	R_{DN-GND}	10	20	
	R_{DP-GND}	10	20	
Согласующее сопротивление на линиях DN, DP, Ом	R_{DN} R_{DP}	14	34	

37 Типовые зависимости

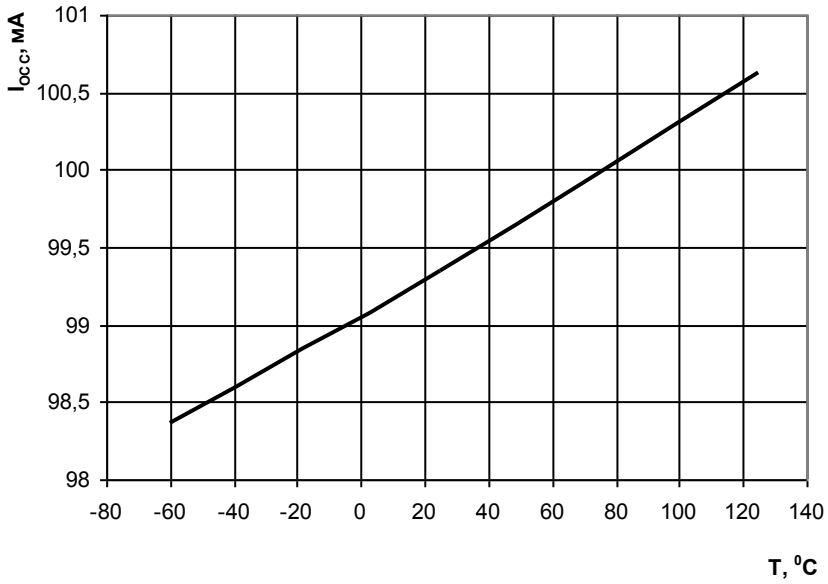


Рисунок 134. Зависимость динамического тока потребления от температуры при: f_C = 80 МГц

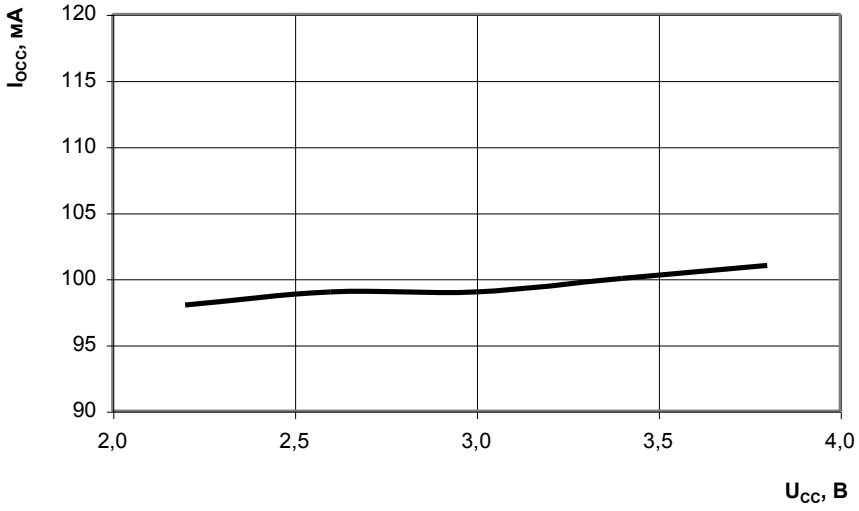


Рисунок 135. Зависимость динамического тока потребления от напряжения питания

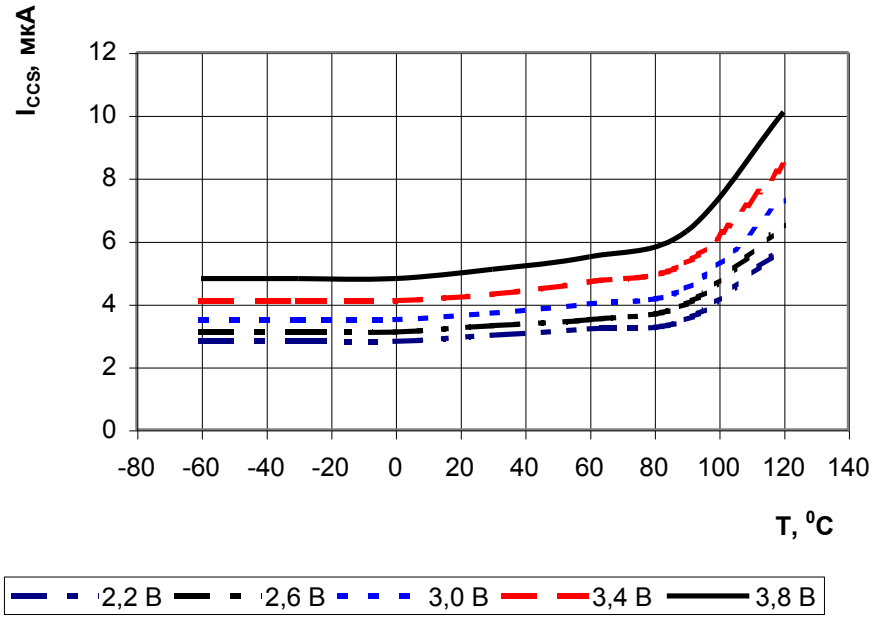


Рисунок 136. Зависимость статического тока потребления в режиме покоя (регулятор напряжения выключен) от температуры при разном напряжении

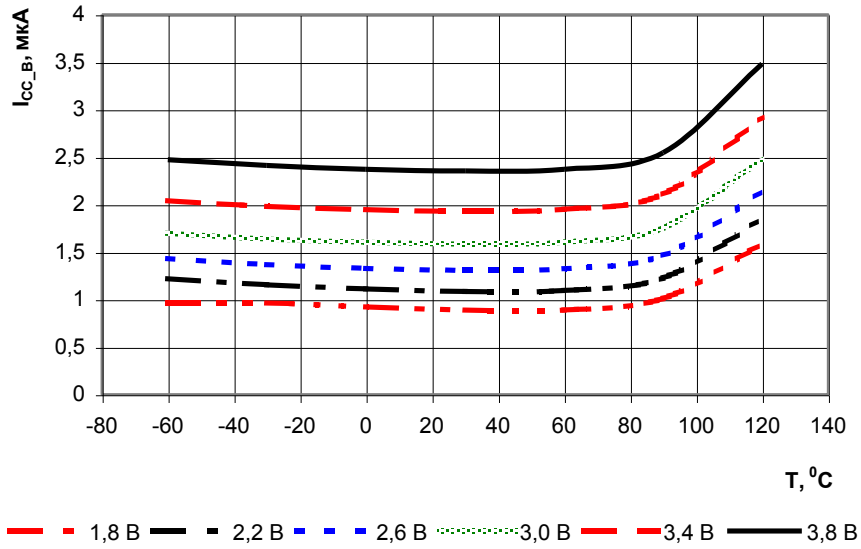


Рисунок 137. Зависимость тока потребления батарейного домена от температуры при разном напряжении

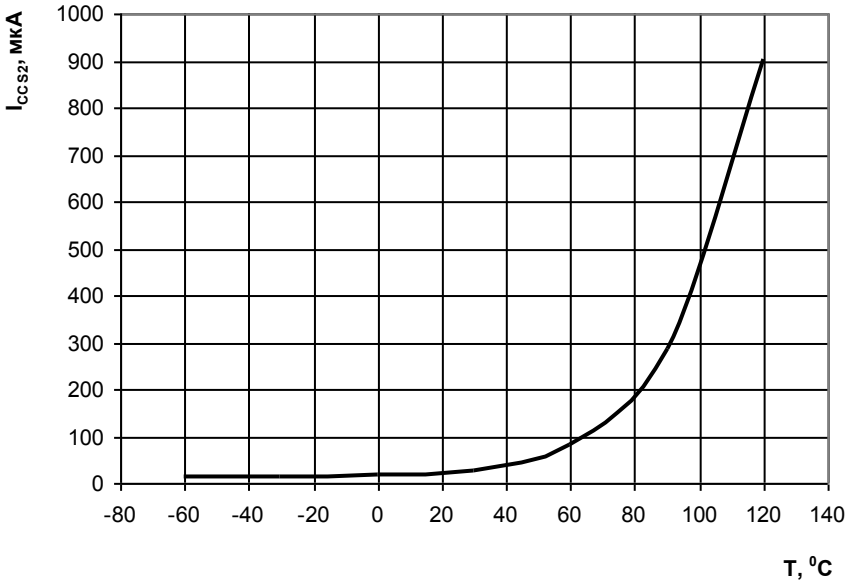


Рисунок 138. Зависимость статического тока потребления в режиме покоя (регулятор напряжения включен) от температуры

