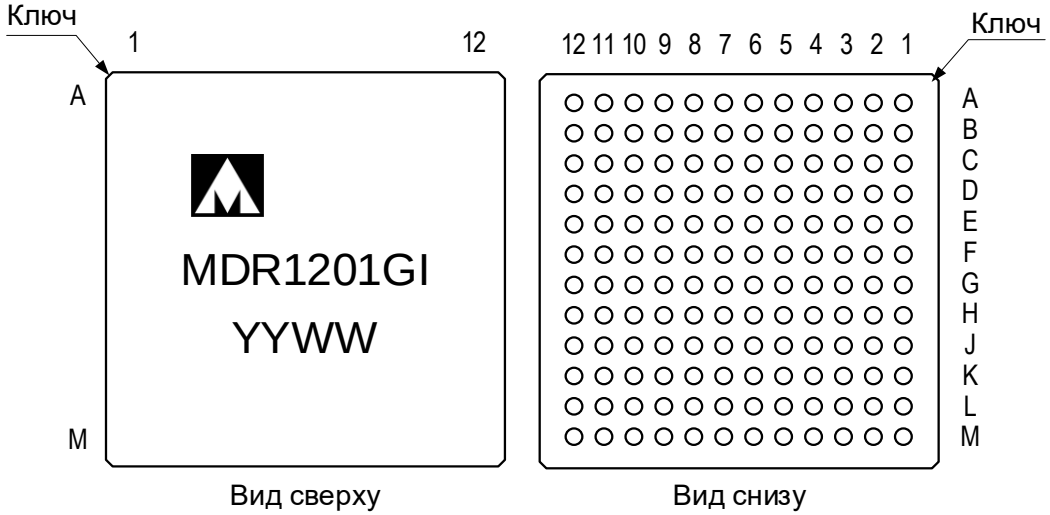
# Микросхема 32-разрядного микроконтроллера К1986ВК01GI




YY – год выпуска
WW – неделя выпуска

## Основные параметры микросхемы:

- Напряжение источника питания от 3,0 до 3,6 В;
- 32-разрядная RISC архитектура ARM Cortex-M4F;
- Два вычислительных ядра;
- Энергонезависимая, перепрограммируемая встроенная память программ 1 Мбайт;
- Встроенная память программ типа СОЗУ 128 Кбайт;
- Встроенная память данных типа СОЗУ 128 Кбайт;
- Температурный диапазон: от минус 40 °С до 85 °С.

## Тип корпуса:

- 144-выводной пластмассовый корпус BGA144.

## Общее описание и области применения микросхемы

Микросхемы интегральные К1986ВК01GI (далее – микросхемы) представляют собой микроконтроллер управления двигателями на базе процессорных ядер ARM Cortex-M4F. Микросхемы предназначены для измерения, контроля, управления и диагностики.

**Важно:** спецификация действительна совместно с документом К1986ВК01GI Errata Notice.

## Основные характеристики

***Ядра:***

- два ядра ARM Cortex-M4F;
- максимальная тактовая частота ядер ARM Cortex-M4F - до 160 МГц;
- поддержка на уровне ядра операций ЦОС;
- аппаратное выполнение операций с плавающей запятой;
- блок аппаратной защиты регионов памяти;
- блок отладки с поддержкой трассы инструкций ETM и интерфейсами JTAG и SWD.
- Ядро ARM Cortex-M0;
- максимальная тактовая частота ядра ARM Cortex-M0 - до 130 МГц;
- блок отладки с интерфейсом SWD.

***Питание:***

- основное питание от 3,0 до 3,6 В;
- встроенный импульсный преобразователь питания цифровой части;
- батарейный домен с автоматическим переключением на питание от батарейки;
- встроенные регуляторы питания батарейного домена;
- аппаратный детектор снижения и превышения допустимого уровня питания;
- изолированные от помех питания для АЦП и ЦАП.

***Тактовые частоты:***

- встроенные высоконадежные RC-генераторы HSI (~8 МГц) и LSI (~40 кГц);
- внешние генераторы HSE0 и HSE1 от 2 до 30 МГц в режиме резонатора;
- внешний часовой генератор LSE в батарейном домене;
- четыре блока PLL умножения тактовых частот;
- блоки аппаратной защиты от снижения/увеличения частоты тактирования.

***Память (подсистема Cortex-M4F):***

- 256 Кбайт ОЗУ с ECC (SEC-DED);
- 1 Мбайт памяти программ с ECC (SEC-DED);
- контроллер внешней системной шины с последовательной/параллельной организацией с ECC (SEC-DED).

***Память (подсистема Cortex-M0):***

- 64 Кбайт OTP ПЗУ;
- 128 Кбайт ОЗУ;
- 8 Кбайт ПЗУ;
- 2 Кбайт память ключей.

***Периферийные блоки 2xCortex-M4F:***

- контроллер EthernetMAC 10/100 Мбит/с;
- два контроллера МКПД в режимах КШ, ОУ, Монитор;
- контроллеры интерфейсов: 2xCAN 2.0, CAN FD, 2xSSP, 1xI2C, 4xUART, 1xUSB HS (встроенный USB PHY до 480Мбит/с);
- два блока контроллера DMA;
- часы реального времени RTC;
- сторожевой таймер WDG;
- четыре таймера общего назначения с функцией ШИМ;
- до 96-ти выводов портов общего назначения;
- три контроллера АЦП;

- 6 аналоговых 12-разрядных АЦП. Суммарно 25 независимых положительных каналов для работы в однополярном режиме (single-ended) и 7 независимых отрицательных каналов для работы в дифференциальном режиме (differential);
- 12-разрядный ЦАП;
- девять независимых блоков ШИМ;
- четыре модуля захвата с функциями ШИМ;
- блок аппаратных вычислений тригонометрических функций;
- четыре аналоговых компаратора;
- четыре цифро-аналоговых преобразователя;
- два модуля квадратурных декодеров eQEP.

***Периферийные блоки Cortex-M0:***

- UART ISO 7816;
- блок длинночисленной арифметики;
- четыре блока P-bit;
- блок P-byte;
- восемь блоков S-block;
- блок L-block;
- генератор случайных чисел;
- DES – вычислитель.

# Содержание


1 Структурная схема ..... 20
2 Условно-графическое изображение ..... 21
3 Описание выводов микросхемы ..... 23
4 Указания по применению и эксплуатации ..... 40
5 Управление питанием микроконтроллера ..... 41
5.1 Структура питания микросхемы ..... 41
5.2 Организация питания цифровой части микросхемы ..... 41
5.3 Организация питания батарейного домена ..... 44
5.4 Схема сброса по питанию (POR) ..... 45
5.5 Схема защиты от перенапряжения POVR ..... 46
5.6 Блок монитора основного и батарейного питаний Ucc и BUcc ..... 46
6 Сигналы сброса микроконтроллера ..... 47
7 Синхросигналы микроконтроллера ..... 49
7.1 Источники синхросигналов микроконтроллера ..... 49
7.1.1 Встроенный высокоскоростной генератор HSI ..... 49
7.1.2 Внешние высокоскоростные генераторы HSEn ..... 49
7.1.3 Встроенный низкоскоростной генератор LSI ..... 49
7.1.4 Внешний низкоскоростной часовой генератор LSE ..... 50
7.1.5 Блоки умножения тактовой частоты PLLn ..... 50
7.2 Домены синхронизации микроконтроллера ..... 51
7.3 Управление тактовыми частотами (CLKCNTR) ..... 53
7.4 Блок контроля частоты внешних генераторов и PLL ..... 53
8 Режимы загрузки микроконтроллера ..... 59
8.1 Режим WAIT_BOOT_JA ..... 62
8.2 Режим TEST_MODE+JB ..... 62
8.3 Режим FLASH+JA ..... 62
8.4 Режим FLASH +JB ..... 62
8.5 Режим EXTBUS_8_ECC+JA ..... 62
8.6 Режим EXTBUS_8_ECC+JB ..... 63
8.7 Режим EXTBUS_CFG+JA ..... 63
8.7.1 Расчет ECC для поля CFG ..... 64
8.8 Режим EXTBUS_CFG+JB ..... 64
8.9 Режим SPI0+JA ..... 64
8.9.1 Параметры связи по SPI ..... 65
8.9.2 Протокол обмена по SPI ..... 65
8.9.3 Вычисление CRC16 ..... 66
8.10 Режим SPI0+JB ..... 66
8.11 Режим SPI1+JA ..... 67
8.11.1 Параметры связи по SPI ..... 67
8.11.2 Протокол обмена по SPI ..... 67
8.12 Режим SPI1+JB ..... 67
8.13 Режим UART0+JA ..... 67
8.13.1 Параметры связи по UART ..... 68
8.13.2 Протокол обмена по UART ..... 68
8.13.3 Синхронизация с внешним устройством ..... 68
8.13.4 Команда CMD_SYNC ..... 68
8.13.5 Команда CMD_CR ..... 69
8.13.6 Команда CMD_BAUD ..... 69
8.13.7 Команда CMD_LOAD ..... 69
8.13.8 Команда CMD_VFY ..... 70
8.13.9 Команда CMD_RUN ..... 70
8.13.10 Прием параметров команды ..... 71
8.13.11 Сообщения об ошибках ..... 71
8.14 Режим UART0+JB ..... 71

8.15 Статус загрузчика ..... 71
8.16 Тестовый режим работы ..... 72
9 Организация памяти ..... 74
9.1 Контроллер кэш-памяти (CACHE_CNTR) ..... 77
9.2 Контроллер ПЗУ (ROM_CNTR) ..... 78
9.3 Контроллер FLASH (FLASH_CNTR) ..... 78
9.3.1 Доступ к памяти FLASH ..... 80
9.3.2 Защищенный режим памяти ..... 84
9.4 Контроллер ОЗУ RAMC (RAMC_CNTR) ..... 85
9.5 Контроллер ОЗУ RAMD (RAMD_CNTR) ..... 86
9.6 Контроллер внешней шины EXTBUS (EXTBUS_CNTR) ..... 86
9.6.1 Помехозащищенное кодирование для шины EXTBUS ..... 88
9.6.2 Параллельная организация ECC ..... 88
9.6.3 Последовательная организация ECC ..... 89
9.6.4 Адреса расположения кодов ECC ..... 89
9.6.5 Доступ во внешнюю память без контроля ECC ..... 90
9.6.6 Регионы внешней памяти ..... 91
9.6.7 Организация доступа с ожиданием сигнала готовности ..... 91
9.7 Контроллер диагностики памяти SCRUBBER (SCR_CNTR) ..... 92
9.8 Организация доступа к памяти контроллера DMA (DMA_CNTR) ..... 94
9.9 Организация доступа к памяти контроллера Ethernet (ETHERNETMAC_CNTR) ..... 94
9.10 Организация доступа к памяти периферии ..... 94
10 Контроллер прямого доступа к памяти (DMA_CNTR) ..... 95
10.1 Основные свойства контроллера DMA ..... 95
10.2 Термины и определения ..... 95
10.3 Функциональное описание ..... 96
10.3.1 Типы передач ..... 97
10.3.2 Разрядность передач данных ..... 97
10.3.3 Управление защитой данных ..... 97
10.3.4 Инкремент адреса ..... 98
10.4 Управление DMA ..... 98
10.4.1 Правила обмена данными ..... 98
10.4.2 Диаграммы работы контроллера DMA ..... 100
10.4.3 Правила арбитража DMA ..... 105
10.4.4 Типы циклов DMA ..... 107
10.4.5 Индикация ошибок ..... 116
10.5 Структура управляющих данных канала DMA ..... 116
10.5.1 Указатель конца данных источника ..... 119
10.5.2 Указатель конца данных приемника ..... 120
11 Контроллер обработки событий отказов, сбоев и ошибок (FT_CNTR) ..... 122
11.1 Уровень диагностики 3 ..... 122
11.2 Уровень диагностики 2 ..... 122
11.3 Уровень диагностики 1 ..... 123
11.4 Уровень диагностики 0 ..... 124
11.5 Помехозащищенное кодирование ..... 124
11.5.1 ECC-кодирование для ПЗУ ..... 124
11.5.2 ECC-кодирование для FLASH ..... 124
11.5.3 ECC-кодирование для RAMC и RAMD ..... 125
11.5.4 ECC-кодирование для внешней шины ..... 126
11.5.5 ECC-кодирование для периферии ..... 126
12 Домен батарейного питания микроконтроллера ..... 127
12.1 Контроллер батарейного домена (BKP_CNTR) ..... 127
12.2 Часы реального времени (RTC) ..... 127
12.3 Контроллер сторожевого таймера (WDG_CNTR) ..... 128
12.4 Способы обнаружения сбоев и исправления ошибок в батарейном домене 129

13 Таймеры, блоки широтно-импульсной модуляции и обработки сигналов датчиков 131
13.1 Контроллер таймеров общего назначения (TIMER_CNTR) ..... 131
13.1.1 Основные характеристики ..... 131
13.1.2 Базовый блок таймера ..... 132
13.1.3 Источники событий для счета ..... 136
13.1.4 Режим захвата ..... 142
13.1.5 Режим ШИМ ..... 144
13.1.6 Блок цифрового фильтра ..... 147
13.1.7 Флаги состояний, прерывания и запросы DMA ..... 149
13.1.8 Примеры ..... 150
13.2 Контролер ШИМ (EPWM_CNTR) ..... 154
13.2.1 Описание работы контроллера ШИМ ..... 154
13.3 Контроллер квадратурных преобразователей (QEP_CNTR) ..... 169
13.3.1 Блок квадратурного декодера QDU ..... 170
13.3.2 Блок счетчика положения PCCU ..... 172
13.3.3 Блок захвата фронтов QCAP ..... 175
13.3.4 Блок таймера временных отсчетов UTIME ..... 177
13.3.5 Блок сторожевого таймера QWDOG ..... 177
13.3.6 Прерывания и флаги состояния ..... 178
13.4 Контроллер захвата (CAP_CNTR) ..... 179
13.4.1 Режим захвата внешних событий ..... 180
13.4.2 Режим вспомогательного ШИМ (APWM) ..... 181
13.4.3 Прерывания ..... 183
14 Контроллеры аналоговых и цифро-аналоговых блоков ..... 184
14.1 Контроллер АЦП (ADC_CNTR) ..... 184
14.1.1 Запуск цепочек преобразований ..... 185
14.1.2 Выбор канала для преобразования ..... 185
14.1.3 Автомат управления (АУ) последовательностью преобразований ..... 186
14.1.4 Синхронизация преобразований ..... 188
14.1.5 Запись результатов преобразований ..... 189
14.1.6 Калибровка блоков АЦП ..... 189
14.1.7 Прерывания и запросы DMA ..... 190
14.1.8 Работа с температурными датчиками ..... 190
14.1.9 Управление источником опорных напряжений и токов (ИОНТ) ..... 190
14.1.10 Входные сопротивление и емкость ..... 191
14.2 Контроллер ЦАП (DAC_CNTR) ..... 191
14.3 Контроллер компаратора (COMP_CNTR) ..... 193
14.4 Контроллер формирования опорных напряжений и токов (ADC_BG_CNTR, ANA_BG_CNTR) ..... 195
15 Блоки контроллеров интерфейсов передачи данных ..... 198
15.1 Контроллер SSP (SSP_CNTR) ..... 198
15.1.1 Программируемые параметры ..... 198
15.1.2 Общий обзор модуля SSP ..... 200
15.1.3 Интерфейс прямого доступа к памяти ..... 214
15.1.4 Прерывания ..... 216
15.2 Контроллер UART (UART_CNTR) ..... 218
15.2.1 Основные сведения ..... 218
15.2.2 Основные характеристики модуля UART ..... 218
15.2.3 Программируемые параметры ..... 219
15.2.4 Описание функционирования блока UART ..... 221
15.2.5 Интерфейс прямого доступа к памяти ..... 222
15.2.6 Блок и регистры синхронизации ..... 222
15.2.7 Описание функционирования ИК кодека IrDASIR ..... 222
15.2.8 Описание работы UART ..... 224
15.2.9 Работа кодека ИК обмена данными IrDASIR ..... 227

15.2.10 Модуляция данных IrDA ..... 228
15.2.11 Линии управления модемом ..... 228
15.2.12 Интерфейс прямого доступа к памяти ..... 230
15.2.13 Прерывания ..... 232
15.3 Контроллер Ethernet (ETHERNETMAC_CNTR) ..... 235
15.3.1 Передача пакета ..... 235
15.3.2 Принцип работы передатчика ..... 236
15.3.3 Прием пакета ..... 236
15.3.4 Принцип работы приемника ..... 237
15.3.5 Линейный режим работы буферов ..... 237
15.3.6 Автоматический режим работы буферов ..... 237
15.3.7 Режим FIFO работы буферов ..... 238
15.3.8 События приемника и передатчика ..... 238
15.3.9 Прерывания ..... 238
15.3.10 Режим детерминированного времени доставки ..... 239
15.3.11 Режим КЗ ..... 239
15.3.12 Режим RMIInMII ..... 239
15.4 Контроллер CAN (CAN_CNTR) ..... 240
15.4.1 Режимы работы ..... 240
15.4.2 Типы пакетов сообщений ..... 242
15.4.3 Структура пакета данных (DataFrame) ..... 243
15.4.4 Структура пакета удаленного запроса данных (Remoteframe) ..... 245
15.4.5 Арбитраж на шине ..... 246
15.4.6 Инициализация ..... 246
15.4.7 Передача сообщений ..... 247
15.4.8 Передача сообщений по RemoteTransmitRequest (RTR) ..... 248
15.4.9 Прием сообщений ..... 248
15.4.10 Автоматическая фильтрация принимаемых сообщений ..... 248
15.4.11 Перезапись принятых сообщений ..... 248
15.4.12 Задание скорости передачи и момента сэмплирования ..... 249
15.4.13 Синхронизация ..... 250
15.4.14 Обработка ошибок ..... 250
15.4.15 Прерывания ..... 253
15.5 Контроллер CAN FD ..... 254
15.5.1 Настройка ..... 254
15.5.2 Передача данных ..... 254
15.5.3 Прием данных ..... 256
15.5.4 Управление частотой MAC ..... 256
15.5.5 Синхронизация данных ..... 256
15.5.6 Описание регистров контроллера CAN FD CANFD_CNTR ..... 257
15.6 Контроллер МКПД (MIL_CNTR) ..... 274
15.6.1 Режимы работы ..... 275
15.6.2 Форматы сообщений ..... 276
15.6.3 Формат слов ..... 277
15.6.4 Структурная схема в режиме КШ ..... 279
15.6.5 Структурная схема в режиме ОУ ..... 280
15.6.6 Структурная схема в режиме М ..... 281
15.6.7 Инициализация ..... 281
15.6.8 Приём и передача в режиме ОУ ..... 282
15.6.9 Приём и передача в режиме КШ ..... 283
15.6.10 Прерывания ..... 284
15.7 Контроллер I2C (I2C_CNTR) ..... 284
15.7.1 Конфигурация контроллера I2C ..... 285
15.7.2 Конфигурация системы ..... 285
15.7.3 Протокол I2C ..... 285
15.7.4 Сигнал START ..... 286

15.7.5 Передача адреса ..... 286
15.7.6 Передача данных ..... 287
15.7.7 Сигнал STOP ..... 287
15.8 Контроллер USB (USB_CNTR) ..... 287
15.8.1 Поддержка нескольких устройств ..... 288
15.8.2 Схема программирования ..... 289
15.8.3 Контроль транзакций (при помощи конечной токи 0) ..... 292
15.8.4 Поточные (Bulk) передачи транзакций ..... 301
15.8.5 Full-speed/low-bandwidth (низкопропускные) передачи по прерываниям ..... 307
15.8.6 Full-speed/low-bandwidth (низкопропускные) изохронные передачи ..... 308
15.8.7 High-bandwidth (высокопропускные) изохронные передачи/ передачи по прерываниям ..... 313
15.8.8 Поточная передача/низкопропускная передача по прерываниям ..... 321
15.8.9 High-speed/low-bandwidth изохронные передачи ..... 323
15.8.10 Высокопропускные передачи (изохронные/по прерываниям) ..... 324
15.8.11 Передача транзакций со стороны хоста ..... 324
15.8.12 Поточная передача/низкопропускная передача по прерываниям ..... 327
15.8.13 Full-speed/низкопропускные передачи по прерываниям ..... 328
15.8.14 Высокопропускные передачи (изохронные/по прерываниям) ..... 329
15.8.15 Режим тестирования ..... 329
15.9 Контроллер SDIO (SDIO_CNTR) ..... 331
15.10 Контроллер портов ввода-вывода (PORT_CNTR) ..... 332
16 Контроллер CRC (CRC_CNTR) ..... 334
17 Контроллер тригонометрических преобразований (CORDIC_CNTR) ..... 336
17.1 Основные функции блока ..... 336
17.2 Входные параметры и результат работы модуля ..... 337
17.3 Функционирование модуля ..... 337
17.4 Прерывания модуля ускорителя тригонометрических преобразований ..... 338
18 Защищенный модуль криптографических вычислений ..... 339
18.1 Алгоритм загрузки защищенной подсистемы ..... 340
18.2 Процессорное ядро ..... 340
18.3 Контроллер шлюза передачи данных между подсистемами (GATE_CNTR) 340
18.4 Память ключей ..... 341
18.5 Контроллер генератора шума ..... 341
18.6 Блокировка отладочного интерфейса ..... 342
18.7 Блоки специальных вычислений ..... 343
18.7.1 Контроллер прямого и обратного L-преобразования (L_BLOCK_CNTR) ..... 343
18.7.2 Контроллер замены (S_BLOCK_CNTR) ..... 345
18.7.3 Контроллер битовой замены (P_BIT_CNTR) ..... 346
18.7.4 Контроллер байтовой замены (P_BYTE_CNTR) ..... 347
18.7.5 Контроллер шифрования (DES_CNTR) ..... 348
18.8 Контроллер сопроцессора длинночисленной арифметики (ALU_CNTR) ..... 349
18.8.1 Реализация длинных операций ..... 352
18.9 Контроллер обработки датчиков безопасности (SENSORS_CNTR) ..... 355
18.9.1 Общее описание ..... 355
18.9.2 Описание функционирования блока и режимов работы ..... 356
18.10 Контроллер монитора частоты (CLK_MEASURE_CNTR) ..... 356
18.11 Контроллер генератора случайных чисел (RANDOM_CNTR) ..... 358
18.12 Контроллер USART ISO7816 (USART_CNTR) ..... 358
18.12.1 Особенности модуля USART ..... 359
18.12.2 Функциональное описание USART ..... 359
18.12.3 Прерывания USART ..... 381
18.13 Контроллер OTP (OTP_CNTR) ..... 382

19 Системный таймер SysTick ..... 385
19.1 Описание регистров системного таймера SysTick ..... 385
19.1.1 SysTick->CTRL ..... 385
19.1.2 SysTick->LOAD ..... 386
19.1.3 SysTick->VAL ..... 386
19.1.4 SysTick->CAL ..... 386
19.2 Советы и особенности при применении системного таймера ..... 387
20 Программная модель микроконтроллера ..... 388
20.1 Таблица векторов прерываний ..... 388
20.2 Адресное пространство микроконтроллера ..... 391
20.3 Адресное пространство периферийных блоков микроконтроллера ..... 392
20.4 Адресное пространство сектора регистров ядра ..... 393
20.5 Регистры ядра (SCB) и контроллер прерываний (ISR) ..... 394
20.6 Описание регистров контроллера кэш-памяти (CACHE_CNTR) ..... 396
20.6.1 KEY ..... 397
20.6.2 CNTR ..... 397
20.6.3 HIT_CNT ..... 398
20.6.4 MISS_CNT ..... 398
20.6.5 ECCCS ..... 398
20.6.6 ECCADR ..... 399
20.6.7 ECCDATA ..... 400
20.6.8 ECCECC ..... 401
20.7 Описание регистров контроллера диагностики памяти SCR_CNTR ..... 401
20.7.1 SCR_CNTRL ..... 401
20.7.2 SCR_SADDR ..... 403
20.7.3 SCR_FADDR ..... 403
20.7.4 SCR_DATA ..... 403
20.7.5 SCR_ECC ..... 404
20.8 Описание регистров контроллера тактовых частот CLK_CNTR ..... 404
20.8.1 KEY ..... 408
20.8.2 MAX_CLK ..... 409
20.8.3 CPU_CLK ..... 410
20.8.4 CPU_CHK0 ..... 411
20.8.5 CPU_CHK1 ..... 412
20.8.6 CPU_CHK2 ..... 412
20.8.7 CPU_STAT ..... 413
20.8.8 LSI_CLK ..... 414
20.8.9 LSI_CHK0 ..... 415
20.8.10 LSI_CHK1 ..... 416
20.8.11 LSI_CHK2 ..... 416
20.8.12 LSI_STAT ..... 417
20.8.13 HSI_STAT ..... 418
20.8.14 LSE_CLK ..... 418
20.8.15 LSE_CHK0 ..... 419
20.8.16 LSE_CHK1 ..... 420
20.8.17 LSE_CHK2 ..... 420
20.8.18 LSE_STAT ..... 421
20.8.19 HSEn_CLK ..... 422
20.8.20 HSEn_CHK0 ..... 424
20.8.21 HSEn_CHK1 ..... 424
20.8.22 HSEn_CHK2 ..... 425
20.8.23 HSEn_STAT ..... 425
20.8.24 PLLn_CLK ..... 426
20.8.25 PLLn_CHK0 ..... 428
20.8.26 PLLn_CHK1 ..... 429
20.8.27 PLLn_CHK2 ..... 429

20.8.28 PLLn_STAT ..... 430
20.8.29 Регистры синхросигналов периферийных блоков ..... 431
20.8.30 Синхронный регистр управления частотой периферийного блока 434
20.8.31 Асинхронный регистр управления частотой периферийного блока ..... 434
20.9 Описание регистров контроллера обработки событий отказов, сбоев и ошибок FT_CNTR ..... 434
20.9.1 KEY ..... 436
20.9.2 CONTROL ..... 436
20.9.3 STATUS ..... 437
20.9.4 TIMEOUT ..... 438
20.9.5 TICKCNT ..... 438
20.9.6 FIRSTEVENT ..... 439
20.9.7 LASTEVENT ..... 439
20.9.8 TIMEOUTCNT ..... 440
20.9.9 EVENT0 (HIGH) ..... 441
20.9.10 EVENT1 (HIGH) ..... 444
20.9.11 EVENT2 (HIGH) ..... 446
20.9.12 EVENT4 (HIGH) ..... 447
20.9.13 EVENT5 (MIDDLE) ..... 448
20.9.14 EVENT6 (MIDDLE) ..... 451
20.9.15 EVENT7 (MIDDLE) ..... 452
20.9.16 EVENT9 (LOW) ..... 456
20.9.17 EVENT10 (LOW) ..... 459
20.9.18 EVENT11 (LOW) ..... 460
20.9.19 EVENT12 (LOW) ..... 461
20.9.20 RESET_EVENT0 ..... 462
20.9.21 RESET_EVENT1 ..... 462
20.9.22 RESET_EVENT2 ..... 463
20.9.23 RESET_EVENT4 ..... 463
20.9.24 IE_EVENT5 ..... 464
20.9.25 IE_EVENT6 ..... 464
20.9.26 IE_EVENT7 ..... 465
20.9.27 IE_EVENT9 ..... 465
20.9.28 IE_EVENT10 ..... 466
20.9.29 IE_EVENT11 ..... 466
20.9.30 IE_EVENT12 ..... 467
20.10 Описание регистров контроллера ПЗУ ROM_CNTR ..... 468
20.10.1 KEY ..... 468
20.10.2 ECCCS ..... 469
20.10.3 ECCADR ..... 470
20.10.4 ECCDATA ..... 470
20.10.5 ECCECC ..... 471
20.11 Описание регистров контроллеров ОЗУ RAMC_CNTR и RAMD_CNTR ..... 471
20.11.1 KEY ..... 471
20.11.2 ECCCS ..... 473
20.11.3 ECCADR ..... 474
20.11.4 ECCDATA ..... 474
20.11.5 ECCECC ..... 475
20.11.6 TEST_TUNING ..... 475
20.12 Описание регистров контроллера FLASH FLASH_CNTR ..... 476
20.12.1 KEY ..... 476
20.12.2 CNTR ..... 477
20.12.3 ADDR ..... 478
20.12.4 WDATA3…WDATA0 ..... 479
20.12.5 WECC0 .. 1 ..... 479

20.12.6 RDATA3…RDATA0 ..... 480
20.12.7 RECC 0..1 ..... 480
20.12.8 ECCCS ..... 481
20.12.9 ECCADR ..... 482
20.12.10 ECCDATA ..... 482
20.12.11 ECCECC ..... 483
20.12.12 BLOCK ..... 483
20.13 Описание регистров контроллера батарейного домена BKP_CNTR ..... 484
20.13.1 KEY ..... 486
20.13.2 BLDO_CNTRx ..... 486
20.13.3 REG_00…REG_59 ..... 487
20.13.4 REG_60_SYSx ..... 487
20.13.5 REG_61_PWRx ..... 489
20.13.6 REG_62_PWRx ..... 491
20.13.7 REG_63_CLKx ..... 492
20.13.8 REG_64_TMRx ..... 493
20.13.9 RTC_CNT_TMRx ..... 494
20.13.10 RTC_DIV_TMRx ..... 495
20.13.11 RTC_PRL_TMRx ..... 495
20.13.12 RTC_ALRM_TMRx ..... 495
20.13.13 RTC_CS_TMRx ..... 496
20.14 Описание регистров контроллера сторожевого таймера WDG_CNTR ..... 498
20.14.1 KEY ..... 498
20.14.2 PRL ..... 498
20.14.3 EN ..... 499
20.14.4 CNT ..... 499
20.15 Описание регистров контроллера прямого доступа в память DMA_CNTR ..... 501
20.15.1 STATUS ..... 502
20.15.2 CFG ..... 503
20.15.3 CNTR_BASE_PTR ..... 503
20.15.4 ALT_CNTR_BASE_PTR ..... 504
20.15.5 WAITONREQ_STATUS ..... 504
20.15.6 CHNL_SW_REQUEST ..... 505
20.15.7 CHNL_USEBURST_SET ..... 505
20.15.8 CHNL_USEBURST_CLR ..... 506
20.15.9 CHNL_REQ_MASK_SET ..... 507
20.15.10 CHNL_REQ_MASK_CLR ..... 508
20.15.11 CHNL_ENABLE_SET ..... 509
20.15.12 CHNL_ENABLE_CLR ..... 509
20.15.13 CHNL_PRI_ALT_SET ..... 511
20.15.14 CHNL_PRI_ALT_CLR ..... 512
20.15.15 CHNL_PRIORITY_SET ..... 513
20.15.16 CHNL_PRIORITY_CLR ..... 513
20.15.17 ERR_CLR ..... 514
20.15.18 CHMUX[x] ..... 515
20.15.19 Структуры управления DMA ..... 515
20.15.20 Распределение виртуальных каналов DMA ..... 519
20.16 Описание регистров контроллеров портов ввода-вывода PORT_CNTR ..... 521
20.16.1 KEY ..... 523
20.16.2 RXTX ..... 523
20.16.3 SRXTX ..... 524
20.16.4 CRXTX ..... 525
20.16.5 SOE ..... 525
20.16.6 COE ..... 526
20.16.7 SFUNCx ..... 527
20.16.8 CFUNCx ..... 528

20.16.9 SANALOG....529
20.16.10 CANALOG....530
20.16.11 SPULLUP....530
20.16.12 CPULLUP....531
20.16.13 SPULLDOWN....531
20.16.14 CPULLDOWN....532
20.16.15 SPD....532
20.16.16 CPD....533
20.16.17 SPWRx....533
20.16.18 CPWRx....534
20.16.19 SCL....535
20.16.20 CCL....536
20.16.21 SIE....536
20.16.22 CIE....537
20.16.23 SIT....537
20.16.24 CIT....538
20.16.25 SIR....538
20.16.26 CIR....539
20.16.27 HCUR....539
20.17 Описание регистров контроллера внешней шины EXT_BUS_CNTR....541
20.17.1 Регистр KEY....541
20.17.2 Регистр REGION[n].CNTRL....542
20.17.3 Регистр REGION[n].ECCBASE....543
20.17.4 Регистр REGION[n].ECCCR....544
20.17.5 Регистр REGION[n].ECCST....545
20.17.6 Регистр ECCADR....546
20.17.7 Регистр ECCDATA....546
20.17.8 Регистр ECCECC....547
20.17.9 OCLKCTL....547
20.18 Описание регистров контроллера Ethernet ETHERNETMAC_CNTR....548
20.18.1 Поле управления передачи пакета....549
20.18.2 Поле состояния передачи пакета....550
20.18.3 Поле состояния приёма пакета....551
20.18.4 DELIMETR....552
20.18.5 MAC_T....552
20.18.6 MAC_M....552
20.18.7 MAC_H....553
20.18.8 G_CFG....553
20.18.9 X_CFG....555
20.18.10 R_CFG....556
20.18.11 IMR/IFR....557
20.18.12 STAT....558
20.18.13 MDIO_CNTR....559
20.18.14 PHY_CONTROL....559
20.19 Описание регистров контроллеров CAN CAN_CNTR....561
20.19.1 CONTROL....562
20.19.2 STATUS....563
20.19.3 BITTMNG....564
20.19.4 INT_EN....565
20.19.5 OVER....566
20.19.6 BUF_CON[x]....567
20.19.7 INT_RX....567
20.19.8 RX....568
20.19.9 INT_TX....568
20.19.10 TX....569
20.19.11 RXID....570