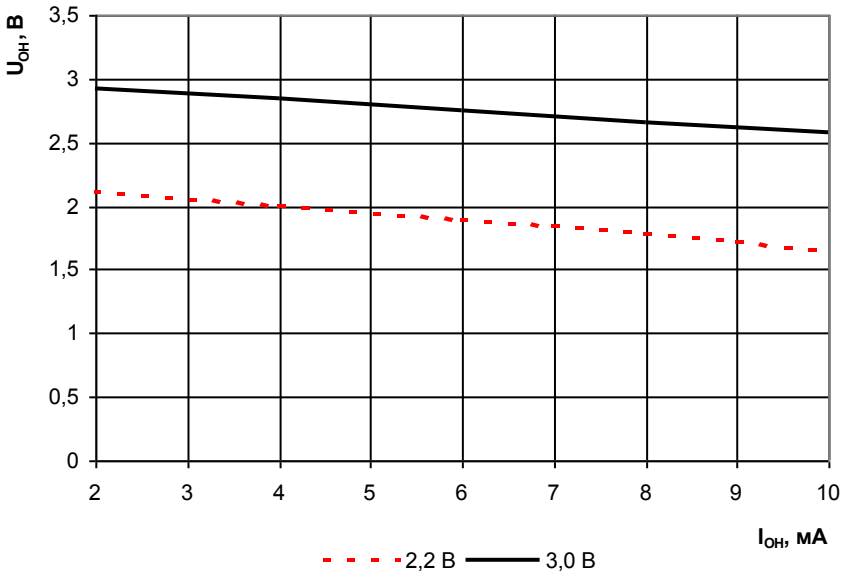


Рисунок 139. Зависимость выходного напряжения высокого уровня от выходного тока высокого уровня при напряжении питания 2,2 и 3,0 В

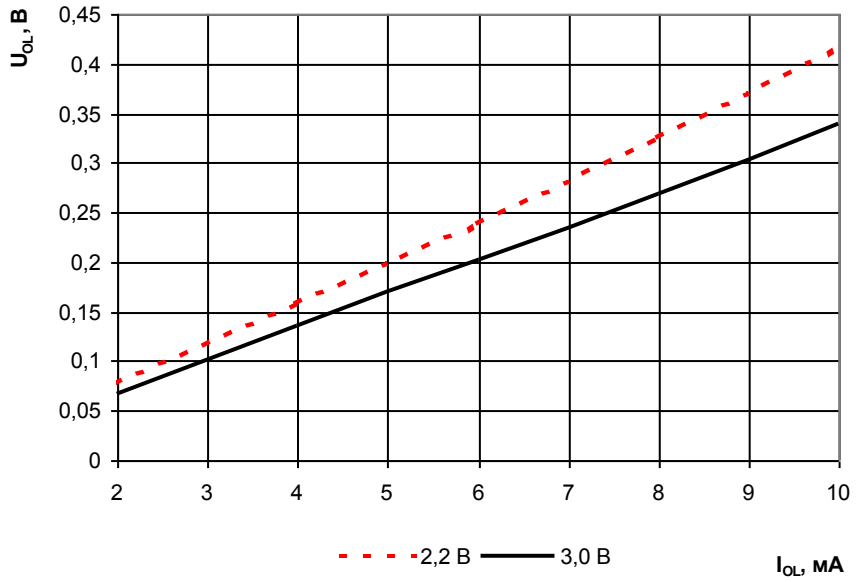
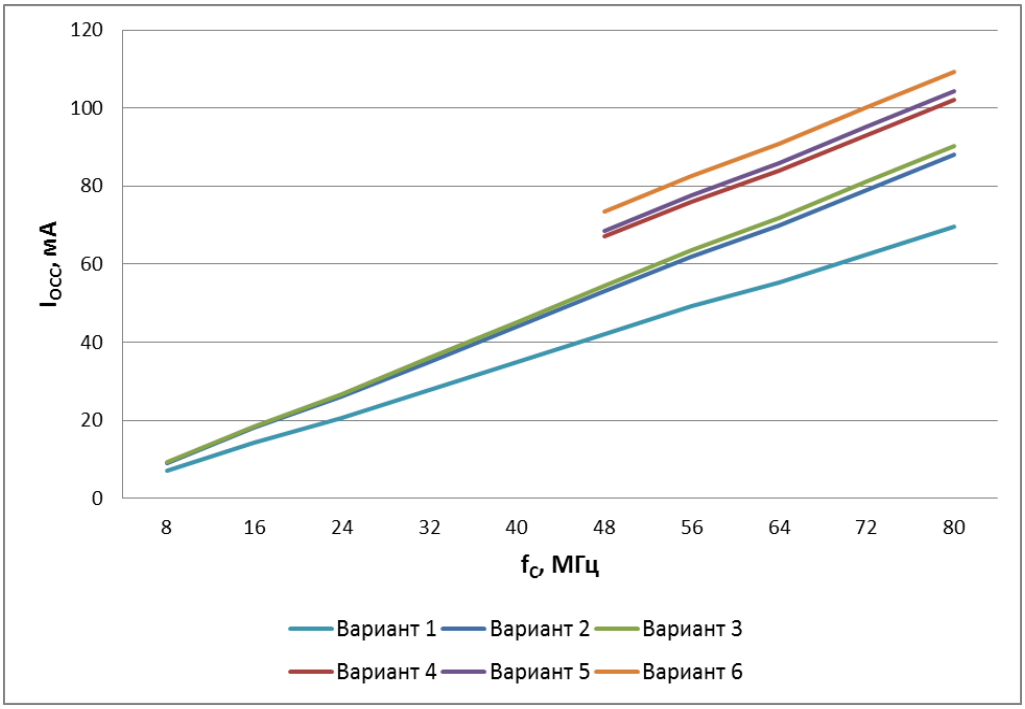


Рисунок 140. Зависимость выходного напряжения низкого уровня от выходного тока низкого уровня при напряжении питания 2,2 и 3,0 В



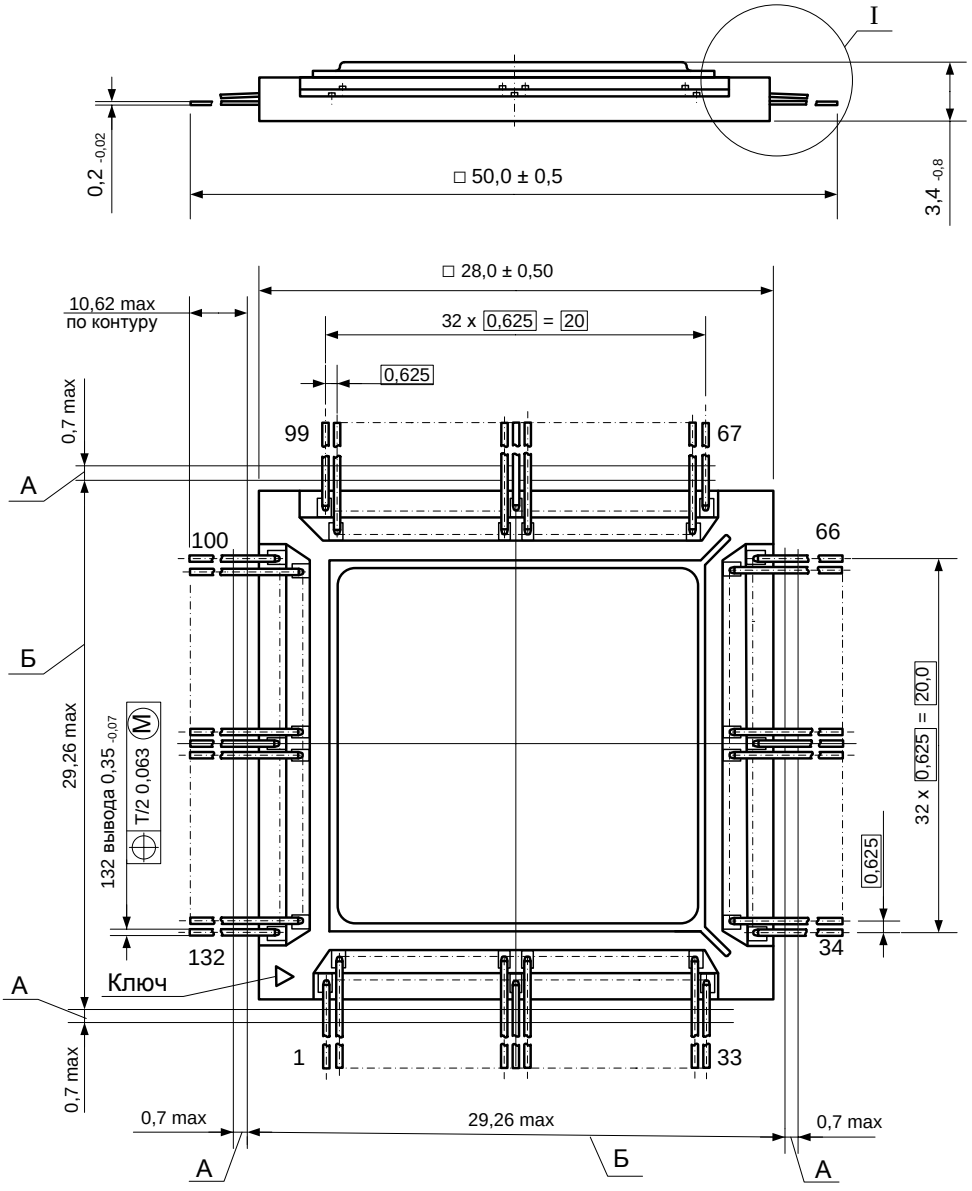
Вариант 1 – процессорное ядро;
Вариант 2 – процессорное ядро + 1xUART+2xSPI+ 3xTIMER;
Вариант 3 – процессорное ядро + 1xUART+2xSPI+ 3xTIMER + 1xCAN;
Вариант 4 – процессорное ядро + 1xUART+2xSPI+ 3xTIMER + USB;
Вариант 5 – процессорное ядро + 1xUART+2xSPI+ 3xTIMER + USB + 1xCAN;
Вариант 6 – процессорное ядро + 1xUART+2xSPI+ 3xTIMER + USB + 1xCAN + 1xADC + 1xDAC.