20.19.12 TXID ..... 570
20.19.13 BUF[x].ID ..... 570
20.19.14 FILTER[x].MASK ..... 570
20.19.15 FILTER[x].FILTER ..... 570
20.19.16 RXDLC ..... 571
20.19.17 TXDLC ..... 571
20.19.18 BUF[x].DLC ..... 571
20.19.19 RXDATAL ..... 572
20.19.20 TXDATAL ..... 572
20.19.21 BUF[x].DATAL ..... 572
20.19.22 RXDATAH ..... 572
20.19.23 TXDATAH ..... 572
20.19.24 BUF[x].DATAH ..... 572
20.20 Описание регистров контроллеров SSP SSP_CNTR ..... 574
20.20.1 CR0 ..... 574
20.20.2 CR1 ..... 575
20.20.3 DR ..... 576
20.20.4 SR ..... 576
20.20.5 CPSR ..... 577
20.20.6 IMSC ..... 577
20.20.7 RIS ..... 578
20.20.8 MIS ..... 578
20.20.9 ICR ..... 579
20.20.10 DMACR ..... 579
20.21 Описание регистров контроллеров UART UART_CNTR ..... 579
20.21.1 DR ..... 581
20.21.2 RSR_ECR ..... 582
20.21.3 FR ..... 583
20.21.4 ILPR ..... 584
20.21.5 IBRD ..... 584
20.21.6 FBRD ..... 585
20.21.7 LCR_H ..... 587
20.21.8 CR ..... 589
20.21.9 IFLS ..... 590
20.21.10 IMSC ..... 591
20.21.11 RIS ..... 592
20.21.12 MIS ..... 593
20.21.13 ICR ..... 594
20.21.14 DMACR ..... 594
20.22 Описание регистров контроллера I2C I2C_CNTR ..... 595
20.22.1 I2C->PRL ..... 595
20.22.2 I2C->PRH ..... 595
20.22.3 I2C->CTR ..... 596
20.22.4 I2C->RXD ..... 596
20.22.5 I2C->STA ..... 597
20.22.6 I2C->TXD ..... 597
20.22.7 I2C->CMD ..... 598
20.23 Описание регистров контроллеров МКПД MIL_CNTR ..... 598
20.23.1 CONTROL ..... 599
20.23.2 STATUS ..... 601
20.23.3 ERROR ..... 603
20.23.4 CommandWord1 ..... 604
20.23.5 CommandWord2 ..... 605
20.23.6 ModeData ..... 606
20.23.7 StatusWord1 ..... 607
20.23.8 StatusWord2 ..... 608

20.23.9 INTEN ..... 609
20.23.10 MSG ..... 610
20.23.11 DATA ..... 611
20.24 Описание регистров контроллеров формирования опорных напряжений и токов ADC_BG_CNTR и ANA_BG_CNTR ..... 611
20.24.1 ADC_BG_CNTR ..... 611
20.24.2 ANA_BG_CNTR ..... 612
20.25 Описание регистров блока контроллера компаратора COMP_CNTR ..... 613
20.25.1 COMP_CNTR ..... 613
20.25.2 COMP_EVNT ..... 614
20.26 Описание регистров контроллера ЦАП DAC_CNTR ..... 615
20.26.1 DAC_CNTR ..... 615
20.26.2 FIFO_CNTR ..... 616
20.26.3 DAC_PRD ..... 617
20.26.4 DAC_CNT ..... 617
20.26.5 DAC_IE ..... 618
20.26.6 DAC_RE ..... 618
20.26.7 DAC_STS ..... 619
20.26.8 DAC_DATA ..... 620
20.27 Описание регистров контроллеров АЦП ADC_CNTR ..... 620
20.27.1 ADC0_CNTR ..... 621
20.27.2 ADC1_CNTR ..... 623
20.27.3 ADCx_SCOPE_CNTR ..... 624
20.27.4 ADC_SYNC_CNTR ..... 624
20.27.5 ADC_BUF_CNTR ..... 625
20.27.6 ADC_BUF_STATE ..... 626
20.27.7 ADC_INT_CNTR ..... 626
20.27.8 ADC_DMA_CNTR ..... 628
20.27.9 ADC_ANALOG_CNTR ..... 628
20.27.10 ADC_BG_CNTR ..... 629
20.27.11 ADCx_CHSEL ..... 630
20.27.12 ADC0_RESULT_i ..... 630
20.27.13 ADC1_RESULT_i ..... 631
20.28 Описание регистров контроллеров ШИМ EPWM_CNTR ..... 632
20.28.1 Регистры блока счетчика таймера ..... 633
20.28.2 Регистры блока сравнения ..... 638
20.28.3 Регистры блока анализа событий ..... 643
20.28.4 Регистры блока управления мертвой зоной ..... 649
20.28.5 Регистры блока модуляции высокочастотного сигнала ..... 651
20.28.6 Регистры блока защиты ..... 652
20.28.7 Регистры блока генерации событий ..... 660
20.28.8 Регистры блока управления ШИМ высокого разрешения HRPWM_CNTR ..... 667
20.29 Описание регистров контроллеров таймеров общего назначения TIMER_CNTR ..... 669
20.29.1 CNT ..... 670
20.29.2 PSG ..... 670
20.29.3 ARR ..... 670
20.29.4 CNTRL ..... 670
20.29.5 CHy_CCR ..... 672
20.29.6 CHy_CCR1 ..... 672
20.29.7 CHy_CNTRL0 ..... 672
20.29.8 CHy_CNTRL1 ..... 674
20.29.9 CHy_CNTRL2 ..... 675
20.29.10 CHy_DTG ..... 676
20.29.11 BRKETR_CNTRL ..... 676

20.29.12 STATUS ..... 677
20.29.13 IE ..... 679
20.29.14 DMA_RE0-DMA_RE4 ..... 680
20.30 Описание регистров контроллеров квадратурных декодеров QEP_CNTR ..... 681
20.30.1 QDECCNTR ..... 682
20.30.2 QEPCNTR ..... 684
20.30.3 QPOSCNTR ..... 685
20.30.4 QCAPCNTR ..... 686
20.30.5 QPOSCNT ..... 687
20.30.6 QPOSINIT ..... 687
20.30.7 QPOSMAX ..... 688
20.30.8 QPOSCMP ..... 688
20.30.9 QPOSILAT ..... 689
20.30.10 QPOSSLAT ..... 689
20.30.11 QPOSLAT ..... 690
20.30.12 QUTMR ..... 690
20.30.13 QUPRD ..... 691
20.30.14 QWDTMR ..... 691
20.30.15 QWDPRD ..... 691
20.30.16 QEINT ..... 692
20.30.17 QFLG ..... 693
20.30.18 QCLR ..... 694
20.30.19 QFRC ..... 696
20.30.20 QEPSTS ..... 697
20.30.21 QCTMR, QCTMR1, QCTMR2, QCTMR3, QCTMR4 ..... 698
20.30.22 QCPRD ..... 698
20.30.23 QCTMRLAT ..... 699
20.30.24 QCPRDLAT ..... 699
20.31 Описание регистров контроллеров захвата CAP_CNTR ..... 700
20.31.1 TSCTR ..... 700
20.31.2 CTRPHS ..... 700
20.31.3 CAP1 ..... 701
20.31.4 CAP2 ..... 702
20.31.5 CAP3 ..... 702
20.31.6 CAP4 ..... 703
20.31.7 ECCTL1 ..... 703
20.31.8 ECCTL2 ..... 705
20.31.9 ECEINT ..... 706
20.31.10 ECFLG ..... 707
20.31.11 ECCLR ..... 708
20.31.12 ECFRC ..... 709
20.32 Описание регистров контроллера тригонометрических преобразований CORDIC_CNTR ..... 710
20.32.1 Регистр управления модулем CRD_CNTR ..... 710
20.32.2 Регистр статуса модуля CRD_STATUS ..... 712
20.32.3 Регистр флагов прерываний CRD_INTF ..... 713
20.32.4 Регистр разрешения прерываний CRD_INTE ..... 714
20.32.5 Регистр записи координаты X вектора IN_X ..... 714
20.32.6 Регистр записи координаты Y вектора IN_Y ..... 714
20.32.7 Регистр записи значения угла α IN_A ..... 714
20.32.8 Регистр последовательной записи IN_SEQ ..... 715
20.32.9 Регистр чтения координаты X вектора результата OUT_X ..... 715
20.32.10 Регистр чтения координаты Y вектора результата OUT_Y ..... 715
20.32.11 Регистр чтения значения угла α OUT_A ..... 716
20.32.12 Регистр последовательного чтения OUT_SEQ ..... 716
20.32.13 Регистр выбора адреса записи IN_ ADDR ..... 716

20.32.14 Регистр выбора адреса чтения OUT_ ADDR ..... 716
20.33 Описание регистров контроллера USB USB_CNTR ..... 717
20.33.1 Общие регистры ..... 720
20.33.2 Индексные регистры ..... 727
20.33.3 Описание дополнительных контрольных/статусных регистров ..... 740
20.34 Описание регистров контроллера SDIO SDIO_CNTR ..... 741
20.34.1 SDIO_POWER ..... 741
20.34.2 SDIO_CLKCR ..... 742
20.34.3 SDIO_ARG ..... 744
20.34.4 SDIO_CMD ..... 744
20.34.5 SDIO_RESPCMD ..... 745
20.34.6 SDIO_RESPx ..... 746
20.34.7 SDIO_DTIMER ..... 747
20.34.8 SDIO_DLEN ..... 747
20.34.9 SDIO_DCNTR ..... 748
20.34.10 SDIO_DCOUNT ..... 749
20.34.11 SDIO_STA ..... 750
20.34.12 SDIO_ICR ..... 751
20.34.13 SDIO_MASK ..... 753
20.34.14 SDIO_FIFOCNT ..... 756
20.34.15 SDIO_FIFO ..... 756
20.35 Описание регистров контроллера CRC CRC_CNTR ..... 757
20.35.1 DR ..... 757
20.35.2 IDR ..... 757
20.35.3 CR ..... 758
20.35.4 INIT ..... 758
20.35.5 POL ..... 760
20.36 Описание регистров контроллеров интерфейсов обмена GATE_CNTR ..... 760
20.36.1 FIFO_OP_TO_SF ..... 761
20.36.2 FIFO_SF_TO_OP ..... 761
20.36.3 FIFO_LEVELS ..... 761
20.36.4 FIFO_INT_MASK ..... 762
20.36.5 FIFO_INT_SOURCE ..... 762
20.36.6 OP_REG0…OP_REG7 ..... 763
20.36.7 OP_REG8…OP_REG15 ..... 764
20.36.8 REGS_BUSY ..... 764
20.36.9 REGS_INT_MASK ..... 765
20.36.10 REGS_INT_SOURCE ..... 766
21 Программная модель защищенной подсистемы выполнения криптографических преобразований ..... 768
21.1 Адресное пространство защищенной подсистемы ..... 768
21.2 Описание регистров контроллера монитора частоты (CLK_MEASURE_CNTR) ..... 769
21.2.1 CLK_CNTR_STAT ..... 769
21.2.2 ALARM_SHIFT_RST ..... 769
21.2.3 ALARM_SHIFT_INT ..... 770
21.2.4 ALARM_PREG_0 ..... 770
21.2.5 ALARM_PREG_1 ..... 771
21.3 Описание регистров контроллера генератора случайных чисел (RANDOM_CNTR) ..... 771
21.3.1 Регистр статуса и управления STAT_CNTR_REG ..... 772
21.3.2 Регистр управления прерыванием INT_CNTR_REG ..... 772
21.3.3 Регистр делителя клока генератора CLK_DIV_REG ..... 773
21.3.4 Регистр паузы включения PAUSE_REG ..... 773
21.3.5 Регистр случайного значения OUTPUT_REG ..... 773
21.3.6 Регистр счетчика паузы PAUSE_CNT_REG ..... 773

21.3.7 Регистр сбора случайного числа TEMP_REG ..... 773
21.4 Описание регистров контроллера обработки датчиков безопасности (SENSORS_CNTR) ..... 774
21.4.1 SENS_STATE ..... 774
21.4.2 SENS_RT_STATE ..... 774
21.4.3 SENS_INT_MASK ..... 775
21.4.4 SENS_KEY_RESET_MASK ..... 775
21.4.5 SENS_ENABLE ..... 776
21.4.6 MESH_CNTR ..... 776
21.5 Описание регистров контроллера USART ISO7816 (USART_CNTR) ..... 777
21.5.1 USART_SR ..... 777
21.5.2 USART_DR ..... 780
21.5.3 USART_BRR ..... 781
21.5.4 USART_CR1 ..... 781
21.5.5 USART_CR2 ..... 784
21.5.6 USART_CR3 ..... 785
21.5.7 USART_GTPR ..... 787
21.6 Описание регистров контроллера генератора шума (NOISE_GEN_CNTR) ..... 787
21.6.1 Регистр CONTROL ..... 789
21.6.2 Регистр SEED_0 ..... 789
21.6.3 Регистр SEED_1 ..... 790
21.6.4 Регистр SEED_2 ..... 790
21.6.5 Регистр SEED_3 ..... 790
21.6.6 Регистр LFSR_0 ..... 790
21.6.7 Регистр LFSR_1 ..... 790
21.6.8 Регистр LFSR_2 ..... 791
21.6.9 Регистр LFSR_3 ..... 791
21.6.10 Регистр RAND_0 ..... 791
21.6.11 Регистр RAND_1 ..... 791
21.6.12 Регистр RAND_2 ..... 791
21.6.13 Регистр RAND_3 ..... 791
21.6.14 Регистр POLY_0 ..... 792
21.6.15 Регистр POLY_1 ..... 792
21.6.16 Регистр POLY_2 ..... 792
21.6.17 Регистр POLY_3 ..... 792
21.6.18 Регистр PERIODS ..... 793
21.6.19 Регистр MASK_0 ..... 793
21.6.20 Регистр MASK_1 ..... 793
21.6.21 Регистр MASK_2 ..... 793
21.6.22 Регистр MASK_3 ..... 793
21.7 Описание регистров системного контроллера (SYS_CNTR) ..... 794
21.7.1 Описание бит регистра STATUS ..... 794
21.8 Описание регистров контроллера портов ввода-вывода (GPIO_CNTR) ..... 795
21.9 Описание регистров контроллера шифрования (DES_CNTR) ..... 796
21.10 Описание регистров контроллера битовой замены (P_BIT_CNTR) ..... 796
21.10.1 Регистр задания n-го слова данных для перестановки TRANSFORM_n ..... 796
21.10.2 Регистр задания перестановки MUX_SET ..... 797
21.10.3 Регистр чтения перестановки MUX_VAL ..... 797
21.11 Описание регистров контроллера байтовой замены (P_BYTE_CNTR) ..... 797
21.11.1 Регистр задания n-го слова данных для перестановки TRANSFORM_n ..... 797
21.11.2 Регистр задания перестановки MUX_SET ..... 798
21.11.3 Регистр чтения перестановки MUX_VAL ..... 798
21.12 Описание регистров контроллера замены (S_BLOCK_CNTR) ..... 798
21.12.1 TRANSFORM_x ..... 799

21.12.2 TABLE_CHANGE ..... 800
21.13 Описание регистров контроллера прямого и обратного L-преобразования (L_BLOCK_CNTR) ..... 801
21.13.1 TRANSFORM_x ..... 801
21.13.2 TABLE_CHANGE_x ..... 802
21.13.3 SETUP ..... 803
21.14 Описание регистров контроллера сопроцессора длинночисленной арифметики (ALU_CNTR) ..... 804
21.14.1 Операции АЛУ ..... 804
21.14.2 Регистры контроллера ..... 804
21.14.3 Память инструкций ..... 807
21.15 Описание регистров контроллера шлюза передачи данных между подсистемами (GATE_CNTR) ..... 809
21.15.1 FIFO_OP_TO_SF ..... 809
21.15.2 FIFO_SF_TO_OP ..... 809
21.15.3 FIFO_LEVELS ..... 810
21.15.4 FIFO_INT_MASK ..... 810
21.15.5 FIFO_INT_SOURCE ..... 811
21.15.6 SF_REG0…SF_REG7 ..... 812
21.15.7 SF_REG8…SF_REG15 ..... 812
21.15.8 REGS_BUSY ..... 813
21.15.9 REGS_INT_MASK ..... 814
21.15.10 REGS_INT_SOURCE ..... 815
21.16 Описание регистров контроллера OTP (OTP_CNTR) ..... 816
21.16.1 Регистр статуса и управления STAT_CNTR ..... 817
21.16.2 Регистр статуса блоков OTP_STAT_REG ..... 819
21.16.3 Регистр задержки DELAY_0_REG ..... 820
21.16.4 Регистр задержки DELAY_1_REG ..... 820
21.16.5 Регистр команды чтения-записи RW_CMD_REG ..... 821
21.16.6 Регистр считанных данных READ_DATA_REG ..... 822
21.16.7 Регистр защиты от записи WRITE_PROTECT_REG ..... 823
21.16.8 Регистр защиты от чтения READ_PROTECT_REG ..... 824
21.16.9 Регистр флагов тестовых опций TEST_OPT ..... 825
22 Прерывания и исключения ..... 826
22.1 Прерывания и исключения (Cortex-M0) ..... 826
22.1.1 Типы исключений ..... 826
22.1.2 Обработчики исключений ..... 827
22.1.3 Приоритеты исключений ..... 829
22.1.4 Вход в обработчик и выход из обработчика ..... 829
22.1.5 Управление электропитанием ..... 831
22.2 Прерывания и исключения (Cortex-M4) ..... 833
22.2.1 Типы исключений ..... 833
22.2.2 Обработчики исключений ..... 835
22.2.3 Приоритеты исключений ..... 836
22.2.4 Вход в обработчик и выход из обработчика ..... 836
22.2.5 Обработка отказов ..... 838
22.2.6 Управление электропитанием ..... 840
23 Контроллеры прерываний NVIC ..... 843
23.1 Контроллер прерываний NVIC (Cortex-M0) ..... 843
23.1.1 Логика работы прерываний контроллера NVIC. ..... 843
23.1.2 Регистр разрешения прерываний ..... 846
23.1.3 Регистр запрета прерываний ..... 847
23.1.4 Регистр установки состояния ожидания для прерывания ..... 847
23.1.5 Регистр сброса состояния ожидания для прерывания ..... 848
23.1.6 Регистры приоритета прерываний ..... 848
23.1.7 Прерывания, срабатывающие по уровню сигнала ..... 849

23.1.8 Аппаратное и программное управление прерываниями ..... 849
23.1.9 Рекомендации по работе с контроллером прерываний ..... 850
23.2 Контроллер прерываний NVIC (Cortex-M4) ..... 850
23.2.1 Логика работы прерываний контроллера NVIC. ..... 851
23.2.2 Регистр разрешения прерываний ..... 855
23.2.3 Упрощенный доступ к регистрам контроллера прерываний ..... 855
23.2.4 Прерывания, срабатывающие по уровню сигнала ..... 859
23.2.5 Аппаратное и программное управление прерываниями ..... 859
23.2.6 Рекомендации по работе с контроллером прерываний ..... 860
24 Блок управления системой ..... 861
24.1 Упрощенный доступ к регистрам блока управления системой ..... 861
24.1.1 InterrupType->ACTLR ..... 862
24.1.2 SCB->CPUID ..... 862
24.1.3 SCB->ICSR ..... 863
24.1.4 SCB->VTOR ..... 864
24.1.5 SCB->AIRCR ..... 865
24.1.6 SCB->SCR ..... 866
24.1.7 SCB->CCR ..... 866
24.1.8 SCB->SHP[x] ..... 868
24.1.9 SCB->SHCSR ..... 869
24.1.10 SCB->CFSR ..... 870
24.1.11 SCB->HFSR ..... 873
24.1.12 SCB->MMFAR ..... 874
24.1.13 SCB->BFAR ..... 874
24.1.14 Рекомендации по программированию блока управления системой ..... 875
25 Типовая схема включения ..... 876
26 Электрические параметры микросхем ..... 878
27 Предельно-допустимые и предельные параметры ..... 880
28 Справочные данные ..... 882
29 Габаритный чертеж микросхемы ..... 884
30 Информация для заказа ..... 885

# 1 Структурная схема

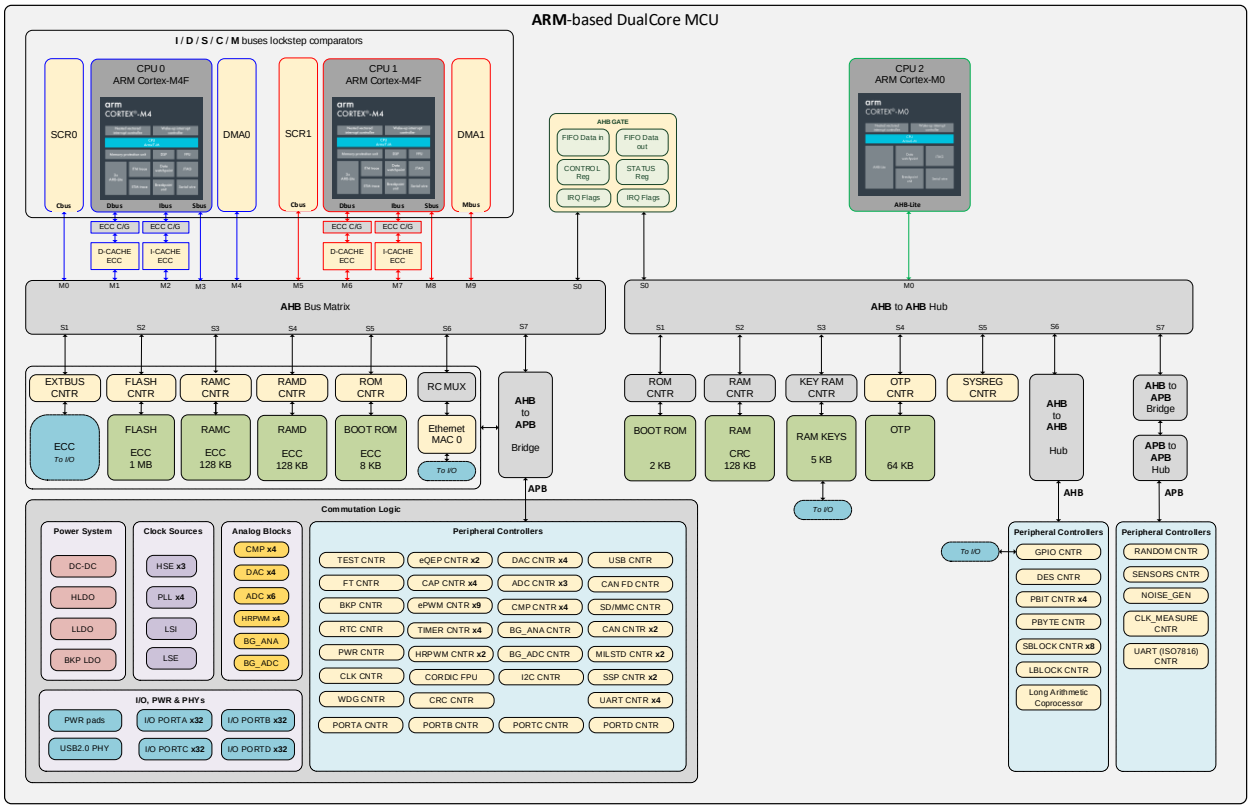



Рисунок 1 – Структурная схема микроконтроллера

# 2 Условно-графическое изображение

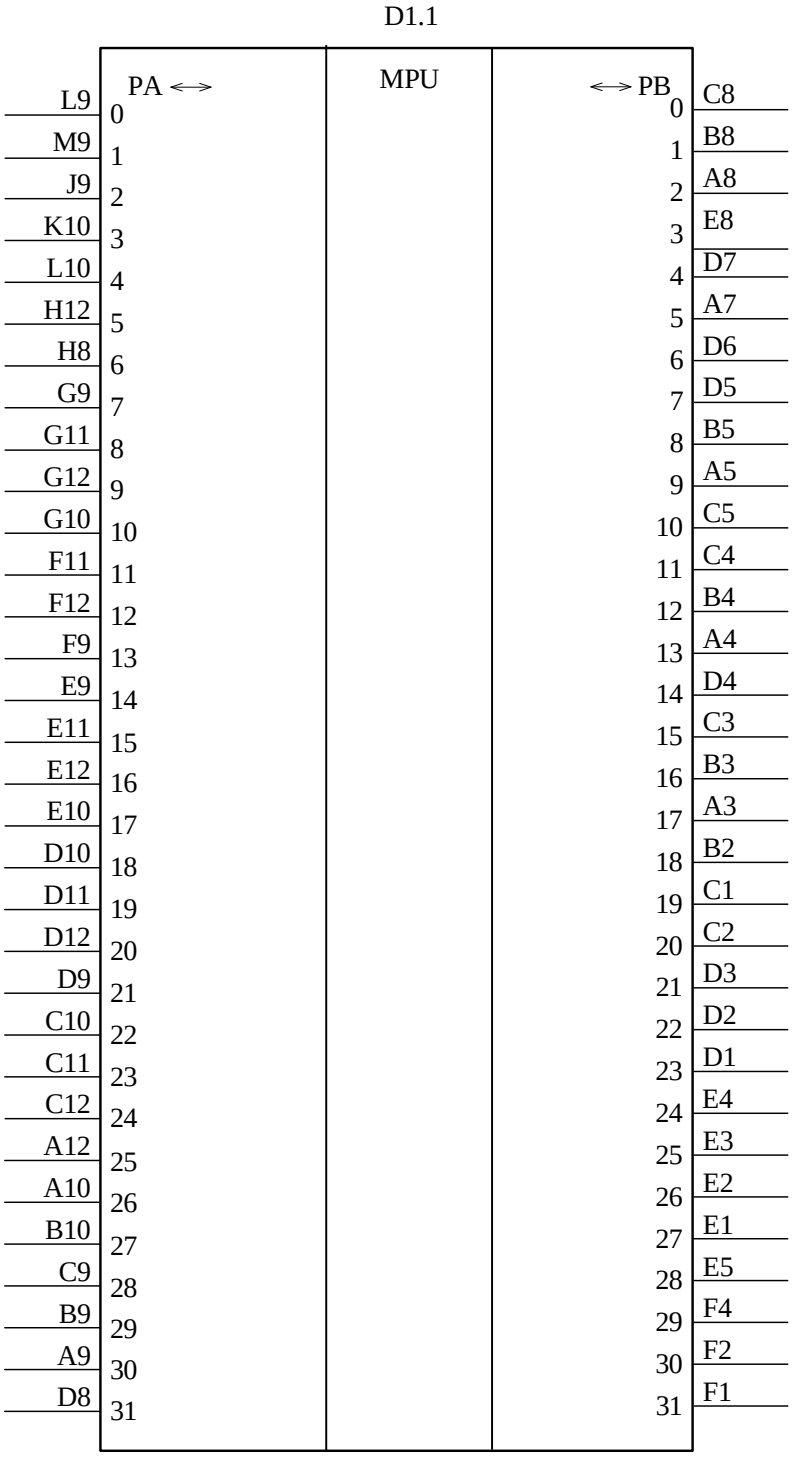



Рисунок 2 – Условно-графическое изображение (лист 1 из 2)

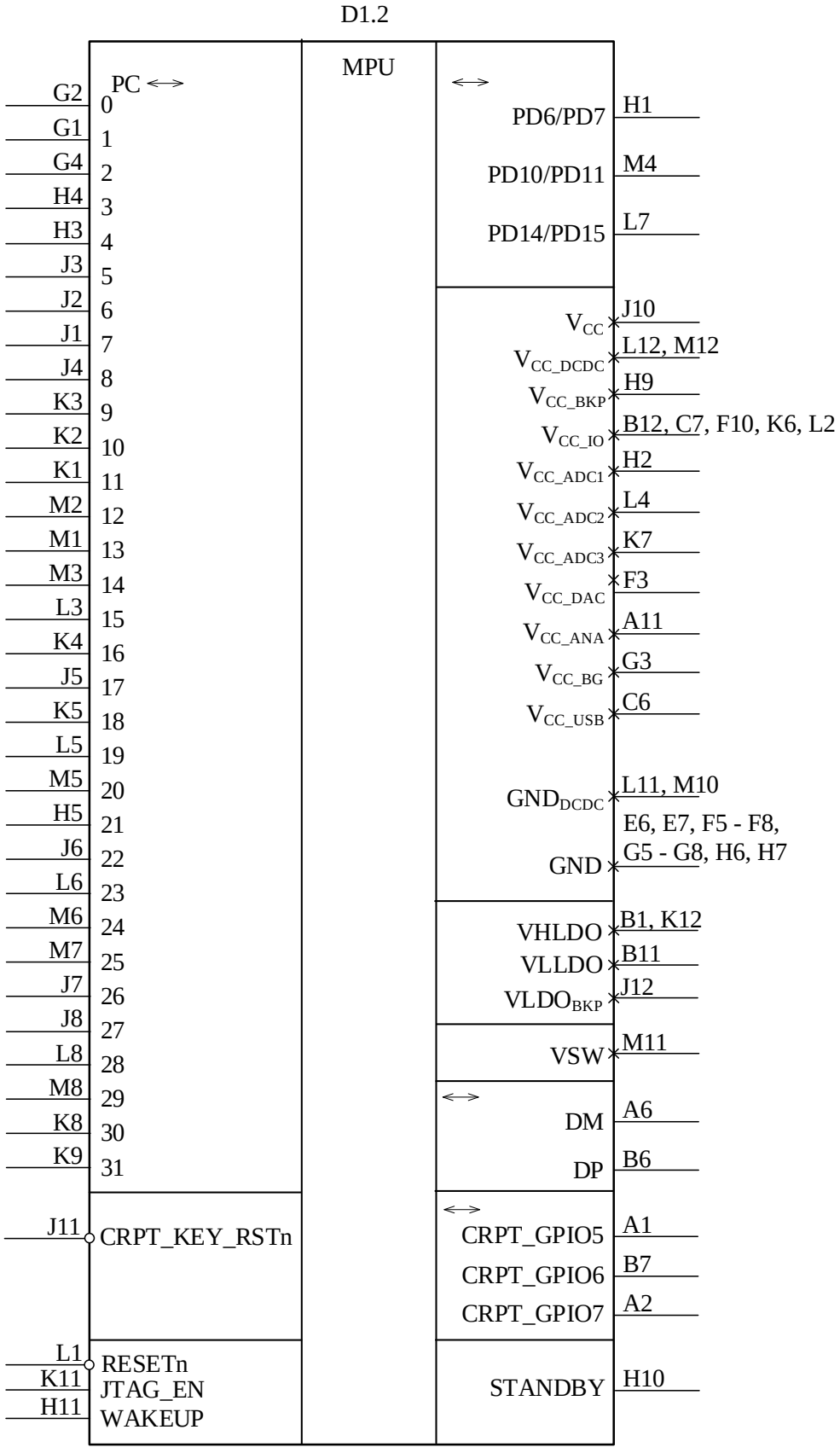



Рисунок 2 (лист 2 из 2)

# 3 Описание выводов микросхемы

Таблица 1 – Описание выводов микросхемы

| Номер вывода корпуса | Обозначение вывода | Тип вывода | Назначение вывода |
|---|---|---|---|
| A1 | CRPT_GPIO5 | IO | Вход/выход защищенного криптографического |
| A2 | CRPT_ GPIO7 | IO | сопроцессора |
| A3 | PB17 | IO | Пользовательский порт B ввода-вывода |
| A4 | PB13 | IO | |
| A5 | PB9 | IO | |
| A6 | DM | AIO | Вход/выход приемопередатчика USB |
| A7 | PB5 | IO | Пользовательский порт B ввода-вывода |
| A8 | PB2 | IO | |
| A9 | PA30 | IO | Пользовательский порт A ввода-вывода |
| A10 | PA26 | IO | |
| A11 | $V_{CC_ANA}$ | PWR | Питание 3,3 В схем генераторов и умножителей частоты |
| A12 | PA25 | IO | Пользовательский порт A ввода-вывода |
| B1 | VHLDO | AO | Выход напряжения встроенного регулятора HLDO |
| B2 | PB18 | IO | Пользовательский порт B ввода-вывода |
| B3 | PB16 | IO | |
| B4 | PB12 | IO | |
| B5 | PB8 | IO | |
| B6 | DP | AIO | Вход/выход приемопередатчика USB |
| B7 | CRPT_GPIO6 | IO | Вход/выход защищенного криптографического сопроцессора |
| B8 | PB1 | IO | Пользовательский порт B ввода-вывода |
| B9 | PA29 | IO | Пользовательский порт A ввода-вывода |
| B10 | PA27 | IO | |
| B11 | VLLDO | AO | Выход напряжения встроенного регулятора LLDO |
| B12 | $V_{CC_IO}$ | PWR | Питание 3,3 В схем ввода-вывода |
| C1 | PB19 | IO | Пользовательский порт B ввода-вывода |
| C2 | PB20 | IO | |
| C3 | PB15 | IO | |
| C4 | PB11 | IO | |
| C5 | PB10 | IO | |
| C6 | $V_{CC_USB}$ | PWR | Питание 3,3 В приемо-передатчиков USB |
| C7 | $V_{CC_IO}$ | PWR | Питание 3,3 В схем ввода-вывода |
| C8 | PB0 | IO | Пользовательский порт B ввода-вывода |
| C9 | PA28 | IO | Пользовательский порт A ввода-вывода |
| C10 | PA22 | IO | |
| C11 | PA23 | IO | |

| Номер вывода корпуса | Обозначение вывода | Тип вывода | Назначение вывода |
|---|---|---|---|
| C12 | PA24 | IO | |
| D1 | PB23 | IO | Пользовательский порт B ввода-вывода |
| D2 | PB22 | IO | |
| D3 | PB21 | IO | |
| D4 | PB14 | IO | |
| D5 | PB7 | IO | |
| D6 | PB6 | IO | |
| D7 | PB4 | IO | |
| D8 | PA31 | IO | Пользовательский порт A ввода-вывода |
| D9 | PA21 | IO | |
| D10 | PA18 | IO | |
| D11 | PA19 | IO | |
| D12 | PA20 | IO | |
| E1 | PB27 | IO | Пользовательский порт B ввода-вывода |
| E2 | PB26 | IO | |
| E3 | PB25 | IO | |
| E4 | PB24 | IO | |
| E5 | PB28 | IO | |
| E6, E7 | GND | PWR | Общий |
| E8 | PB3 | IO | Пользовательский порт B ввода-вывода |
| E9 | PA14 | IO | Пользовательский порт A ввода-вывода |
| E10 | PA17 | IO | |
| E11 | PA15 | IO | |
| E12 | PA16 | IO | |
| F1 | PB31 | IO | Пользовательский порт B ввода-вывода |
| F2 | PB30 | IO | |
| F3 | $V_{CC_DAC}$ | PWR | Питание 3,3 В схем ЦАП и компараторов |
| F4 | PB29 | IO | Пользовательский порт B ввода-вывода |
| F5, F6, F7, F8 | GND | PWR | Общий |
| F9 | PA13 | IO | Пользовательский порт A ввода-вывода |
| F10 | $V_{CC_IO}$ | PWR | Питание 3,3 В схем ввода-вывода |
| F11 | PA11 | IO | Пользовательский порт A ввода-вывода |
| F12 | PA12 | IO | |
| G1 | PC1 | IO | Пользовательский порт C ввода-вывода |
| G2 | PC0 | IO | |
| G3 | $V_{CC_BG}$ | PWR | Питание 3,3 В формирователя опорного напряжения АЦП, ЦАП и компараторов |
| G4 | PC2 | IO | Пользовательский порт C ввода-вывода |
| G5, G6, G7, G8, | GND | PWR | Общий |