Рисунок 141. Типовая зависимость тока потребления микросхемы от тактовой частоты, в зависимости от набора используемых периферийных блоков

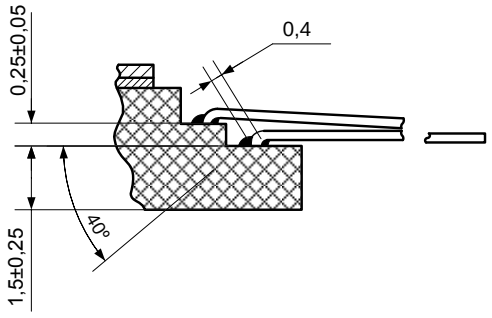
Таблица 491 – Типовые значения потребления основных блоков микросхемы при нормальных условиях: T = 25 °C, Ucc = 3,3 В

Название блока	Потребление блока	Примечание
Процессорное ядро	0,5	мА/МГц
USB	15	мА (фиксированная частота 48 МГц)
TIMER	0,04	мА/МГц
SPI	0,03	мА/МГц
UART	0,05	мА/МГц
CAN	0,03	мА/МГц
I2C	0,02	мА/МГц

38 Габаритный чертеж микросхемы

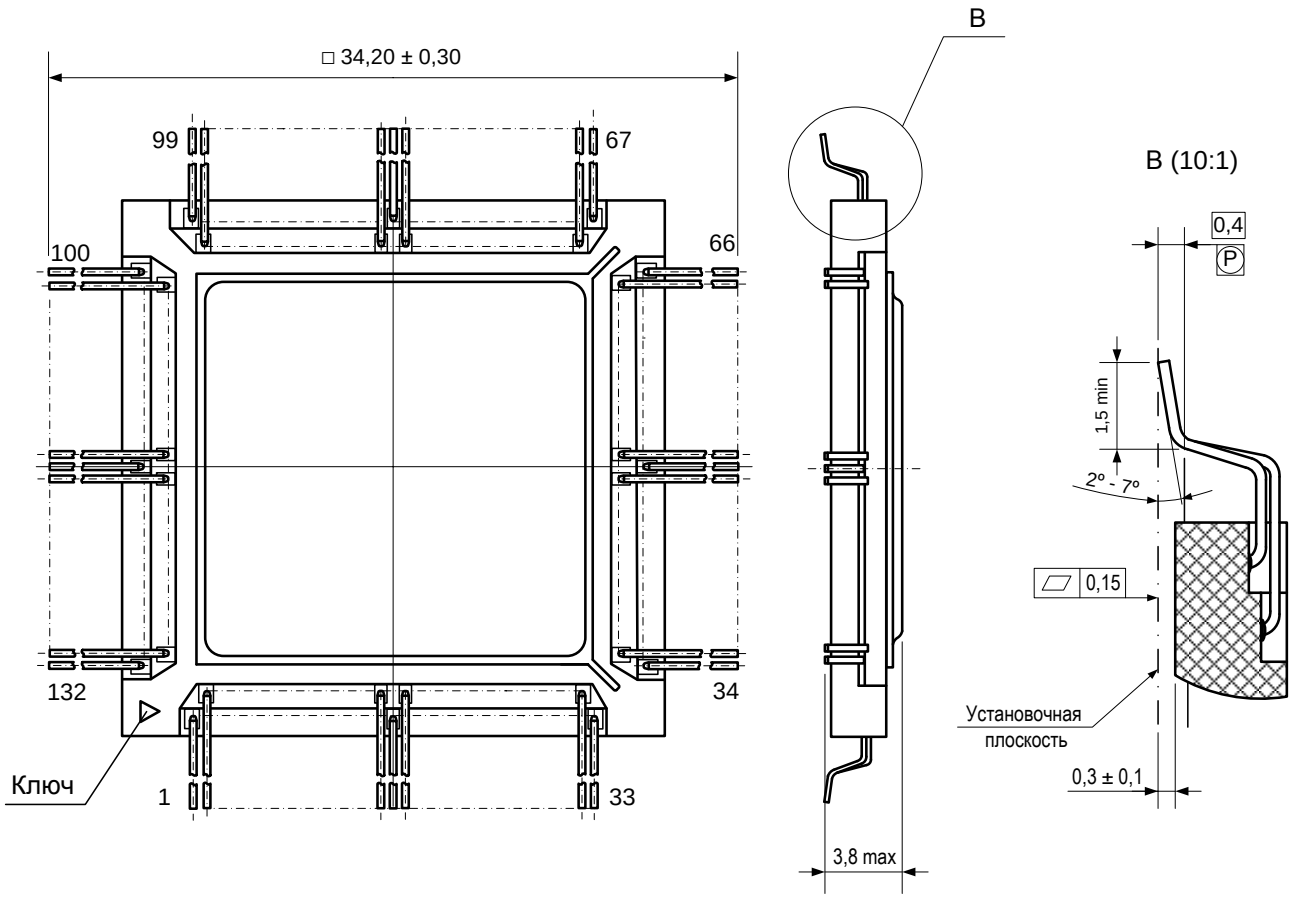


I
(M10:1)



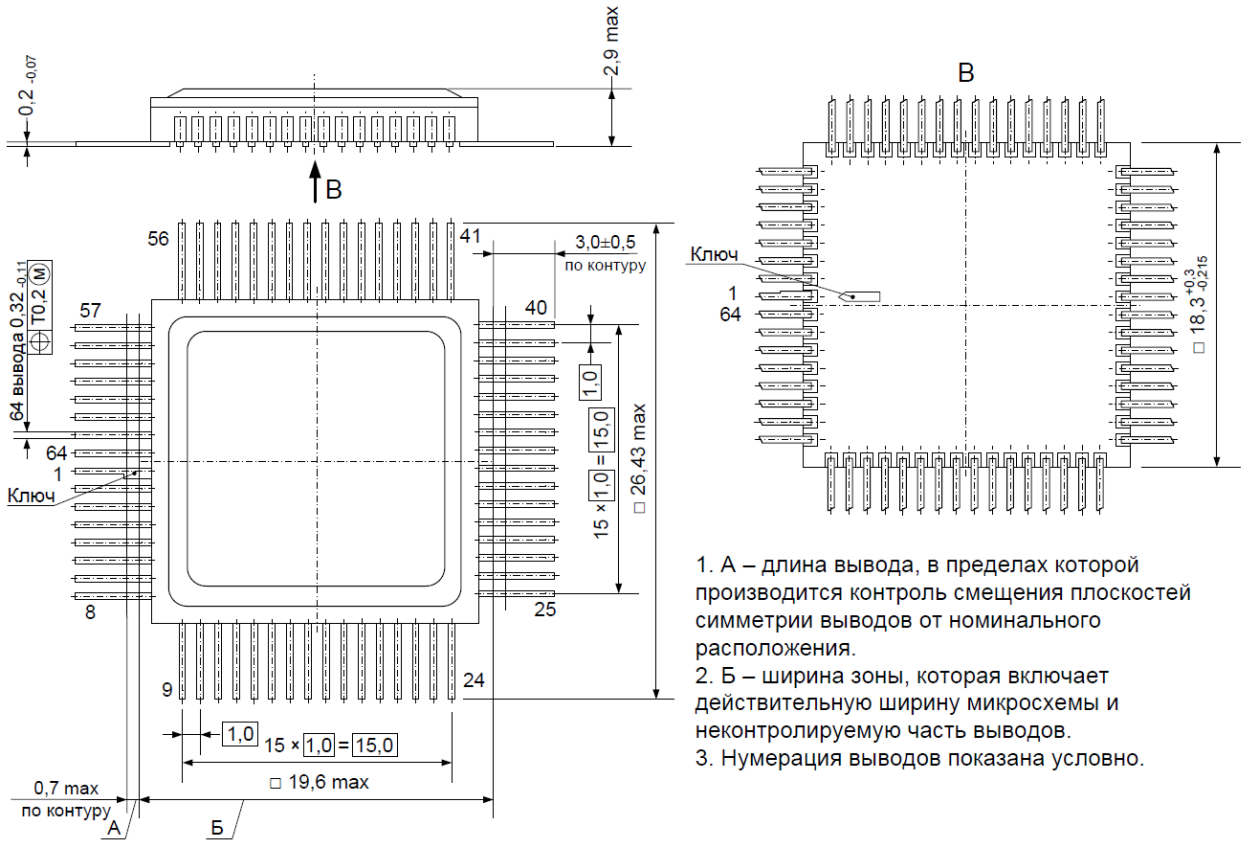
1. А – ширина зоны, которая включает действительную ширину микросхемы и часть выводов, непригодную для монтажа.
2. Б – длина вывода, в пределах которой производится контроль смещения плоскостей симметрии выводов от номинального расположения.
3. Нумерация выводов показана условно.