| Номер вывода корпуса | Обозначение вывода | Тип вывода | Назначение вывода |
|---|---|---|---|
| G9 | PA7 | IO | Пользовательский порт A ввода-вывода |
| G10 | PA10 | IO | |
| G11 | PA8 | IO | |
| G12 | PA9 | IO | |
| H1 | PD6/PD7 | IO | Пользовательский порт D ввода-вывода |
| H2 | $V_{CC_ADC1}$ | PWR | Питание 3,3 В АЦП 1 |
| H3 | PC4 | IO | Пользовательский порт C ввода-вывода |
| H4 | PC3 | IO | |
| H5 | PC21 | IO | |
| H6, H7 | GND | PWR | Общий |
| H8 | PA6 | IO | Пользовательский порт A ввода-вывода |
| H9 | $V_{CC_BKP}$ | PWR | Питание 3,3 В встроенного регулятора формирования напряжения питания батарейного домена |
| H10 | STANDBY | O | Выход состояния встроенного регулятора:<br>– “0” – регулятор включен;<br>– “1” – регулятор программно выключен |
| H11 | WAKEUP | I | Вход сигнала запроса включения регулятора:<br>– “0” – нет запроса включения;<br>– “1” – есть запрос включения встроенного регулятора |
| H12 | PA5 | IO | Пользовательский порт A ввода-вывода |
| J1 | PC7 | IO | Пользовательский порт C ввода-вывода |
| J2 | PC6 | IO | |
| J3 | PC5 | IO | |
| J4 | PC8 | IO | |
| J5 | PC17 | IO | |
| J6 | PC22 | IO | |
| J7 | PC26 | IO | |
| J8 | PC27 | IO | |
| J9 | PA2 | IO | Пользовательский порт A ввода-вывода |
| J10 | $V_{CC}$ | PWR | Питание 3,3 В управляющего блока DCDC преобразователя |
| J11 | CRPT_KEY_RSTn | I | Вход защищенного криптографического сопроцессора |
| J12 | $VLDO_{BKP}$ | AO | Выход напряжения батарейного домена |
| K1 | PC11 | IO | Пользовательский порт C ввода-вывода |
| K2 | PC10 | IO | |
| K3 | PC9 | IO | |
| K4 | PC16 | IO | |
| K5 | PC18 | IO | |
| K6 | $V_{CC_IO}$ | PWR | Питание 3,3 В схем ввода-вывода |
| K7 | $V_{CC_ADC3}$ | PWR | Питание 3,3 В АЦП 3 |
| K8 | PC30 | IO | Пользовательский порт C ввода-вывода |
| K9 | PC31 | IO | |

| Номер вывода корпуса | Обозначение вывода | Тип вывода | Назначение вывода |
|---|---|---|---|
| K10 | PA3 | IO | Пользовательский порт A ввода-вывода |
| K11 | JTAG_EN | I | Вход выбора тестового режима:<br>– “0” – рабочий режим;<br>– “1” – тестовый режим |
| K12 | VHLDO | AO | Выход напряжения встроенного регулятора HLDO |
| L1 | RESETn | I | Вход сигнала сброса:<br>– “0” – сброс микросхемы;<br>– “1” – рабочий режим |
| L2 | $V_{CC_IO}$ | PWR | Питание 3,3 В схем ввода-вывода |
| L3 | PC15 | IO | Пользовательский порт C ввода-вывода |
| L4 | $V_{CC_ADC2}$ | PWR | Питание 3,3 В АЦП 2 |
| L5 | PC19 | IO | Пользовательский порт C ввода-вывода |
| L6 | PC23 | IO | |
| L7 | PD14/PD15 | IO | Пользовательский порт D ввода-вывода |
| L8 | PC28 | IO | Пользовательский порт C ввода-вывода |
| L9 | PA0 | IO | Пользовательский порт A ввода-вывода |
| L10 | PA4 | IO | |
| L11 | $GND_{DCDC}$ | PWR | Общий встроенного регулятора HLDO, DCDC преобразователя |
| L12 | $V_{CC_DCDC}$ | PWR | Питание 3,3 В встроенного регулятора HLDO, DCDC преобразователя |
| M1 | PC13 | IO | Пользовательский порт C ввода-вывода |
| M2 | PC12 | IO | |
| M3 | PC14 | IO | |
| M4 | PD10/PD11 | IO | Пользовательский порт D ввода-вывода |
| M5 | PC20 | IO | Пользовательский порт C ввода-вывода |
| M6 | PC24 | IO | |
| M7 | PC25 | IO | |
| M8 | PC29 | IO | |
| M9 | PA1 | IO | Пользовательский порт A ввода-вывода |
| M10 | $GND_{DCDC}$ | PWR | Общий встроенного регулятора HLDO, DCDC преобразователя |
| M11 | VSW | AO | Выход петли обратной связи DCDC преобразователя |
| M12 | $V_{CC_DCDC}$ | PWR | Питание 3,3 В встроенного регулятора HLDO, DCDC преобразователя |

Примечания

1 Обозначение типов выводов:

- AI – аналоговый вход/выход;
- AIO – аналоговый вход/выход;
- AO – аналоговый выход;
- I – цифровой вход;
- IO – цифровой вход/выход;
- O – цифровой выход;
- PWR – вывод «Питание» и «Общий»

2 Для корпусов МК 8307.144-АН3 и МК 6109.144-А монтажная площадка GND электрически соединена с выводами E6, E7, F5, F6, F7, F8, G5, G6, G7, G8, H6, H7.

Таблица 2 – Назначение выводов по блокам

| Номер вывода корпуса | Обозначение вывода | Тип вывода | Назначение вывода |
|---|---|---|---|
| **Порт A** | | | |
| A9 | PA30 | IO | Пользовательский порт A ввода-вывода |
| A10 | PA26 | IO | |
| A12 | PA25 | IO | |
| B9 | PA29 | IO | |
| B10 | PA27 | IO | |
| C9 | PA28 | IO | |
| C10 | PA22 | IO | |
| C11 | PA23 | IO | |
| C12 | PA24 | IO | |
| D8 | PA31 | IO | |
| D9 | PA21 | IO | |
| D10 | PA18 | IO | |
| D11 | PA19 | IO | |
| D12 | PA20 | IO | |
| E9 | PA14 | IO | |
| E10 | PA17 | IO | |
| E11 | PA15 | IO | |
| E12 | PA16 | IO | |
| F9 | PA13 | IO | |
| F11 | PA11 | IO | |
| F12 | PA12 | IO | |
| G9 | PA7 | IO | |
| G10 | PA10 | IO | |
| G11 | PA8 | IO | Пользовательский порт A ввода-вывода |
| G12 | PA9 | IO | |
| H8 | PA6 | IO | |
| H12 | PA5 | IO | |
| J9 | PA2 | IO | |
| K10 | PA3 | IO | |
| L9 | PA0 | IO | |
| L10 | PA4 | IO | |
| M9 | PA1 | IO | |
| **Порт B** | | | |
| A3 | PB17 | IO | Пользовательский порт B ввода-вывода |
| A4 | PB13 | IO | |
| A5 | PB9 | IO | |
| A7 | PB5 | IO | |

| Номер вывода корпуса | Обозначение вывода | Тип вывода | Назначение вывода |
|---|---|---|---|
| A8 | PB2 | IO | |
| B2 | PB18 | IO | |
| B3 | PB16 | IO | |
| B4 | PB12 | IO | |
| B5 | PB8 | IO | |
| B8 | PB1 | IO | |
| C1 | PB19 | IO | |
| C2 | PB20 | IO | |
| C3 | PB15 | IO | |
| C4 | PB11 | IO | |
| C5 | PB10 | IO | |
| C8 | PB0 | IO | |
| D1 | PB23 | IO | |
| D2 | PB22 | IO | |
| D3 | PB21 | IO | |
| D4 | PB14 | IO | |
| D5 | PB7 | IO | |
| D6 | PB6 | IO | |
| D7 | PB4 | IO | |
| E1 | PB27 | IO | |
| E2 | PB26 | IO | |
| E3 | PB25 | IO | |
| E4 | PB24 | IO | |
| E5 | PB28 | IO | Пользовательский порт B ввода-вывода |
| E8 | PB3 | IO | |
| F1 | PB31 | IO | |
| F2 | PB30 | IO | |
| F4 | PB29 | IO | |
| **Порт C** | | | |
| G1 | PC1 | IO | Пользовательский порт C ввода-вывода |
| G2 | PC0 | IO | |
| G4 | PC2 | IO | |
| H3 | PC4 | IO | |
| H4 | PC3 | IO | |
| H5 | PC21 | IO | |
| J1 | PC7 | IO | |
| J2 | PC6 | IO | |
| J3 | PC5 | IO | |
| J4 | PC8 | IO | |

| Номер вывода корпуса | Обозначение вывода | Тип вывода | Назначение вывода |
|---|---|---|---|
| J5 | PC17 | IO | |
| J6 | PC22 | IO | |
| J7 | PC26 | IO | |
| J8 | PC27 | IO | |
| K1 | PC11 | IO | |
| K2 | PC10 | IO | |
| K3 | PC9 | IO | |
| K4 | PC16 | IO | |
| K5 | PC18 | IO | |
| K8 | PC30 | IO | |
| K9 | PC31 | IO | |
| L3 | PC15 | IO | |
| L5 | PC19 | IO | |
| L6 | PC23 | IO | |
| L8 | PC28 | IO | |
| M1 | PC13 | IO | |
| M2 | PC12 | IO | |
| M3 | PC14 | IO | |
| M5 | PC20 | IO | |
| M6 | PC24 | IO | |
| M7 | PC25 | IO | |
| M8 | PC29 | IO | Пользовательский порт C ввода-вывода |
| **Порт D** | | | |
| H1 | PD6/PD7 | IO | Пользовательский порт D ввода-вывода |
| L7 | PD14/PD15 | IO | |
| M4 | PD10/PD11 | IO | |
| **Сигналы управления защищенного криптографического сопроцессора** | | | |
| A1 | CRPT_GPIO5 | IO | Входы/выходы защищенного криптографического сопроцессора |
| A2 | CRPT_ GPIO7 | IO | |
| B7 | CRPT_GPIO6 | IO | |
| J11 | CRPT_KEY_RSTn | I | Вход защищенного криптографического сопроцессора |
| **Сигналы управления** | | | |
| H10 | STANDBY | O | Выход состояния встроенного регулятора:<br>– “0” – регулятор включен;<br>– “1” – регулятор программно выключен |
| H11 | WAKEUP | I | Вход сигнала запроса включения регулятора:<br>– “0” – нет запроса включения;<br>– “1” – есть запрос включения встроенного регулятора |
| K11 | JTAG_EN | I | Вход выбора тестового режима:<br>– “0” – рабочий режим;<br>– “1” – тестовый режим |

| Номер вывода корпуса | Обозначение вывода | Тип вывода | Назначение вывода |
|---|---|---|---|
| L1 | RESETn | I | Вход сигнала сброса:<br>– “0” – сброс микросхемы;<br>– “1” – рабочий режим |
| **Сигналы блока USB** | | | |
| A6<br>B6 | DM<br>DP | AIO<br>AIO | Входы/выходы приемопередатчика USB |
| **Выделенные аналоговые сигналы** | | | |
| M11 | VSW | AO | Выход петли обратной связи DCDC преобразователя |
| **Выходы напряжения** | | | |
| B1, K12 | VHLDO | AO | Выход напряжения встроенного регулятора HLDO |
| B11 | VLLDO | AO | Выход напряжения встроенного регулятора LLDO |
| J12 | $VLDO_{BKP}$ | AO | Выход напряжения цифровой части батарейного домена |
| **Выводы земли и питания** | | | |
| A11 | $V_{CC_ANA}$ | PWR | Питание 3,3 В схем генераторов и умножителей частоты |
| B12, C7, F10, K6, L2 | $V_{CC_IO}$ | PWR | Питание 3,3 В схем ввода-вывода |
| C6 | $V_{CC_USB}$ | PWR | Питание 3,3 В приемо-передатчиков USB |
| F3 | $V_{CC_DAC}$ | PWR | Питание 3,3 В схем ЦАП и компараторов |
| G3 | $V_{CC_BG}$ | PWR | Питание 3,3 В формирователя опорного напряжения АЦП, ЦАП и компараторов |
| H2 | $V_{CC_ADC1}$ | PWR | Питание 3,3 В АЦП 1 |
| H9 | $V_{CC_BKP}$ | PWR | Питание 3,3 В встроенного регулятора формирования напряжения питания батарейного домена |
| J10 | $V_{CC}$ | PWR | Питание 3,3 В управляющего блока DCDC преобразователя |
| K7 | $V_{CC_ADC3}$ | PWR | Питание 3,3 В АЦП 3 |
| L4 | $V_{CC_ADC2}$ | PWR | Питание 3,3 В АЦП 2 |
| L12, M12 | $V_{CC_DCDC}$ | PWR | Питание 3,3 В встроенного регулятора HLDO, DCDC преобразователя |
| L11, M10 | $GND_{DCDC}$ | PWR | Общий встроенного регулятора HLDO, DCDC преобразователя |
| E6, E7, F5, F6, F7, F8, G5, G6, G7, G8, H6, H7 | GND | PWR | Общий |

Примечание – Обозначение типов выводов:
- AI – аналоговый вход/выход;
- AIO – аналоговый вход/выход;
- AO – аналоговый выход;
- I – цифровой вход;
- IO – цифровой вход/выход;
- O – цифровой выход;
- PWR – вывод «Питание» и «Общий»

Таблица 3 – Соответствие выводов порта А и функций с 1-й по 8-ю

| *№ вывода корпуса BGA144* | *№ КП кристалла* | SPECIAL | PORT | SDIO | A\|D | A\|D | CAN | UART | SSP | MIL | ETH |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | - | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| L9 | 89 | JTAGA_ nTRST | **PA0** | | ADDR[31] | DATA[32] | CAN0_RX | UART0_RX | SSP0_CLK | MIL0_TXAN | ETH_TXD_0 |
| M9 | 90 | JTAGA_ SWCLK/TCK | **PA1** | | ADDR[30] | DATA[33] | CAN0_TX | UART0_TX | SSP0_FS | MIL0_TXAP | ETH_TXD_1 |
| J9 | 91 | JTAGA_ SWDIO/TMS | **PA2** | | ADDR[29] | DATA[34] | CAN1_RX | UART1_RX | SSP0_RX | MIL0_ENA | ETH_TXD_2 |
| K10 | 92 | JTAGA_ TDI | **PA3** | | ADDR[28] | DATA[35] | CAN1_TX | UART1_TX | SSP0_TX | MIL0_RXAN | ETH_TXD_3 |
| L10 | 93 | JTAGA_ SWO/TDO | **PA4** | | ADDR[27] | DATA[36] | CAN0_RX | UART2_RX | SSP1_CLK | MIL0_RXAP | ETH_RXD_0 |
| H12 | 116 | | **PA5** | | ADDR[26] | DATA[37] | CAN0_TX | UART2_TX | SSP1_FS | MIL0_TXBN | ETH_RXD_1 |
| H8 | 117 | | **PA6** | | ADDR[25] | DATA[38] | CAN1_RX | UART3_RX | SSP1_RX | MIL0_TXBP | ETH_RXD_2 |
| G9 | 120 | | **PA7** | | ADDR[24] | DATA[39] | CAN1_TX | UART3_TX | SSP1_TX | MIL0_ENB | ETH_RXD_3 |
| G11 | 121 | | **PA8** | | ADDR[23] | READY[0] | CAN0_RX | UART0_RX | SSP0_CLK | MIL0_RXBN | ETH_TX_EN |
| G12 | 122 | | **PA9** | | ADDR[22] | READY[1] | CAN0_TX | UART0_TX | SSP0_FS | MIL0_RXBP | ETH_RX_DV |
| G10 | 123 | | **PA10** | | ADDR[21] | READY[2] | CAN1_RX | UART1_RX | SSP0_RX | MIL0_TXAN | ETH_RX_CLK |
| F11 | 126 | | **PA11** | | ADDR[20] | READY[3] | CAN1_TX | UART1_TX | SSP0_TX | MIL0_TXAP | ETH_TX_CLK |
| F12 | 127 | | **PA12** | | ADDR[19] | READY[4] | CAN0_RX | UART2_RX | SSP1_CLK | MIL0_ENA | ETH_COL |
| F9 | 128 | | **PA13** | | ADDR[18] | READY[5] | CAN0_TX | UART2_TX | SSP1_FS | MIL0_RXAN | ETH_CRS |
| E9 | 129 | | **PA14** | | ADDR[17] | READY[6] | CAN1_RX | UART3_RX | SSP1_RX | MIL0_RXAP | ETH_RX_ERR |
| E11 | 131 | | **PA15** | | ADDR[16] | READY[7] | CAN1_TX | UART3_TX | SSP1_TX | MIL0_TXBN | ETH_RMII_CLK |
| E12 | 132 | MODE0 | **PA16** | | ADDR[15] | DATA[40] | CAN0_RX | UART0_RX | SSP0_CLK | MIL0_TXBP | ETH_MDIO |
| E10 | 134 | MODE1 | **PA17** | | ADDR[14] | DATA[41] | CAN0_TX | UART0_TX | SSP0_FS | MIL0_ENB | ETH_MDC |
| D10 | 135 | MODE2 | **PA18** | | ADDR[13] | DATA[42] | CAN1_RX | UART1_RX | SSP0_RX | MIL0_RXBN | |
| D11 | 138 | MODE3 | **PA19** | | ADDR[12] | DATA[43] | CAN1_TX | UART1_TX | SSP0_TX | MIL0_RXBP | |
| D12 | 139 | MODE4 | **PA20** | | ADDR[11] | DATA[44] | CAN0_RX | UART2_RX | SSP1_CLK | MIL0_TXAN | ETH_RX_DV |
| D9 | 141 | MODE5 | **PA21** | | ADDR[10] | DATA[45] | CAN0_TX | UART2_TX | SSP1_FS | MIL0_TXAP | ETH_TX_EN |
| C10 | 142 | | **PA22** | | ADDR[09] | DATA[46] | CAN1_RX | UART3_RX | SSP1_RX | MIL0_ENA | ETH_RX_CLK |
| C11 | 144 | | **PA23** | | ADDR[08] | DATA[47] | CAN1_TX | UART3_TX | SSP1_TX | MIL0_RXAN | ETH_TX_CLK |
| C12 | 145 | | **PA24** | | ADDR[07] | READY[0] | CAN0_RX | UART0_RX | SSP0_CLK | MIL0_RXAP | ETH_TXD_0 |
| A12 | 151 | | **PA25** | | ADDR[06] | READY[1] | CAN0_TX | UART0_TX | SSP0_FS | MIL0_TXBN | ETH_TXD_1 |
| A10 | 152 | | **PA26** | | ADDR[05] | READY[2] | CAN1_RX | UART1_RX | SSP0_RX | MIL0_TXBP | ETH_TXD_2 |
| B10 | 154 | | **PA27** | | ADDR[04] | READY[3] | CAN1_TX | UART1_TX | SSP0_TX | MIL0_ENB | ETH_TXD_3 |
| C9 | 155 | | **PA28** | | ADDR[03] | READY[4] | CAN0_RX | UART2_RX | SSP1_CLK | MIL0_RXBN | ETH_RXD_0 |
| B9 | 156 | | **PA29** | | ADDR[02] | READY[5] | CAN0_TX | UART2_TX | SSP1_FS | MIL0_RXBP | ETH_RXD_1 |
| A9 | 157 | | **PA30** | | ADDR[01] | READY[6] | CAN1_RX | UART3_RX | SSP1_RX | MIL0_TXAN | ETH_RXD_2 |
| D8 | 159 | | **PA31** | | ADDR[00] | READY[7] | CAN1_TX | UART3_TX | SSP1_TX | MIL0_TXAP | ETH_RXD_3 |

Таблица 4 – Соответствие выводов порта А и функций с 9-й по 15-ю

| *№ вывода корпуса BGA144* | *№ КП кристалла* | SPECIAL | PORT | I2C CAN-FD | TMR | PWM | PWM | CAP | CAP COMP ADC | QEP | ANALOG |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | - | 0 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | |
| L9 | 89 | JTAGA_ nTRST | **PA0** | I2C_SCL | TMR0_CH1P | EPWM0_A | EPWMx_SYNCI | ECAP3_PWM | ADC0_ER0 | EQEP0_A | |
| M9 | 90 | JTAGA_ SWCLK/TCK | **PA1** | I2C_SDA | TMR0_CH1N | EPWM0_B | EPWMx_nTZ1_EXT | ECAP2_PWM | ADC0_ER1 | EQEP0_B | |
| J9 | 91 | JTAGA_ SWDIO/TMS | **PA2** | CANFD_TX | TMR0_CH2P | EPWM1_A | EPWMx_nTZ2_EXT | ECAP1_PWM | ADC1_ER0 | EQEP0_I | |
| K10 | 92 | JTAGA_ TDI | **PA3** | CANFD_RX | TMR0_CH2N | EPWM1_B | EPWM2_A | ECAP0_PWM | ADC1_ER1 | EQEP0_S | |
| L10 | 93 | JTAGA_ SWO/TDO | **PA4** | CANFD_TX | TMR0_CH3P | EPWM2_A | EPWM2_B | ECAP0_PWM | ADC2_ER0 | EQEP1_A | |
| H12 | 116 | | **PA5** | CANFD_RX | TMR0_CH3N | EPWM2_B | EPWM3_A | ECAP1_PWM | ADC2_ER1 | EQEP1_B | LSE_OSCIN |
| H8 | 117 | | **PA6** | I2C_SCL | TMR0_CH4P | EPWMx_nTZ1_EXT | EPWM3_B | ECAP2_PWM | | EQEP1_I | LSE_OSCOUT |
| G9 | 120 | | **PA7** | I2C_SDA | TMR0_CH4N | EPWMx_SYNCI | | ECAP3_PWM | | EQEP1_S | |
| G11 | 121 | | **PA8** | CANFD_TX | TMR0_ETR | EPWM3_A | EPWM0_A | ECAP3_PWM | COMP0_OUT | EQEP0_A | |
| G12 | 122 | | **PA9** | CANFD_RX | TMR0_BRK | EPWM3_B | EPWM0_B | ECAP2_PWM | COMP1_OUT | EQEP0_B | |
| G10 | 123 | | **PA10** | CANFD_TX | TMR1_ETR | EPWM4_A | EPWM1_A | ECAP1_PWM | COMP2_OUT | EQEP0_I | ADC_BG_EXTREF0 |
| F11 | 126 | | **PA11** | CANFD_RX | TMR1_BRK | EPWM4_B | EPWM1_B | ECAP0_PWM | COMP3_OUT | EQEP0_S | ADC_BG_EXTREF1 |
| F12 | 127 | | **PA12** | I2C_SCL | TMR2_ETR | EPWM5_A | EPWM4_A | ECAP0_PWM | | EQEP1_A | ADC_BG_EXTREF2 |
| F9 | 128 | | **PA13** | I2C_SDA | TMR2_BRK | EPWM5_B | EPWM4_B | ECAP1_PWM | | EQEP1_B | ADC_BG_EXTREF3 |
| E9 | 129 | | **PA14** | CANFD_TX | TMR3_ETR | EPWM6_A | EPWM5_A | ECAP2_PWM | ECAPx_SYNCI | EQEP1_I | ADC_BG_EXTREF4 |
| E11 | 131 | | **PA15** | CANFD_RX | TMR3_BRK | EPWM6_B | EPWM5_B | ECAP3_PWM | ECAPx_SYNCO | EQEP1_S | |
| E12 | 132 | | **PA16** | CANFD_TX | TMR1_CH1P | EPWM7_A | EPWM6_A | ECAP3_PWM | ADC0_ER0 | EQEP0_A | |
| E10 | 134 | | **PA17** | CANFD_RX | TMR1_CH1N | EPWM7_B | EPWM6_B | ECAP2_PWM | ADC0_ER1 | EQEP0_B | |
| D10 | 135 | | **PA18** | I2C_SCL | TMR1_CH2P | EPWM8_A | EPWM7_A | ECAP1_PWM | ADC1_ER0 | EQEP0_I | |
| D11 | 138 | | **PA19** | I2C_SDA | TMR1_CH2N | EPWM8_B | EPWM7_B | ECAP0_PWM | ADC1_ER1 | EQEP0_S | HRPWM3 |
| D12 | 139 | | **PA20** | CANFD_TX | TMR1_CH3P | EPWMx_nTZ2_EXT | EPWM8_A | ECAP0_PWM | ADC2_ER0 | EQEP1_A | HRPWM2 |
| D9 | 141 | | **PA21** | CANFD_RX | TMR1_CH3N | EPWMx_nTZ3_EXT | EPWM8_B | ECAP1_PWM | ADC2_ER1 | EQEP1_B | |
| C10 | 142 | | **PA22** | CANFD_TX | TMR1_CH4P | EPWMx_nTZ4_EXT | EPWMx_nTZ3_EXT | ECAP2_PWM | | EQEP1_I | |
| C11 | 144 | | **PA23** | CANFD_RX | TMR1_CH4N | EPWMx_nTZ5_EXT | EPWMx_nTZ4_EXT | ECAP3_PWM | | EQEP1_S | HRPWM1 |
| C12 | 145 | | **PA24** | I2C_SCL | TMR2_CH1P | EPWMx_nTZ6_EXT | EPWMx_nTZ5_EXT | ECAP3_PWM | COMP0_OUT | EQEP0_A | HRPWM0 |
| A12 | 151 | | **PA25** | I2C_SDA | TMR2_CH1N | EPWMx_SYNCI | EPWMx_nTZ6_EXT | ECAP2_PWM | COMP1_OUT | EQEP0_B | |
| A10 | 152 | | **PA26** | CANFD_TX | TMR2_CH2P | EPWM6_A | EPWMx_SYNCI | ECAP1_PWM | COMP2_OUT | EQEP0_I | |
| B10 | 154 | | **PA27** | CANFD_RX | TMR2_CH2N | EPWM6_B | EPWM7_A | ECAP0_PWM | COMP3_OUT | EQEP0_S | HSE0_OSCIN |
| C9 | 155 | | **PA28** | CANFD_TX | TMR2_CH3P | EPWM7_A | EPWM7_B | ECAP0_PWM | ECAPx_SYNCI | EQEP1_A | HSE0_OSCOUT |
| B9 | 156 | | **PA29** | CANFD_RX | TMR2_CH3N | EPWM7_B | EPWM8_A | ECAP1_PWM | ECAPx_SYNCO | EQEP1_B | HSE1_OSCIN |
| A9 | 157 | | **PA30** | I2C_SCL | TMR2_CH4P | EPWM8_A | EPWM8_B | ECAP2_PWM | ADCx_SYNCI | EQEP1_I | HSE1_OSCOUT |
| D8 | 159 | | **PA31** | I2C_SDA | TMR2_CH4N | EPWM8_B | EPWMx_SYNCI | ECAP3_PWM | ADCx_SYNCO | EQEP1_S | |

Таблица 5 – Соответствие выводов порта B и функций с 1-й по 8-ю

| *№ вывода корпуса BGA144* | *№ КП кристалла* | SPECIAL | PORT | SDIO | A\|D | A\|D | CAN | UART | SSP | MIL | ETH |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | - | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| C8 | 160 | | **PB0** | SDIO_SCK | BEn[0] | BWEn[0] | CAN0_RX | UART0_RX | SSP0_CLK | MIL0_TXAN | ETH_COL |
| B8 | 161 | | **PB1** | SDIO_SCD | BEn[1] | BWEn[1] | CAN0_TX | UART0_TX | SSP0_FS | MIL0_TXAP | ETH_CRS |
| A8 | 162 | | **PB2** | SDIO_SDD[0] | BEn[2] | BWEn[2] | CAN1_RX | UART1_RX | SSP0_RX | MIL0_ENA | ETH_RX_CLK |
| E8 | 164 | | **PB3** | SDIO_SDD[1] | BEn[3] | BWEn[3] | CAN1_TX | UART1_TX | SSP0_TX | MIL0_RXAN | ETH_TX_CLK |
| D7 | 165 | JTAGB_ nTRST | **PB4** | SDIO_SDD[2] | BEn[4] | BWEn[4] | CAN0_RX | UART2_RX | SSP1_CLK | MIL0_RXAP | ETH_MDIO |
| A7 | 168 | JTAGB_ SWCLK/TCK | **PB5** | SDIO_SDD[3] | BEn[5] | BWEn[0] | CAN0_TX | UART2_TX | SSP1_FS | MIL0_TXBN | ETH_MDC |
| D6 | 179 | JTAGB_ SWDIO/TMS | **PB6** | SDIO_SDD[4] | CSn[0] | CS[0] | CAN1_RX | UART3_RX | SSP1_RX | MIL0_TXBP | ETH_RX_DV |
| D5 | 180 | JTAGB_ TDI | **PB7** | SDIO_SDD[5] | CSn[1] | CS[1] | CAN1_TX | UART3_TX | SSP1_TX | MIL0_ENB | ETH_TX_EN |
| B5 | 182 | JTAGB_ SWO/TDO | **PB8** | SDIO_SDD[6] | CSn[2] | CS[2] | CAN0_RX | UART0_RX | SSP0_CLK | MIL0_RXBN | ETH_RXD_0 |
| A5 | 183 | | **PB9** | SDIO_SDD[7] | CSn[3] | CS[4] | CAN0_TX | UART0_TX | SSP0_FS | MIL0_RXBP | ETH_RXD_1 |
| C5 | 185 | | **PB10** | SDIO_SCO | CSn[4] | CS[4] | CAN1_RX | UART1_RX | SSP0_RX | MIL0_TXAN | ETH_RXD_2 |
| C4 | 186 | | **PB11** | SDIO_SCK | CSn[5] | CS[5] | CAN1_TX | UART1_TX | SSP0_TX | MIL0_TXAP | ETH_RXD_3 |
| B4 | 188 | | **PB12** | SDIO_SCD | CSn[6] | CS[6] | CAN0_RX | UART2_RX | SSP1_CLK | MIL0_ENA | ETH_TXD_0 |
| A4 | 189 | | **PB13** | SDIO_SDD[0] | CSn[7] | CS[7] | CAN0_TX | UART2_TX | SSP1_FS | MIL0_RXAN | ETH_TXD_1 |
| D4 | 191 | | **PB14** | SDIO_SDD[1] | WEn | DATA[00] | CAN1_RX | UART3_RX | SSP1_RX | MIL0_RXAP | ETH_TXD_2 |
| C3 | 195 | | **PB15** | SDIO_SDD[2] | OEn | DATA[01] | CAN1_TX | UART3_TX | SSP1_TX | MIL0_TXBN | ETH_TXD_3 |
| B3 | 196 | | **PB16** | SDIO_SDD[3] | CLOCK | DATA[02] | CAN0_RX | UART0_RX | SSP0_CLK | MIL0_TXBP | ETH_RMII_CLK |
| A3 | 197 | | **PB17** | SDIO_SDD[4] | OCLK | DATA[03] | CAN0_TX | UART0_TX | SSP0_FS | MIL0_ENB | ETH_RX_ERR |
| B2 | 198 | | **PB18** | SDIO_SDD[5] | ADDR[6] | DATA[04] | CAN1_RX | UART1_RX | SSP0_RX | MIL0_RXBN | |
| C1 | 4 | | **PB19** | SDIO_SDD[6] | ADDR[5] | DATA[05] | CAN1_TX | UART1_TX | SSP0_TX | MIL0_RXBP | |
| C2 | 5 | | **PB20** | SDIO_SDD[7] | ADDR[4] | DATA[06] | CAN0_RX | UART2_RX | SSP1_CLK | MIL0_TXAN | ETH_RX_CLK |
| D3 | 7 | | **PB21** | SDIO_SDOE | ADDR[3] | DATA[07] | CAN0_TX | UART2_TX | SSP1_FS | MIL0_TXAP | ETH_TX_CLK |
| D2 | 8 | | **PB22** | SDIO_SCK | ADDR[2] | DATA[08] | CAN1_RX | UART3_RX | SSP1_RX | MIL0_ENA | ETH_RX_DV |
| D1 | 12 | | **PB23** | SDIO_SCD | ADDR[1] | DATA[09] | CAN1_TX | UART3_TX | SSP1_TX | MIL0_RXAN | ETH_TX_EN |
| E4 | 14 | | **PB24** | SDIO_SDD[0] | READY[0] | DATA[10] | CAN0_RX | UART0_RX | SSP0_CLK | MIL0_RXAP | ETH_TXD_0 |
| E3 | 15 | | **PB25** | SDIO_SDD[1] | READY[1] | DATA[11] | CAN0_TX | UART0_TX | SSP0_FS | MIL0_TXBN | ETH_TXD_1 |
| E2 | 16 | | **PB26** | SDIO_SDD[2] | READY[2] | DATA[12] | CAN1_RX | UART1_RX | SSP0_RX | MIL0_TXBP | ETH_TXD_2 |
| E1 | 17 | | **PB27** | SDIO_SDD[3] | READY[3] | DATA[13] | CAN1_TX | UART1_TX | SSP0_TX | MIL0_ENB | ETH_TXD_3 |
| E5 | 18 | | **PB28** | SDIO_SDD[4] | READY[4] | DATA[14] | CAN0_RX | UART2_RX | SSP1_CLK | MIL0_RXBN | ETH_RXD_0 |
| F4 | 19 | | **PB29** | SDIO_SDD[5] | READY[5] | DATA[15] | CAN0_TX | UART2_TX | SSP1_FS | MIL0_RXBP | ETH_RXD_1 |
| F2 | 20 | | **PB30** | SDIO_SDD[6] | READY[6] | DATA[16] | CAN1_RX | UART3_RX | SSP1_RX | MIL0_TXAN | ETH_RXD_2 |
| F1 | 21 | | **PB31** | SDIO_SDD[7] | READY[7] | DATA[17] | CAN1_TX | UART3_TX | SSP1_TX | MIL0_TXAP | ETH_RXD_3 |