Рисунок 142. Корпус 4229.132-3



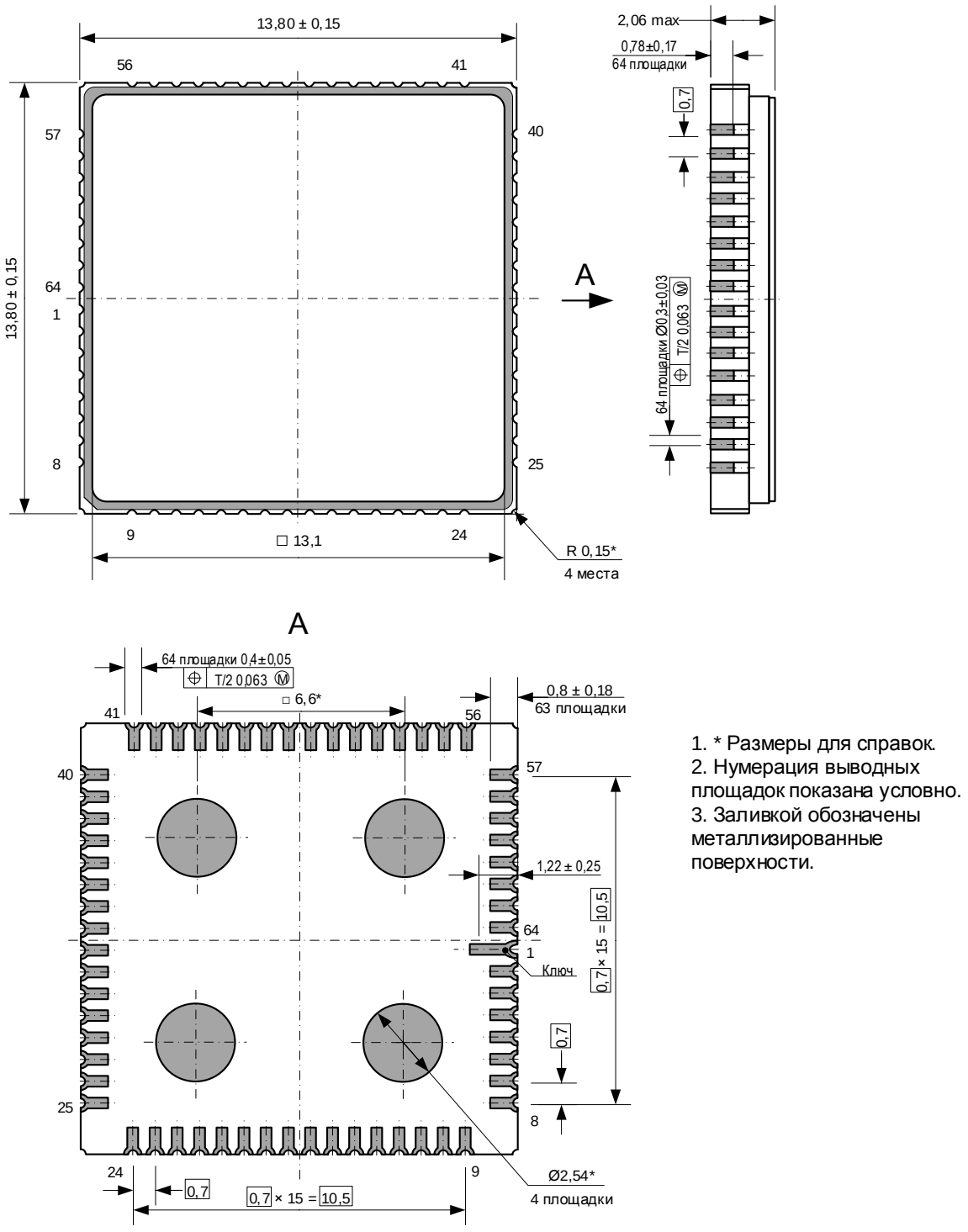
Остальное см. рисунок 142

Рисунок 143. Корпус 4229.132-3 с формованными выводами



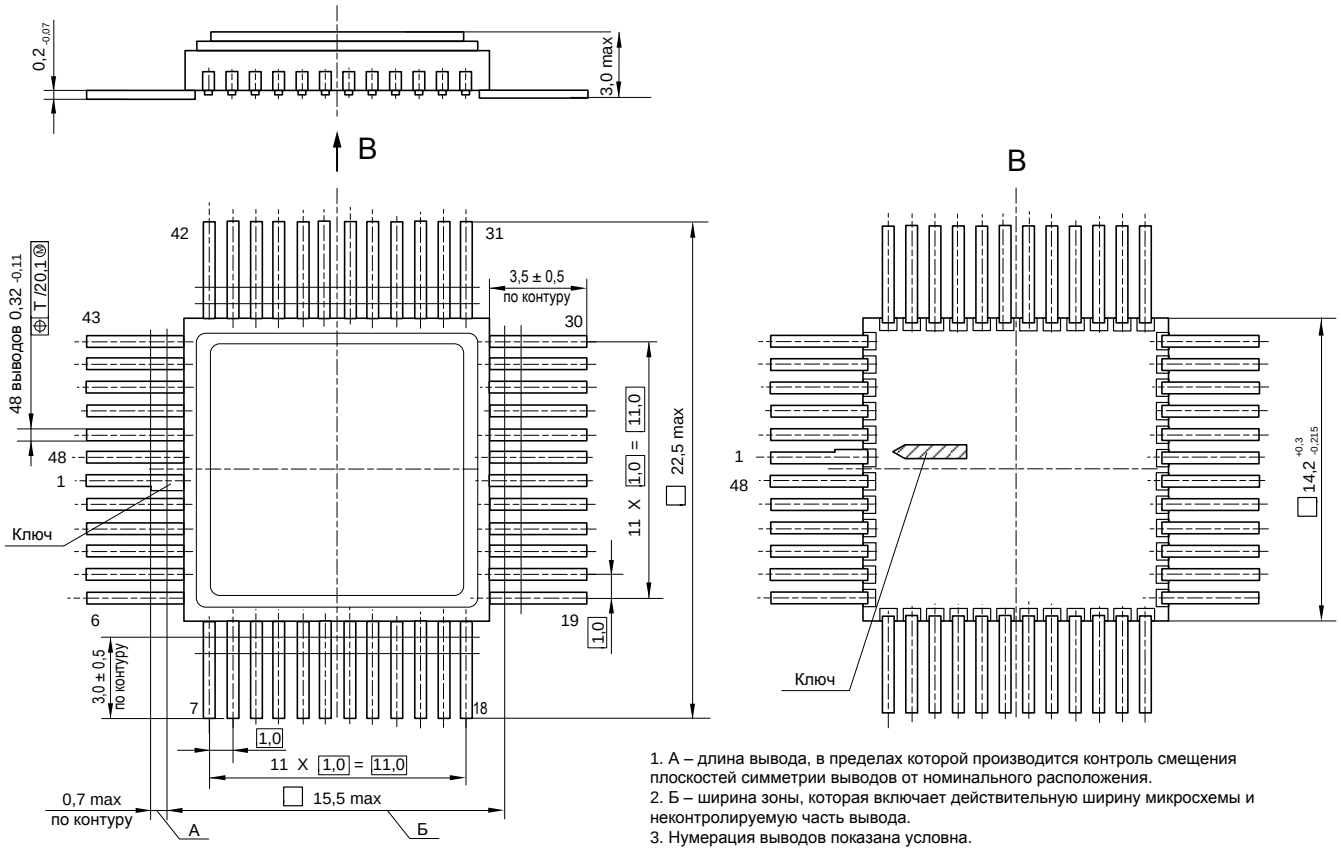
1. А – длина вывода, в пределах которой производится контроль смещения плоскостей симметрии выводов от номинального расположения.
2. Б – ширина зоны, которая включает действительную ширину микросхемы и неконтролируемую часть выводов.
3. Нумерация выводов показана условно.

Рисунок 144. Корпус H18.64-1B



1. * Размеры для справок.
2. Нумерация выводных площадок показана условно.
3. Заливкой обозначены металлизированные поверхности.

Рисунок 145. Корпус МК 5153.64-2



1. А – длина вывода, в пределах которой производится контроль смещения плоскостей симметрии выводов от номинального расположения.
2. Б – ширина зоны, которая включает действительную ширину микросхемы и неконтролируемую часть вывода.
3. Нумерация выводов показана условна.

Рисунок 146. Корпус H16.48-1B

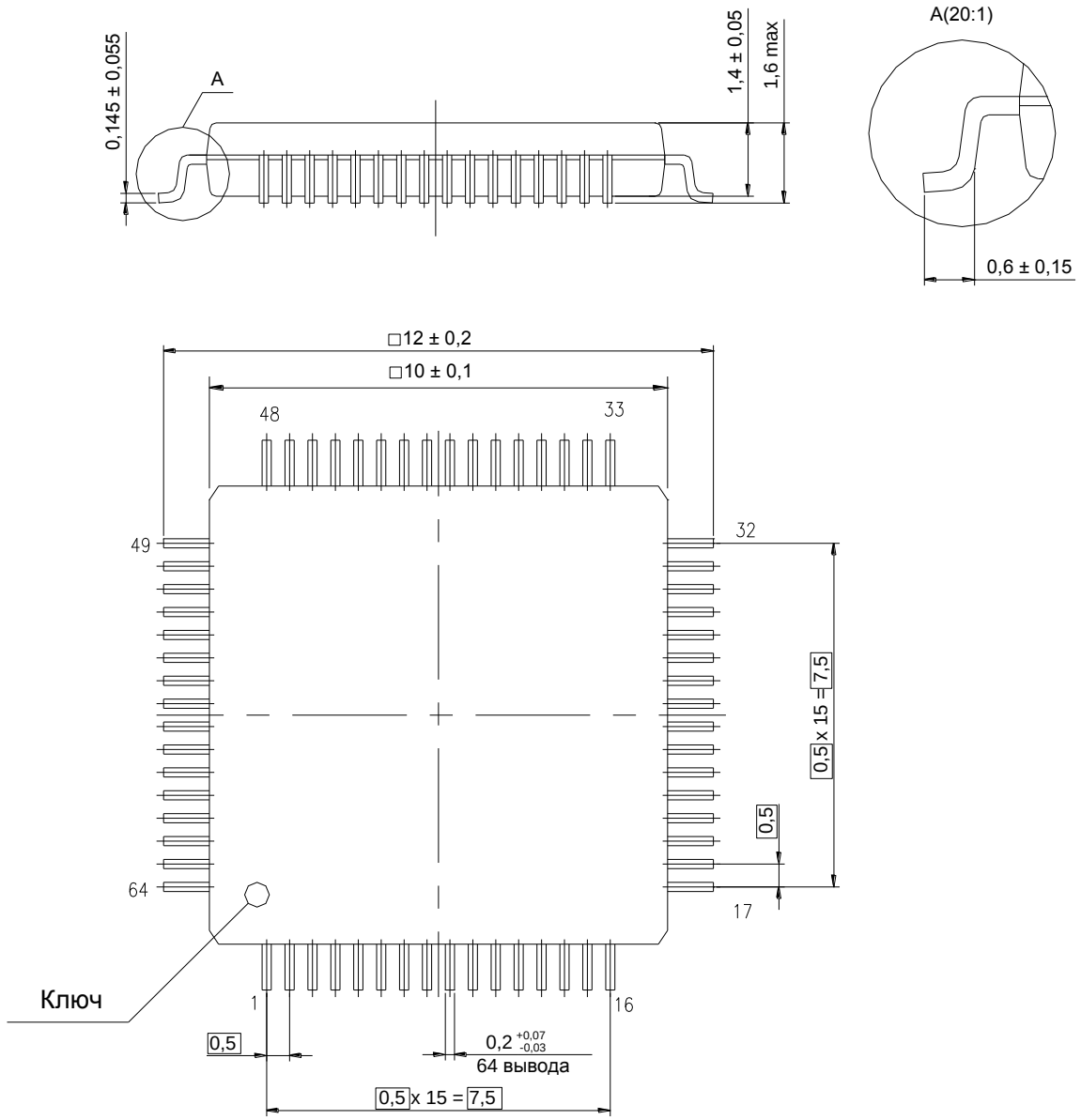


Рисунок 147. Корпус LQFP64

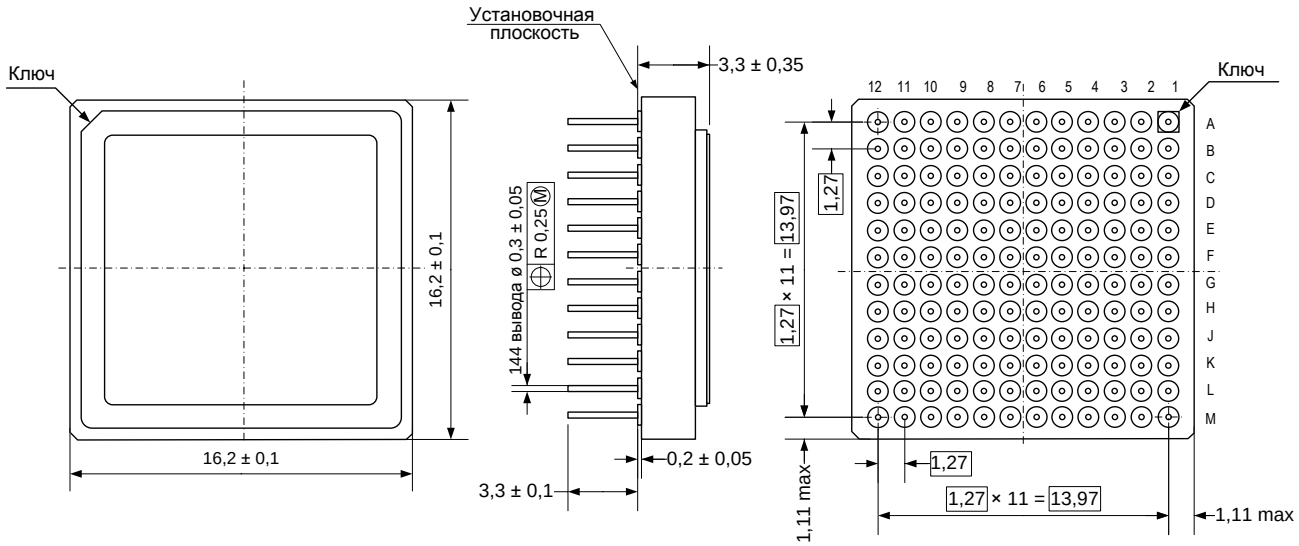


Рисунок 148. Корпус МК 6109.144-А

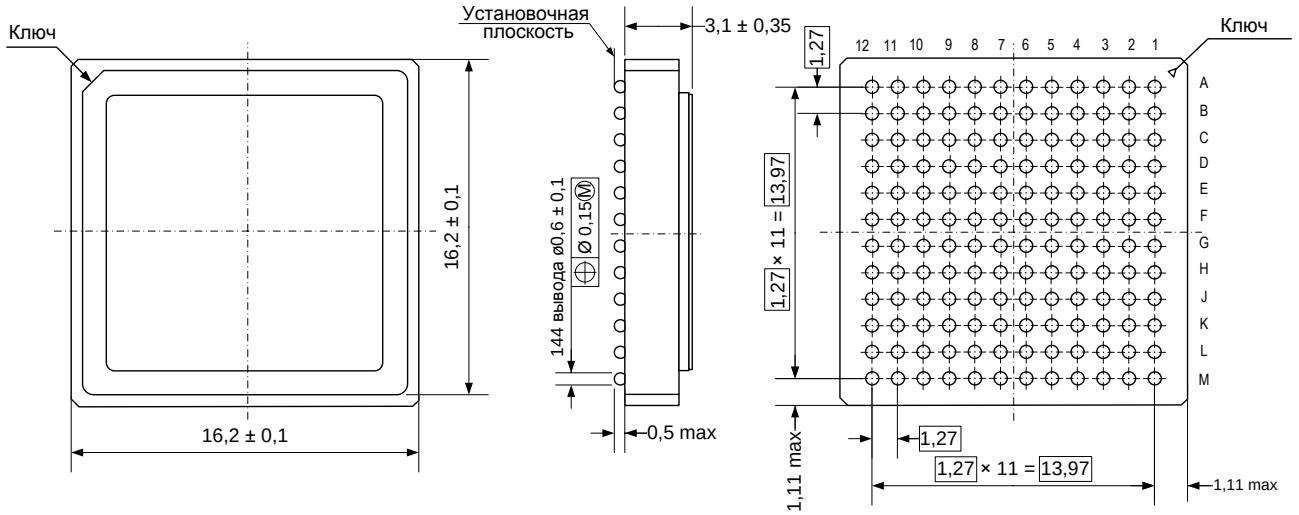
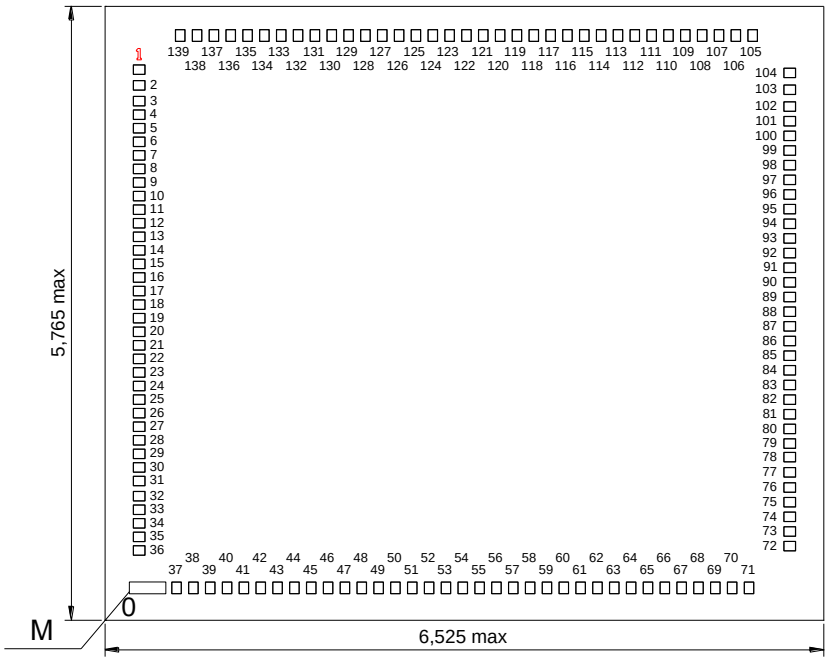


Рисунок 149. Корпус МК 8307.144-АН3



Размеры КП 1 - 36, 72 - 104 (105 x 58) мкм
Размеры КП 37 - 71, 105 - 139 (85 x 105) мкм

1 Номера контактных площадок (КП), кроме первой КП, присвоены условно, их расположение соответствует топологическому чертежу.
2 M - маркировка кристалла, MLDR44.
3 Координаты КП смотри в таблице.