Таблица 6 – Соответствие выводов порта B и функций с 9-й по 15-ю

| *№ вывода корпуса BGA144* | *№ КП кристалла* | SPECIAL | PORT | I2C CAN-FD | TMR | PWM | PWM | CAP | CAP COMP ADC | QEP | ANALOG |
|---|---|---|---|---|---|---|---|---|---|---|---|
| - | - | 9 | 10 | 11 | 12 | 13 | 14 | 15 | | | |
| C8 | 160 | | **PB0** | I2C_SCL | TMR0_ETR | EPWMx_SYNCI | EPWM0_A | ECAP3_PWM | ADC0_ER0 | EQEP0_A | |
| B8 | 161 | | **PB1** | I2C_SDA | TMR0_BRK | EPWMx_nTZ1_EXT | EPWM0_B | ECAP2_PWM | ADC0_ER1 | EQEP0_B | |
| A8 | 162 | | **PB2** | CANFD_TX | TMR1_ETR | EPWMx_nTZ2_EXT | EPWM1_A | ECAP1_PWM | ADC1_ER0 | EQEP0_I | |
| E8 | 164 | | **PB3** | CANFD_RX | TMR1_BRK | EPWM2_A | EPWM1_B | ECAP0_PWM | ADC1_ER1 | EQEP0_S | |
| D7 | 165 | | **PB4** | CANFD_TX | TMR2_ETR | EPWM2_B | EPWM2_A | ECAP0_PWM | ADC2_ER0 | EQEP1_A | COMP3_VIN3 |
| A7 | 168 | | **PB5** | CANFD_RX | TMR2_BRK | EPWM3_A | EPWM2_B | ECAP1_PWM | ADC2_ER1 | EQEP1_B | COMP3_VIN2 |
| D6 | 179 | | **PB6** | I2C_SCL | TMR3_ETR | EPWM3_B | EPWMx_nTZ1_EXT | ECAP2_PWM | | EQEP1_I | COMP3_VIN1 |
| D5 | 180 | | **PB7** | I2C_SDA | TMR3_BRK | EPWMx_SYNCI | | ECAP3_PWM | | EQEP1_S | COMP2_VIN3 |
| B5 | 182 | | **PB8** | CANFD_TX | TMR3_CH1P | EPWM0_A | EPWM3_A | ECAP3_PWM | COMP0_OUT | EQEP0_A | COMP2_VIN2 |
| A5 | 183 | | **PB9** | CANFD_RX | TMR3_CH1N | EPWM0_B | EPWM3_B | ECAP2_PWM | COMP1_OUT | EQEP0_B | COMP2_VIN1 |
| C5 | 185 | | **PB10** | CANFD_TX | TMR3_CH2P | EPWM1_A | EPWM4_A | ECAP1_PWM | COMP2_OUT | EQEP0_I | COMP1_VIN3 |
| C4 | 186 | | **PB11** | CANFD_RX | TMR3_CH2N | EPWM1_B | EPWM4_B | ECAP0_PWM | COMP3_OUT | EQEP0_S | COMP1_VIN2 |
| B4 | 188 | | **PB12** | I2C_SCL | TMR3_CH3P | EPWM4_A | EPWM5_A | ECAP0_PWM | ECAPx_SYNCI | EQEP1_A | COMP1_VIN1 |
| A4 | 189 | | **PB13** | I2C_SDA | TMR3_CH3N | EPWM4_B | EPWM5_B | ECAP1_PWM | ECAPx_SYNCO | EQEP1_B | COMP0_VIN3 |
| D4 | 191 | | **PB14** | CANFD_TX | TMR3_CH4P | EPWM5_A | EPWM6_A | ECAP2_PWM | | EQEP1_I | COMP0_VIN2 |
| C3 | 195 | | **PB15** | CANFD_RX | TMR3_CH4N | EPWM5_B | EPWM6_B | ECAP3_PWM | | EQEP1_S | COMP0_VIN1 |
| B3 | 196 | | **PB16** | CANFD_TX | TMR0_CH1P | EPWM6_A | EPWM7_A | ECAP3_PWM | ADC0_ER0 | EQEP0_A | DAC3_VRFP1 |
| A3 | 197 | | **PB17** | CANFD_RX | TMR0_CH1N | EPWM6_B | EPWM7_B | ECAP2_PWM | ADC0_ER1 | EQEP0_B | DAC3_VRFP0 |
| B2 | 198 | | **PB18** | I2C_SCL | TMR0_CH2P | EPWM7_A | EPWM8_A | ECAP1_PWM | ADC1_ER0 | EQEP0_I | DAC2_VRFP1 |
| C1 | 4 | | **PB19** | I2C_SDA | TMR0_CH2N | EPWM7_B | EPWM8_B | ECAP0_PWM | ADC1_ER1 | EQEP0_S | DAC2_VRFP0 |
| C2 | 5 | | **PB20** | CANFD_TX | TMR0_CH3P | EPWM8_A | EPWMx_nTZ2_EXT | ECAP0_PWM | ADC2_ER0 | EQEP1_A | DAC1_VRFN1 |
| D3 | 7 | | **PB21** | CANFD_RX | TMR0_CH3N | EPWM8_B | EPWMx_nTZ3_EXT | ECAP1_PWM | ADC2_ER1 | EQEP1_B | DAC1_VRFP1 |
| D2 | 8 | | **PB22** | CANFD_TX | TMR0_CH4P | EPWMx_nTZ3_EXT | EPWMx_nTZ4_EXT | ECAP2_PWM | | EQEP1_I | DAC3_OUT |
| D1 | 12 | | **PB23** | CANFD_RX | TMR0_CH4N | EPWMx_nTZ4_EXT | EPWMx_nTZ5_EXT | ECAP3_PWM | | EQEP1_S | DAC2_OUT |
| E4 | 14 | | **PB24** | I2C_SCL | TMR1_CH1P | EPWMx_nTZ5_EXT | EPWMx_nTZ6_EXT | ECAP3_PWM | COMP0_OUT | EQEP0_A | DAC1_VRFN0 |
| E3 | 15 | | **PB25** | I2C_SDA | TMR1_CH1N | EPWMx_nTZ6_EXT | EPWMx_SYNCI | ECAP2_PWM | COMP1_OUT | EQEP0_B | DAC1_VRFP0 |
| E2 | 16 | | **PB26** | CANFD_TX | TMR1_CH2P | EPWMx_SYNCI | EPWM6_A | ECAP1_PWM | COMP2_OUT | EQEP0_I | DAC1_OUT |
| E1 | 17 | | **PB27** | CANFD_RX | TMR1_CH2N | EPWM7_A | EPWM6_B | ECAP0_PWM | COMP3_OUT | EQEP0_S | DAC0_OUT |
| E5 | 18 | | **PB28** | CANFD_TX | TMR1_CH3P | EPWM7_B | EPWM7_A | ECAP0_PWM | ECAPx_SYNCI | EQEP1_A | DAC0_VRFN1 |
| F4 | 19 | | **PB29** | CANFD_RX | TMR1_CH3N | EPWM8_A | EPWM7_B | ECAP1_PWM | ECAPx_SYNCO | EQEP1_B | DAC0_VRFP1 |
| F2 | 20 | | **PB30** | I2C_SCL | TMR1_CH4P | EPWM8_B | EPWM8_A | ECAP2_PWM | ADCx_SYNCI | EQEP1_I | DAC0_VRFN0 |
| F1 | 21 | | **PB31** | I2C_SDA | TMR1_CH4N | EPWMx_SYNCI | EPWM8_B | ECAP3_PWM | ADCx_SYNCO | EQEP1_S | DAC0_VRFP0 |

Таблица 7 – Соответствие выводов порта С и функций с 1-й по 8-ю

| *№ вывода корпуса BGA144* | *№ КП кристалла* | SPECIAL | PORT | SDIO | A\|D | A\|D DAC | CAN | UART | SSP | MIL | ETH |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | - | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| G2 | 29 | | **PC0** | | DATA[00] | BEn[0] | CAN0_RX | UART0_RX | SSP0_CLK | MIL1_TXAN | ETH_TXD_0 |
| G1 | 30 | | **PC1** | | DATA[01] | BEn[1] | CAN0_TX | UART0_TX | SSP0_FS | MIL1_TXAP | ETH_TXD_1 |
| G4 | 31 | | **PC2** | | DATA[02] | BEn[2] | CAN1_RX | UART1_RX | SSP0_RX | MIL1_ENA | ETH_TXD_2 |
| H4 | 32 | | **PC3** | | DATA[03] | BEn[3] | CAN1_TX | UART1_TX | SSP0_TX | MIL1_RXAN | ETH_TXD_3 |
| H3 | 40 | | **PC4** | | DATA[04] | BEn[4] | CAN0_RX | UART2_RX | SSP1_CLK | MIL1_RXAP | ETH_RXD_0 |
| J3 | 41 | | **PC5** | | DATA[05] | BEn[5] | CAN0_TX | UART2_TX | SSP1_FS | MIL1_TXBN | ETH_RXD_1 |
| J2 | 42 | | **PC6** | | DATA[06] | CSn[0] | CAN1_RX | UART3_RX | SSP1_RX | MIL1_TXBP | ETH_RXD_2 |
| J1 | 43 | | **PC7** | | DATA[07] | CSn[1] | CAN1_TX | UART3_TX | SSP1_TX | MIL1_ENB | ETH_RXD_3 |
| J4 | 44 | | **PC8** | | DATA[08] | CSn[2] | CAN0_RX | UART0_RX | SSP0_CLK | MIL1_RXBN | ETH_TX_EN |
| K3 | 45 | | **PC9** | | DATA[09] | CSn[3] | CAN0_TX | UART0_TX | SSP0_FS | MIL1_RXBP | ETH_RX_DV |
| K2 | 46 | | **PC10** | | DATA[10] | CSn[4] | CAN1_RX | UART1_RX | SSP0_RX | MIL1_TXAN | ETH_RX_CLK |
| K1 | 48 | | **PC11** | | DATA[11] | CSn[5] | CAN1_TX | UART1_TX | SSP0_TX | MIL1_TXAP | ETH_TX_CLK |
| M2 | 51 | | **PC12** | | DATA[12] | CSn[6] | CAN0_RX | UART2_RX | SSP1_CLK | MIL1_ENA | ETH_COL |
| M1 | 52 | | **PC13** | | DATA[13] | CSn[7] | CAN0_TX | UART2_TX | SSP1_FS | MIL1_RXAN | ETH_CRS |
| M3 | 54 | | **PC14** | | DATA[14] | WEn | CAN1_RX | UART3_RX | SSP1_RX | MIL1_RXAP | ETH_RX_ERR |
| L3 | 55 | | **PC15** | | DATA[15] | OEn | CAN1_TX | UART3_TX | SSP1_TX | MIL1_TXBN | ETH_RMII_CLK |
| K4 | 56 | | **PC16** | | DATA[16] | CLOCK | CAN0_RX | UART0_RX | SSP0_CLK | MIL1_TXBP | ETH_MDIO |
| J5 | 63 | | **PC17** | | DATA[17] | OCLK | CAN0_TX | UART0_TX | SSP0_FS | MIL1_ENB | ETH_MDC |
| K5 | 64 | | **PC18** | | DATA[18] | | CAN1_RX | UART1_RX | SSP0_RX | MIL1_RXBN | |
| L5 | 65 | | **PC19** | | DATA[19] | DACx_SYNC_en[1] | CAN1_TX | UART1_TX | SSP0_TX | MIL1_RXBP | |
| M5 | 66 | | **PC20** | | DATA[20] | | CAN0_RX | UART2_RX | SSP1_CLK | MIL1_TXAN | ETH_RX_DV |
| H5 | 67 | | **PC21** | | DATA[21] | | CAN0_TX | UART2_TX | SSP1_FS | MIL1_TXAP | ETH_TX_EN |
| J6 | 68 | | **PC22** | | DATA[22] | | CAN1_RX | UART3_RX | SSP1_RX | MIL1_ENA | ETH_TX_CLK |
| L6 | 69 | | **PC23** | | DATA[23] | DACx_SYNC_en[1] | CAN1_TX | UART3_TX | SSP1_TX | MIL1_RXAN | ETH_RX_CLK |
| M6 | 70 | | **PC24** | | DATA[24] | | CAN0_RX | UART0_RX | SSP0_CLK | MIL1_RXAP | ETH_TXD_0 |
| M7 | 79 | | **PC25** | | DATA[25] | | CAN0_TX | UART0_TX | SSP0_FS | MIL1_TXBN | ETH_TXD_1 |
| J7 | 80 | | **PC26** | | DATA[26] | | CAN1_RX | UART1_RX | SSP0_RX | MIL1_TXBP | ETH_TXD_2 |
| J8 | 81 | | **PC27** | | DATA[27] | DAC0_SYNC/ DACx_SYNC_en[1] | CAN1_TX | UART1_TX | SSP0_TX | MIL1_ENB | ETH_TXD_3 |
| L8 | 82 | | **PC28** | | DATA[28] | | CAN0_RX | UART2_RX | SSP1_CLK | MIL1_RXBN | ETH_RXD_0 |
| M8 | 84 | | **PC29** | | DATA[29] | | CAN0_TX | UART2_TX | SSP1_FS | MIL1_RXBP | ETH_RXD_1 |
| K8 | 85 | | **PC30** | | DATA[30] | | CAN1_RX | UART3_RX | SSP1_RX | MIL1_TXAN | ETH_RXD_2 |
| K9 | 86 | | **PC31** | | DATA[31] | | CAN1_TX | UART3_TX | SSP1_TX | MIL1_TXAP | ETH_RXD_3 |

[1] Учитывать errata при разработке.

Таблица 8 – Соответствие выводов порта С и функций с 9-й по 15-ю

| *№ вывода корпуса BGA144* | *№ КП кристалла* | SPECIAL | PORT | I2C CAN-FD | TMR | PWM | PWM | CAP | CAP COMP ADC DAC | QEP | ANALOG |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | - | 0 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | |
| G2 | 29 | | **PC0** | I2C_SCL | TMR2_CH1P | EPWM0_A | EPWMx_nTZ1_EXT | ECAP3_PWM | ADC0_ER0 | EQEP0_A | ADC0_CHP00 |
| G1 | 30 | | **PC1** | I2C_SDA | TMR2_CH1N | EPWM0_B | EPWMx_SYNCI | ECAP2_PWM | ADC0_ER1 | EQEP0_B | ADC0_CHN00 |
| G4 | 31 | | **PC2** | CANFD_TX | TMR2_CH2P | EPWM1_A | EPWM0_A | ECAP1_PWM | ADC1_ER0 | EQEP0_I | ADC0_CHP01 |
| H4 | 32 | | **PC3** | CANFD_RX | TMR2_CH2N | EPWM1_B | EPWM0_B | ECAP0_PWM | ADC1_ER1 | EQEP0_S | ADC0_CHP02 |
| H3 | 40 | | **PC4** | CANFD_TX | TMR2_CH3P | EPWM2_A | EPWM1_A | ECAP0_PWM | ADC2_ER0 | EQEP1_A | ADC0_CHP03 |
| J3 | 41 | | **PC5** | CANFD_RX | TMR2_CH3N | EPWM2_B | EPWM1_B | ECAP1_PWM | ADC2_ER1 | EQEP1_B | ADC0_CHP04 |
| J2 | 42 | | **PC6** | I2C_SCL | TMR2_CH4P | EPWM3_A | EPWM2_A | ECAP2_PWM | ECAPx_SYNCI | EQEP1_I | ADC0_CHP05 |
| J1 | 43 | | **PC7** | I2C_SDA | TMR2_CH4N | EPWM3_B | EPWM2_B | ECAP3_PWM | ECAPx_SYNCO | EQEP1_S | ADC0_CHN05 |
| J4 | 44 | | **PC8** | CANFD_TX | TMR3_CH1P | EPWM4_A | EPWMx_nTZ2_EXT | ECAP3_PWM | COMP0_OUT | EQEP0_A | ADC0_CHP06 |
| K3 | 45 | | **PC9** | CANFD_RX | TMR3_CH1N | EPWM4_B | EPWMx_nTZ3_EXT | ECAP2_PWM | COMP1_OUT | EQEP0_B | ADC0_CHP07 |
| K2 | 46 | | **PC10** | CANFD_TX | TMR3_CH2P | EPWM5_A | EPWMx_nTZ4_EXT | ECAP1_PWM | COMP2_OUT | EQEP0_I | ADC0_CHP08 |
| K1 | 48 | | **PC11** | CANFD_RX | TMR3_CH2N | EPWM5_B | EPWMx_nTZ5_EXT | ECAP0_PWM | COMP3_OUT | EQEP0_S | ADC0_CHN08 |
| M2 | 51 | | **PC12** | I2C_SCL | TMR3_CH3P | EPWM6_A | EPWM0_A | ECAP0_PWM | ECAPx_SYNCI | EQEP1_A | ADC1_CHP00 |
| M1 | 52 | | **PC13** | I2C_SDA | TMR3_CH3N | EPWM6_B | EPWM0_B | ECAP1_PWM | ECAPx_SYNCO | EQEP1_B | ADC1_CHP01 |
| M3 | 54 | | **PC14** | CANFD_TX | TMR3_CH4P | EPWM7_A | EPWM1_A | ECAP2_PWM | DAC0_SYNC/ DACx_SYNC_en[1] | EQEP1_I | ADC1_CHP02 |
| L3 | 55 | | **PC15** | CANFD_RX | TMR3_CH4N | EPWM7_B | EPWM1_B | ECAP3_PWM | | EQEP1_S | ADC1_CHN02 |
| K4 | 56 | | **PC16** | CANFD_TX | TMR0_CH1P | EPWM8_A | EPWM2_A | ECAP3_PWM | ADC0_ER0 | EQEP0_A | ADC1_CHP03 |
| J5 | 63 | | **PC17** | CANFD_RX | TMR0_CH1N | EPWM8_B | EPWM2_B | ECAP2_PWM | ADC0_ER1 | EQEP0_B | ADC1_CHP04 |
| K5 | 64 | | **PC18** | I2C_SCL | TMR0_CH2P | EPWMx_nTZ1_EXT | EPWM3_A | ECAP1_PWM | ADC1_ER0 | EQEP0_I | ADC1_CHP05 |
| L5 | 65 | | **PC19** | I2C_SDA | TMR0_CH2N | EPWMx_nTZ2_EXT | EPWM3_B | ECAP0_PWM | ADC1_ER1 | EQEP0_S | ADC1_CHP06 |
| M5 | 66 | | **PC20** | CANFD_TX | TMR0_CH3P | EPWMx_nTZ3_EXT | EPWM4_A | ECAP0_PWM | ADC2_ER0 | EQEP1_A | ADC1_CHN06 |
| H5 | 67 | | **PC21** | CANFD_RX | TMR0_CH3N | EPWMx_nTZ4_EXT | EPWM4_B | ECAP1_PWM | ADC2_ER1 | EQEP1_B | ADC1_CHP07 |
| J6 | 68 | | **PC22** | CANFD_TX | TMR0_CH4P | EPWMx_nTZ5_EXT | EPWM5_A | ECAP2_PWM | | EQEP1_I | ADC2_CHP00 |
| L6 | 69 | | **PC23** | CANFD_RX | TMR0_CH4N | EPWMx_nTZ6_EXT | EPWM5_B | ECAP3_PWM | | EQEP1_S | ADC2_CHP01 |
| M6 | 70 | | **PC24** | I2C_SCL | TMR0_ETR | EPWMx_SYNCI | EPWM6_A | ECAP3_PWM | COMP0_OUT | EQEP0_A | ADC2_CHN01 |
| M7 | 79 | | **PC25** | I2C_SDA | TMR0_BRK | EPWM0_A | EPWM6_B | ECAP2_PWM | COMP1_OUT | EQEP0_B | ADC2_CHP02 |
| J7 | 80 | | **PC26** | CANFD_TX | TMR1_ETR | EPWM0_B | EPWM7_A | ECAP1_PWM | COMP2_OUT | EQEP0_I | ADC2_CHP03 |
| J8 | 81 | | **PC27** | CANFD_RX | TMR1_BRK | EPWM1_A | EPWM7_B | ECAP0_PWM | COMP3_OUT | EQEP0_S | ADC2_CHP04 |
| L8 | 82 | | **PC28** | CANFD_TX | TMR2_ETR | EPWM1_B | EPWM8_A | ECAP0_PWM | | EQEP1_A | ADC2_CHP05 |
| M8 | 84 | | **PC29** | CANFD_RX | TMR2_BRK | EPWM2_A | EPWM8_B | ECAP1_PWM | | EQEP1_B | ADC2_CHN05 |
| K8 | 85 | CRPT_SWDIO[2] | **PC30** | I2C_SCL | TMR3_ETR | EPWM2_B | EPWMx_nTZ6_EXT | ECAP2_PWM | ADCx_SYNCI | EQEP1_I | ADC2_CHP06 |
| K9 | 86 | CRPT_SWCLK[2] | **PC31** | I2C_SDA | TMR3_BRK | EPWMx_SYNCI | | ECAP3_PWM | ADCx_SYNCO | EQEP1_S | ADC2_CHP07 |