4 Толщина кристалла (0,480±0,015) мм.
5 Материал КП - сплав AlCu толщиной 3 мкм с содержанием Cu 0,5%.

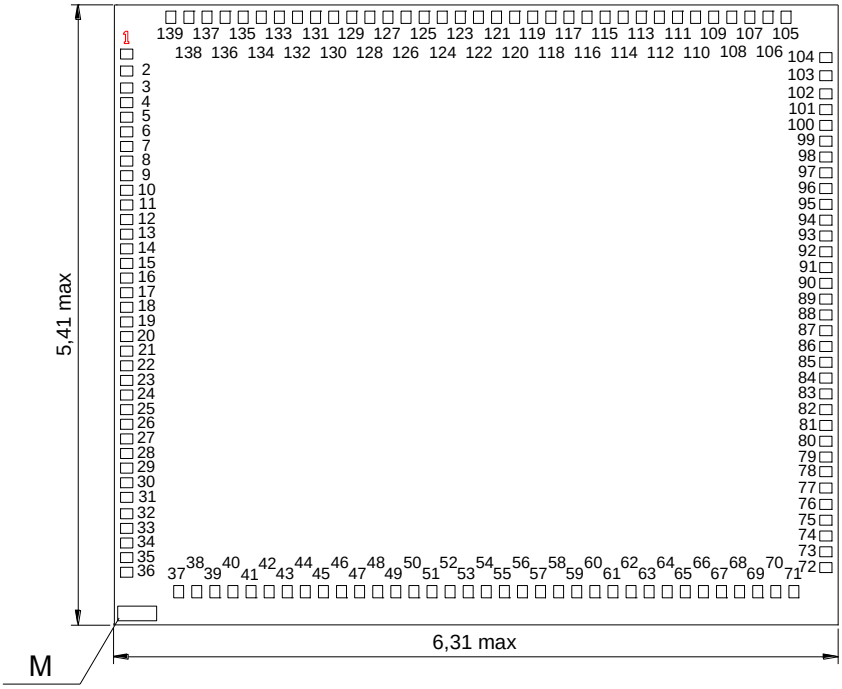
Рисунок 150. Кристалл (бескорпусное исполнение 1986BE91H4, K1986BE91H4)

Таблица 492 – Координаты КП

№ КП	Обозначение КП	Координаты КП		№ КП	Обозначение КП	Координаты КП	
		X	Y			X	Y
1	Ucc (VD33!)	0,000	4774,63	17	PF14	0,000	2734,63
2	Ucc (VD33!)	0,000	4629,63	18	PF15	0,000	2609,63
3	PF0	0,000	4484,63	19	PE15	0,000	2484,63
4	PF1	0,000	4359,63	20	PE13	0,000	2359,63
5	PF2	0,000	4234,63	21	PE12	0,000	2234,63
6	PF3	0,000	4109,63	22	USBD+	0,000	2109,63
7	PF4	0,000	3984,63	23	USBDR+	0,000	1984,63
8	PF5	0,000	3859,63	24	USBDR-	0,000	1859,63
9	PF6	0,000	3734,63	25	USBD-	0,000	1734,63
10	PF7	0,000	3609,63	26	PE11	0,000	1609,63
11	PF8	0,000	3484,63	27	SHDN	0,000	1484,63
12	PF9	0,000	3359,63	28	Ducc (VDD!)	0,000	1359,63
13	PF10	0,000	3234,63	29	GND (VSS!)	0,000	1234,63
14	PF11	0,000	3109,63	30	GND* (VSS!)	0,000	1109,63
15	PF12	0,000	2984,63	31	Ucc (VD33!)	0,000	984,63
16	PF13	0,000	2859,63	32	Ucc (VD33!)	0,000	844,63

№ КП	Обозначение КП	Координаты КП		№ КП	Обозначение КП	Координаты КП	
		X	Y			X	Y
33	BUcc	0,000	719,63	87	PC9	5989,000	2409,50
34	Standby	0,000	594,63	88	PC8	5989,000	2544,50
35	PE7	0,000	469,63	89	PC7	5989,000	2679,50
36	PE6	0,000	344,63	90	PC6	5989,000	2814,50
37	BDUcc	344,130	0,00	91	PC5	5989,000	2949,50
38	WAKEUP	499,130	0,00	92	PC4	5989,000	3084,50
39	EXT_POR	654,130	0,00	93	PC3	5989,000	3219,50
40	RESET	809,130	0,00	94	PC2	5989,000	3354,50
41	OSC_IN	964,130	0,00	95	PC1	5989,000	3489,50
42	OSC_OUT	1119,130	0,00	96	PC0	5989,000	3624,50
43	PE14	1274,130	0,00	97	PB0	5989,000	3759,50
44	PE5	1429,130	0,00	98	PB1	5989,000	3894,50
45	PE4	1584,130	0,00	99	PB2	5989,000	4029,50
46	PE8	1739,130	0,00	100	PB3	5989,000	4164,50
47	PE3	1894,130	0,00	101	PB4	5989,000	4299,50
48	PE2	2049,130	0,00	102	JTAG_EN	5989,000	4434,50
49	AGND (VSS!)	2204,130	0,00	103	Ucc (VD33!)	5989,000	4589,50
50	AGND (VSS!)	2359,130	0,00	104	Ucc (VD33!)	5989,000	4744,50
51	AUcc1	2514,130	0,00	105	GND (VSS!)	5647,945	5089,00
52	AUcc1	2669,130	0,00	106	Ducc (VDD!)	5492,945	5089,00
53	PE10	2824,130	0,00	107	PB5	5337,945	5089,00
54	PE9	2979,130	0,00	108	PB6	5182,945	5089,00
55	PE1	3134,130	0,00	109	PB7	5027,945	5089,00
56	PE0	3289,130	0,00	110	PB8	4872,945	5089,00
57	AGND (VSS!)	3444,130	0,00	111	PB9	4717,945	5089,00
58	AGND (VSS!)	3599,130	0,00	112	PB10	4562,945	5089,00
59	AUcc	3754,130	0,00	113	PB11	4407,945	5089,00
60	AUcc	3909,130	0,00	114	PB12	4252,945	5089,00
61	PD15	4064,130	0,00	115	PB13	4097,945	5089,00
62	PD14	4219,130	0,00	116	PB14	3942,945	5089,00
63	PD13	4374,130	0,00	117	TM[0]	3787,945	5089,00
64	PD12	4529,130	0,00	118	TM[1]	3632,945	5089,00
65	PD11	4684,130	0,00	119	PB15	3477,945	5089,00
66	PD10	4839,130	0,00	120	PA15	3322,945	5089,00
67	PD8	4994,130	0,00	121	PA14	3167,945	5089,00
68	PD7	5149,130	0,00	122	PA13	3012,945	5089,00
69	PD4	5304,130	0,00	123	PA12	2857,945	5089,00
70	PD0	5459,130	0,00	124	PA11	2702,945	5089,00
71	PD1	5614,130	0,00	125	PA10	2547,945	5089,00
72	PD2	5989,000	384,50	126	TM[2]	2392,945	5089,00
73	PD3	5989,000	519,50	127	Flash VPP	2237,945	5089,00
74	PD5	5989,000	654,50	128	PA9	2082,945	5089,00
75	PD6	5989,000	789,50	129	PA8	1927,945	5089,00
76	PD9	5989,000	924,50	130	PA7	1772,945	5089,00
77	Ucc (VD33!)	5989,000	1064,50	131	PA6	1617,945	5089,00
78	Ucc (VD33!)	5989,000	1204,50	132	PA5	1462,945	5089,00
79	GND (VSS!)	5989,000	1329,50	133	PA4	1307,945	5089,00
80	Ducc (VDD!)	5989,000	1464,50	134	PA3	1152,945	5089,00
81	PC15	5989,000	1599,50	135	PA2	997,945	5089,00
82	PC14	5989,000	1734,50	136	PA1	842,945	5089,00
83	PC13	5989,000	1869,50	137	PA0	687,945	5089,00
84	PC12	5989,000	2004,50	138	Ducc (VDD!)	532,945	5089,00
85	PC11	5989,000	2139,50	139	GND (VSS!)	377,945	5089,00
86	PC10	5989,000	2274,50				

Примечание – Указаны координаты левых нижних углов контактных площадок относительно точки пересечения осей.



Размеры КП 85 x 105 мкм^2

1 Номера контактных площадок (КП) присвоены условно, кроме первой КП и их расположение соответствует топологическому чертежу.
2 M - маркировка кристалла, MLDR75.

Рисунок 151. Кристалл (бескорпусное исполнение 1986BE94H4, К1986BE94H4)

Таблица 493 – Координаты КП

№ КП	Обозначение КП	Координаты КП		№ КП	Обозначение КП	Координаты КП	
		X	Y			X	Y
1	Ucc (VD33!)	0,000	4774,63	24	USBDR-	0,000	1859,63
2	Ucc (VD33!)	0,000	4629,63	25	USBD-	0,000	1734,63
3	PF0	0,000	4484,63	26	PE11	0,000	1609,63
4	PF1	0,000	4359,63	27	SHDN	0,000	1484,63
5	PF2	0,000	4234,63	28	Ducc (VDD!)	0,000	1359,63
6	PF3	0,000	4109,63	29	GND (VSS!)	0,000	1234,63
7	PF4	0,000	3984,63	30	GND* (VSS!)	0,000	1109,63
8	PF5	0,000	3859,63	31	Ucc (VD33!)	0,000	984,63
9	PF6	0,000	3734,63	32	Ucc (VD33!)	0,000	844,63
10	PF7	0,000	3609,63	33	BUcc	0,000	719,63
11	PF8	0,000	3484,63	34	Standby	0,000	594,63
12	PF9	0,000	3359,63	35	PE7	0,000	469,63
13	PF10	0,000	3234,63	36	PE6	0,000	344,63
14	PF11	0,000	3109,63	37	BDUcc	344,130	0,00
15	PF12	0,000	2984,63	38	WAKEUP	499,130	0,00
16	PF13	0,000	2859,63	39	EXT_POR	654,130	0,00
17	PF14	0,000	2734,63	40	RESET	809,130	0,00
18	PF15	0,000	2609,63	41	OSC_IN	964,130	0,00
19	PE15	0,000	2484,63	42	OSC_OUT	1119,130	0,00
20	PE13	0,000	2359,63	43	PE14	1274,130	0,00
21	PE12	0,000	2234,63	44	PE5	1429,130	0,00
22	USBD+	0,000	2109,63	45	PE4	1584,130	0,00
23	USBDR+	0,000	1984,63	46	PE8	1739,130	0,00

№ КП	Обозначение КП	Координаты КП		№ КП	Обозначение КП	Координаты КП	
		X	Y			X	Y
47	PE3	1894,130	0,00	94	PC2	5989,000	3354,50
48	PE2	2049,130	0,00	95	PC1	5989,000	3489,50
49	AGND (VSS!)	2204,130	0,00	96	PC0	5989,000	3624,50
50	AGND (VSS!)	2359,130	0,00	97	PB0	5989,000	3759,50
51	AUcc1	2514,130	0,00	98	PB1	5989,000	3894,50
52	AUcc1	2669,130	0,00	99	PB2	5989,000	4029,50
53	PE10	2824,130	0,00	100	PB3	5989,000	4164,50
54	PE9	2979,130	0,00	101	PB4	5989,000	4299,50
55	PE1	3134,130	0,00	102	JTAG_EN	5989,000	4434,50
56	PE0	3289,130	0,00	103	Ucc (VD33!)	5989,000	4589,50
57	AGND (VSS!)	3444,130	0,00	104	Ucc (VD33!)	5989,000	4744,50
58	AGND (VSS!)	3599,130	0,00	105	GND (VSS!)	5647,945	5089,00
59	AUcc	3754,130	0,00	106	Ducc (VDD!)	5492,945	5089,00
60	AUcc	3909,130	0,00	107	PB5	5337,945	5089,00
61	PD15	4064,130	0,00	108	PB6	5182,945	5089,00
62	PD14	4219,130	0,00	109	PB7	5027,945	5089,00
63	PD13	4374,130	0,00	110	PB8	4872,945	5089,00
64	PD12	4529,130	0,00	111	PB9	4717,945	5089,00
65	PD11	4684,130	0,00	112	PB10	4562,945	5089,00
66	PD10	4839,130	0,00	113	PB11	4407,945	5089,00
67	PD8	4994,130	0,00	114	PB12	4252,945	5089,00
68	PD7	5149,130	0,00	115	PB13	4097,945	5089,00
69	PD4	5304,130	0,00	116	PB14	3942,945	5089,00
70	PD0	5459,130	0,00	117	TM[0]	3787,945	5089,00
71	PD1	5614,130	0,00	118	TM[1]	3632,945	5089,00
72	PD2	5989,000	384,50	119	PB15	3477,945	5089,00
73	PD3	5989,000	519,50	120	PA15	3322,945	5089,00
74	PD5	5989,000	654,50	121	PA14	3167,945	5089,00
75	PD6	5989,000	789,50	122	PA13	3012,945	5089,00
76	PD9	5989,000	924,50	123	PA12	2857,945	5089,00
77	Ucc (VD33!)	5989,000	1064,50	124	PA11	2702,945	5089,00
78	Ucc (VD33!)	5989,000	1204,50	125	PA10	2547,945	5089,00
79	GND (VSS!)	5989,000	1329,50	126	TM[2]	2392,945	5089,00
80	Ducc (VDD!)	5989,000	1464,50	127	Flash VPP	2237,945	5089,00
81	PC15	5989,000	1599,50	128	PA9	2082,945	5089,00
82	PC14	5989,000	1734,50	129	PA8	1927,945	5089,00
83	PC13	5989,000	1869,50	130	PA7	1772,945	5089,00
84	PC12	5989,000	2004,50	131	PA6	1617,945	5089,00
85	PC11	5989,000	2139,50	132	PA5	1462,945	5089,00
86	PC10	5989,000	2274,50	133	PA4	1307,945	5089,00
87	PC9	5989,000	2409,50	134	PA3	1152,945	5089,00
88	PC8	5989,000	2544,50	135	PA2	997,945	5089,00
89	PC7	5989,000	2679,50	136	PA1	842,945	5089,00
90	PC6	5989,000	2814,50	137	PA0	687,945	5089,00
91	PC5	5989,000	2949,50	138	Ducc (VDD!)	532,945	5089,00
92	PC4	5989,000	3084,50	139	GND (VSS!)	377,945	5089,00
93	PC3	5989,000	3219,50				

Примечание – Указаны координаты левых нижних углов контактных площадок относительно точки пересечения осей.

39 Информация для заказа

Обозначение	Маркировка	Тип корпуса	Температурный диапазон
1986BE91T	1986BE91T	4229.132-3	минус 60 – 125 °С
K1986BE91T	K1986BE91T	4229.132-3	минус 60 – 125 °С
K1986BE91TK	K1986BE91T•	4229.132-3	0 – 70 °С
K1986BE91ГТК	K1986BE91T•-1	4229.132-3	0 – 70 °С
K1986BE91ДТК	K1986BE91T•-2	4229.132-3	0 – 70 °С
1986BE92У	1986BE92У	Н.18.64-1В	минус 60 – 125 °С
K1986BE92У	K1986BE92У	Н.18.64-1В	минус 60 – 125 °С
K1986BE92УК	K1986BE92У•	Н.18.64-1В	0 – 70 °С
1986BE92У1	1986BE92У1	МК 5153.64-2	минус 60 – 125 °С
K1986BE92У1	K1986BE92У1	МК 5153.64-2	минус 60 – 125 °С
K1986BE92У1К	K1986BE92У1•	МК 5153.64-2	0 – 70 °С
K1986BE92QC	MDR32F9Q2C ARM	LQFP64	0 – 70 °С
K1986BE92QI	MDR32F9Q2I ARM	LQFP64	минус 40 – 85 °С
K1986BE92ГУК	K1986BE92У•-1	Н.18.64-1В	0 – 70 °С
K1986BE92ДУК	K1986BE92У•-2	Н.18.64-1В	0 – 70 °С
1986BE93У	1986BE93У	Н.16.48-1В	минус 60 – 125 °С
K1986BE93У	K1986BE93У	Н.16.48-1В	минус 60 – 125 °С
K1986BE93УК	K1986BE93У•	Н.16.48-1В	0 – 70 °С
K1986BE93ГУК	K1986BE93У•-1	Н.16.48-1В	0 – 70 °С
K1986BE93ДУК	K1986BE93У•-2	Н.16.48-1В	0 – 70 °С
1986BE94T	1986BE94T	4229.132-3	минус 60 – 125 °С
K1986BE94T	K1986BE94T	4229.132-3	минус 60 – 125 °С
K1986BE94TK	K1986BE94T•	4229.132-3	0 – 70 °С
1986BE94Ф	1986BE94Ф	МК 6109.144-А	минус 60 – 125 °С
K1986BE94Ф	K1986BE94Ф	МК 6109.144-А	минус 60 – 125 °С
K1986BE94ФК	K1986BE94Ф•	МК 6109.144-А	0 – 70 °С
1986BE94Я	1986BE94Я	МК 8307.144-АН3	минус 60 – 125 °С
K1986BE94Я	K1986BE94Я	МК 8307.144-АН3	минус 60 – 125 °С
K1986BE94ЯК	K1986BE94Я•	МК 8307.144-АН3	0 – 70 °С

Примечания:

1. Микросхемы в бескорпусном исполнении поставляются в виде отдельных кристаллов, получаемых разделением пластины. Микросхемы поставляются в таре (кейсах) без потери ориентации. Маркировка микросхем 1986BE91H4, K1986BE91H4, 1986BE94H4, K1986BE94H4 наносится на тару.
2. Необходимость поставки микросхем с формованными выводами указывается в договоре на поставку.

Микросхемы с приемкой «ВП» маркируются ромбом.
Микросхемы с приемкой «ОТК» маркируются буквой «К».

40 Лист регистрации изменений

№ п/п	Дата	Версия	Краткое содержание изменения	№№ изменяемых листов
1	14.10.2010	1.1	1. Приведение в соответствие с ТУ таблиц параметров 2. Внесение зависимостей	
2	18.02.2011	2.0	1. Исправлены опечатки в тексте 2. Введены группы исполнения Г и Д	
3	24.06.2011	2.1	Введение корпуса LQFP64 для ИМС 1986BE92У	
4	05.08.2011	2.3	Приведены названия регистров в соответствие с CMSIS, исправлены опечатки в тексте, расширены описания АЦП, неточности в CAN, уточнены значения после сброса в регистрах.	
5	10.08.2011	2.4	Наименования приведены в соответствие с системой предприятия. Отредактировано описание работы микросхемы.	
6	25.08.2011	2.5	Редактирование текста	
7	28.09.2011	3.0	Редактирование текста	
8	14.06.2012	3.1.0	Введена микросхема в бескорпусном исполнении	По тексту
9	20.09.2012	3.2.0	Исправлена ошибка наименования м/схемы К…Н4	1
10	20.12.2012	3.2.1	Исправлена ошибка наименования м/схемы К…Н4	8
11	15.05.2013	3.2.2	Исправлена ошибка: номер контактной площадки кристалла 66	11
12	21.06.2013	3.3.0	Введена ИМС 1986BE94T	По тексту
13	26.06.2013	3.4.0	Исправление названия и маркировки микросхем	По тексту
14	17.07.2013	3.4.1	Внесены изменения в описание	По тексту
15	22.07.2013	3.4.2	Приведение в соответствие с ТУ Табл. параметров	525-532
16	22.07.2013	3.4.3	Исправлены адреса регистров SysTick, уточнены знаки выходных токов высокого и низкого уровней, добавлены примечания по рабочему диапазону температур	142,525, 527-532
17	12.05.2014	3.4.4	Добавлены номера КП кристалла в таблице 2	12
18	07.07.2014	3.5.0	Исправление надписи на рис.61. Уточнение в описании работы счетчика ошибок	258
19	07.08.2014	3.6.0	Исправлен рисунок	1
20	20.11.2014	3.7.0	Исправлено значение параметра в таблице 487	530
21	11.03.2015	3.7.1	1. Исправление ошибок, опечаток 2. Исправления в таблице 144	По тексту 191
22	08.09.2015	3.8.0	Введение бескорпусных исполнений 1986BE91H4, 1986BE94H4, K1986BE94H4	По тексту
23	06.11.2015	3.9.0	Добавлена зависимость тока потребления микросхемы от тактовой частоты Заменен рисунок 100 Удалены фразы с бит/флаг SCTDONE Внесены исправления в таблице 270 Исправлены опечатки в таблицах 142, 150	510, 511 337 По тексту 289 188, 197

			Внесены исправления в таблицу 109 Исправление опечаток Внесено дополнение в раздел Порты ввода-вывода MDR_PORTx Исправление замечаний	172, 173 По тексту 177 По тексту
24	04.10.2016	3.10.0	Плановая корректировка по замечаниям	По тексту
25	24.07.2018	3.11.0	Добавлена микросхема в корпусе с формованными выводами	519, 527
26	30.08.2018	3.12.0	Внесение изменений по замечаниям главного конструктора. Плановая корректировка по замечаниям пользователей	17, 23, 214, 228, 270,279, 291, 306, 307, 314, 485, 521
27	21.09.2018	3.13.0	Введение исполнений 1986BE92У1, 1986BE94Ф, 1986BE94Я	По тексту
28	12.10.2018	3.14.0	Дополнена таблица описания выводов микросхем в корпусах МК 6109.144-А, МК 8307.144-АН3	25, 26
29	01.03.2019	3.15.0	Внесение уточнения в Рисунок 145 Исправление опечаток Внесение изменений в Таблицу 23	523 252. 275 62
30	06.03.2019	3.16.0	Исправлена ошибка нумерации выводов PE4, PE5, PE6, PE7 в корпусе МК 5153.64-2	11
31	16.04.2019	3.17.0	В таблицы 281, 282 добавлено описание бита EV_DELAY регистра CHy_CNTRL2. Исправлен габаритный чертеж корпуса Н18.64-1В	306 522
32	10.09.2019	3.18.0	Исправление ошибок, опечаток Исправлено описание задания режимов работы АЦП Добавлено примечание для бита 6 таблицы 369 Изменен Рисунок 28 Исправление названия битов 30, 31 в таблице 397	По тексту 319 402 165 438
33	16.09.2019	3.19.0	Исправлен габаритный чертеж корпуса Н16.48-1В	524