[1] Учитывать errata при разработке.

[2] В случае, если не заблокирован отладочный интерфейс

Таблица 9 – Соответствие выводов порта D и функций с 1-ой по 8-ю

| *№ вывода корпуса BGA144* | *№ КП кристалла* | SPECIAL | PORT | SDIO | A\|D | A\|D | CAN | UART | SSP | MIL | ETH |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | - | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| | 146 | | **PD5** | | - | | CAN0_TX | UART2_TX | SSP1_FS | MIL1_TXBN | ETH_MDC |
| H1 | 35 | | **PD6** | | DATA[45] | | CAN1_RX | UART3_RX | SSP1_RX | MIL1_TXBP | ETH_RX_DV |
| H1 | 36 | | **PD7** | | | | CAN1_TX | UART3_TX | SSP1_TX | MIL1_ENB | ETH_TX_EN |
| GND | 38 | | **PD8** | | | | CAN0_RX | UART0_RX | SSP0_CLK | MIL1_RXBN | ETH_TX_D0 |
| GND | 39 | | **PD9** | | | | CAN0_TX | UART0_TX | SSP0_FS | MIL1_RXBP | ETH_TX_D1 |
| M4 | 58 | | **PD10** | | DATA[46] | | CAN1_RX | UART1_RX | SSP0_RX | MIL1_TXAN | ETH_TX_D2 |
| M4 | 59 | | **PD11** | | | | CAN1_TX | UART1_TX | SSP0_TX | MIL1_TXAP | ETH_TX_D3 |
| GND | 60 | | **PD12** | | | | CAN0_RX | UART2_RX | SSP1_CLK | MIL1_ENA | ETH_RX_D0 |
| GND | 61 | | **PD13** | | | | CAN0_TX | UART2_TX | SSP1_FS | MIL1_RXAN | ETH_RX_D1 |
| L7 | 74 | | **PD14** | | DATA[47] | | CAN1_RX | UART3_RX | SSP1_RX | MIL1_RXAP | ETH_RX_D2 |
| L7 | 75 | | **PD15** | | | | CAN1_TX | UART3_TX | SSP1_TX | MIL1_TXBN | ETH_RX_D3 |
| GND | 76 | | **PD16** | | | | CAN0_RX | UART0_RX | SSP0_CLK | MIL1_TXBP | ETH_RMII_CLK |
| GND | 77 | | **PD17** | | | | CAN0_TX | UART0_TX | SSP0_FS | MIL1_ENB | ETH_RX_ERR |
| | 118 | | **PD18** | | | | CAN1_RX | UART1_RX | SSP0_RX | MIL1_RXBN | |
| | 166 | | **PD19** | | | | CAN1_TX | UART1_TX | SSP0_TX | MIL1_RXBP | |
| | 130 | | **PD20** | | | | CAN0_RX | UART2_RX | SSP1_CLK | MIL1_TXAN | ETH_RX_CLK |
| | 133 | **TEST_O** | | | | | | | | | |
| | 140 | | **PD22** | | | | CAN1_RX | UART3_RX | SSP1_RX | MIL1_ENA | ETH_RX_DV |
| | 143 | | **PD23** | | | | CAN1_TX | UART3_TX | SSP1_TX | MIL1_RXAN | ETH_TX_EN |
| | 181 | | **PD24** | | | | CAN0_RX | UART0_RX | SSP0_CLK | MIL1_RXAP | ETH_RX_D0 |
| | 184 | | **PD25** | | | | CAN0_TX | UART0_TX | SSP0_FS | MIL1_TXBN | ETH_RX_D1 |
| | 187 | | **PD26** | | | | CAN1_RX | UART1_RX | SSP0_RX | MIL1_TXBP | ETH_RX_D2 |
| | 13 | | **PD27** | | | | CAN1_TX | UART1_TX | SSP0_TX | MIL1_ENB | ETH_RX_D3 |
| | 6 | | **PD28** | | | | CAN0_RX | UART2_RX | SSP1_CLK | MIL1_RXBN | ETH_TX_D0 |
| GND | 192 | | **PD29** | | | | CAN0_TX | UART2_TX | SSP1_FS | MIL1_RXBP | ETH_TX_D1 |
| GND | 193 | | **PD30** | | | | CAN1_RX | UART3_RX | SSP1_RX | MIL1_TXAN | ETH_TX_D2 |
| | 169 | | **PD31** | | | | CAN1_TX | UART3_TX | SSP1_TX | MIL1_TXAP | ETH_TX_D3 |

Примечание – Использовать выводы PD8, PD9, PD12, PD13, PD16, PD17, PD29 и PD30 (соединены с GND внутри корпуса) в функциях, отличных от аналоговых, запрещено.

Таблица 10 – Соответствие выводов порта D и функций с 9-ой по 15-ю

| *№ вывода корпуса BGA144* | *№ КП кристалла* | SPECIAL | PORT | I2C CAN-FD | TMR | PWM | PWM | CAP | CAP COMP ADC | QEP | ANALOG |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | - | 0 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | |
| | 146 | | **PD5** | | TMR1_CH3N | EPWM1_B | EPWM2_B | ECAP1_PWM | ADC2_ER1 | EQEP1_B | |
| H1 | 35 | | **PD6** | I2C_SCL | TMR1_CH4P | EPWM2_A | EPWM3_A | ECAP2_PWM | | EQEP1_I | ADC0_REFP |
| H1 | 36 | | **PD7** | I2C_SDA | TMR1_CH4N | EPWM2_B | EPWM3_B | ECAP3_PWM | | EQEP1_S | ADC0_REFP |
| GND | 38 | | **PD8** | | TMR2_CH1P | EPWMx_nTZ2_EXT | EPWM4_A | ECAP3_PWM | COMP0_OUT | EQEP0_A | ADC0_REFN |
| GND | 39 | | **PD9** | | TMR2_CH1N | EPWMx_nTZ3_EXT | EPWM4_B | ECAP2_PWM | COMP1_OUT | EQEP0_B | ADC0_REFN |
| M4 | 58 | | **PD10** | | TMR2_CH2P | EPWMx_nTZ4_EXT | EPWM5_A | ECAP1_PWM | COMP2_OUT | EQEP0_I | ADC1_REFP |
| M4 | 59 | | **PD11** | | TMR2_CH2N | EPWMx_nTZ5_EXT | EPWM5_B | ECAP0_PWM | COMP3_OUT | EQEP0_S | ADC1_REFP |
| GND | 60 | | **PD12** | I2C_SCL | TMR2_CH3P | EPWM0_A | EPWM6_A | ECAP0_PWM | | EQEP1_A | ADC1_REFN |
| GND | 61 | | **PD13** | I2C_SDA | TMR2_CH3N | EPWM0_B | EPWM6_B | ECAP1_PWM | ECAPx_SYNCO | EQEP1_B | ADC1_REFN |
| L7 | 74 | | **PD14** | | TMR2_CH4P | EPWM1_A | EPWM7_A | ECAP2_PWM | | EQEP1_I | ADC2_REFP |
| L7 | 75 | | **PD15** | | TMR2_CH4N | EPWM1_B | EPWM7_B | ECAP3_PWM | | EQEP1_S | ADC2_REFP |
| GND | 76 | | **PD16** | | TMR0_ETR | EPWM2_A | EPWM8_A | ECAP3_PWM | ADC0_ER0 | EQEP0_A | ADC2_REFN |
| GND | 77 | | **PD17** | | TMR0_BRK | EPWM2_B | EPWM8_B | ECAP2_PWM | ADC0_ER1 | EQEP0_B | ADC2_REFN |
| | 118 | | **PD18** | I2C_SCL | TMR1_ETR | EPWM3_A | EPWMx_nTZ1_EXT | ECAP1_PWM | ADC1_ER0 | EQEP0_I | |
| | 166 | | **PD19** | I2C_SDA | TMR1_BRK | EPWM3_B | EPWMx_nTZ2_EXT | ECAP0_PWM | ADC1_ER1 | EQEP0_S | |
| | 130 | | **PD20** | | TMR2_ETR | EPWM4_A | EPWMx_nTZ3_EXT | ECAP0_PWM | ADC2_ER0 | EQEP1_A | |
| | 133 | **TEST_O** | | | | | | | | | |
| | 140 | | **PD22** | | TMR3_ETR | EPWM5_A | EPWMx_nTZ5_EXT | ECAP2_PWM | | EQEP1_I | |
| | 143 | | **PD23** | | TMR3_BRK | EPWM5_B | EPWMx_nTZ6_EXT | ECAP3_PWM | | EQEP1_S | |
| | 181 | | **PD24** | I2C_SCL | TMR3_CH1P | EPWM6_A | EPWMx_SYNCI | ECAP3_PWM | COMP0_OUT | EQEP0_A | ANA_BG_ EXTREF4 |
| | 184 | | **PD25** | I2C_SDA | TMR3_CH1N | EPWM6_B | EPWM0_A | ECAP2_PWM | COMP1_OUT | EQEP0_B | ANA_BG_ EXTREF3 |
| | 187 | | **PD26** | | TMR3_CH2P | EPWM7_A | EPWM0_B | ECAP1_PWM | COMP2_OUT | EQEP0_I | ANA_BG_ EXTREF2 |
| | 13 | | **PD27** | | TMR3_CH2N | EPWM7_B | EPWM1_A | ECAP0_PWM | COMP3_OUT | EQEP0_S | ANA_BG_ EXTREF1 |
| | 6 | | **PD28** | | TMR3_CH3P | EPWM8_A | EPWM1_B | ECAP0_PWM | | EQEP1_A | ANA_BG_ EXTREF0 |
| GND | 192 | | **PD29** | | TMR3_CH3N | EPWM8_B | EPWM2_A | ECAP1_PWM | ECAPx_SYNCO | EQEP1_B | DAC3_VRFN0 |
| GND | 193 | | **PD30** | I2C_SCL | TMR3_CH4P | EPWMx_nTZ6_EXT | EPWM2_A | ECAP2_PWM | | EQEP1_I | DAC3_VRFN1 |
| | 169 | | **PD31** | I2C_SDA | TMR3_CH4N | EPWMx_SYNCI | | ECAP3_PWM | | EQEP1_S | |

Примечание – Использовать выводы PD8, PD9, PD12, PD13, PD16, PD17, PD29 и PD30 (соединены с GND внутри корпуса) в функциях, отличных от аналоговых, запрещено.

Таблица 11 – Соответствие выводов криптографического сопроцессора и функций с 0-ой по 3-ю

| *№ вывода корпуса BGA144* | *№ КП кристалла* | SPECIAL | INPUT | OUTPUT | ALTER | ANALOG |
|---|---|---|---|---|---|---|
| | | - | 00 | 01 | 10 | 11 |
| J11 | 106 | **CRPT_KEY_RSTn** | | | | |
| A1 | 2 | | CRPT_GPIO5 | CRPT_GPIO5 | **CRPT_UARTTX** | |
| B7 | 167 | | CRPT_GPIO6 | CRPT_GPIO6 | **CRPT_UARTRX** | |
| A2 | 199 | | CRPT_GPIO7 | CRPT_GPIO7 | **CRPT_UARTCLK** | |
| K8 | 85 | PC30 | CRPT_GPIO10 | CRPT_GPIO10 | **CRPT_SWDIO** | |
| K9 | 86 | PC31 | CRPT_GPIO11 | CRPT_GPIO11 | **CRPT_SWCLK** | |

# 4 Указания по применению и эксплуатации

Материал шариковых выводов Sn63/Pb37.

Требования к выполнению технологических операций пайки см. ГОСТ Р 56427-2015.

При хранении микросхем более 12 месяцев выводы обезжиривают путем погружения в спирт (по ГОСТ Р 55878 или ГОСТ Р 51999) и высушивают в течение от 3 до 5 мин при комнатной температуре.

Очистку микросхем проводят спиртонефрасовой смесью (1:1) в режиме виброочистки по ГОСТ Р 56427-2015.

При ремонте аппаратуры и измерении параметров 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 таблице выводов.

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

– подача (включение микросхем) – общий, питание, входные сигналы или одновременно;

– снятие (выключение микросхем) – в обратном порядке или одновременно.

# 5 Управление питанием микроконтроллера

## 5.1 Структура питания микросхемы

В микросхеме реализованы следующие домены питания:

- питание схем ввода-вывода $V_{CC_IO}$ (3,3 В);
- питание встроенного регулятора HLDO, DCDC преобразователя $V_{CC_DCDC}$ (3,3 В);
- питание встроенного регулятора HLDO, DCDC преобразователя $V_{CC}$ (3,3 В);
- питание встроенного регулятора формирования напряжения питания батарейного домена $V_{CC_BKP}$ (3,3 В);
- питание формирователя опорного напряжения АЦП, ЦАП и компараторов $V_{CC_BG}$ (3,3 В);
- питание АЦП1 $V_{CC_ADC0}$ (3,3 В);
- питание АЦП2 $V_{CC_ADC1}$ (3,3 В);
- питание АЦП3 $V_{CC_ADC2}$ (3,3 В);
- питание схем ЦАП и компараторов $V_{CC_DAC}$ (3,3 В);
- питание схем генераторов и умножителей частоты $V_{CC_ANA}$ (3,3 В);
- питание приемо-передатчиков USB $V_{CC_USB}$ (3,3 В);
- выход напряжения встроенного регулятора HLDO VHLDO (1,6 В);
- выход напряжения встроенного регулятора LLDO VLLDO (1,2 В);
- питание цифровой части батарейного домена $VLDO_{BKP}$ (1,2 В).

## 5.2 Организация питания цифровой части микросхемы

Для формирования напряжения питания 1,2 В в микросхеме реализованы два блока линейных регуляторов напряжения питания HLDO и LLDO и блок импульсного преобразователя DC-DC.

В начале работы микросхемы преобразователь DC-DC выключен, питание формируется цепочкой HLDO (преобразование 3,3 В в 1,6 В) и LLDO (преобразование 1,6 В в 1,2 В).

Далее может быть осуществлен перевод микроконтроллера в режим питания от DC-DC преобразователя. Для этого в регистре REG_61_PWR блока батарейного домена следует установить бит разрешения работы импульсного преобразователя (dcdc_en), дождаться выхода преобразователя в рабочий режим (бит dcdc_ready), после чего выключить линейный регулятор HLDO (бит hldo_dis) регистра REG_61_PWR.

Импульсный преобразователь осуществляет понижение питания до 1,6 В. Линейный регулятор LLDO должен оставаться активным как в режиме HLDO, так и DC-DC.

Для работы микросхемы в режиме DC-DC требуется установка на плате внешних элементов (см. типовую схему включения).

В режиме работы DC-DC динамическое потребление микросхемы снижается на величину до 50 % относительно потребления микросхемы в режиме работы HLDO при одинаковой нагрузке. Пример диаграммы потребления в различных режимах работы представлен на рисунке 3.

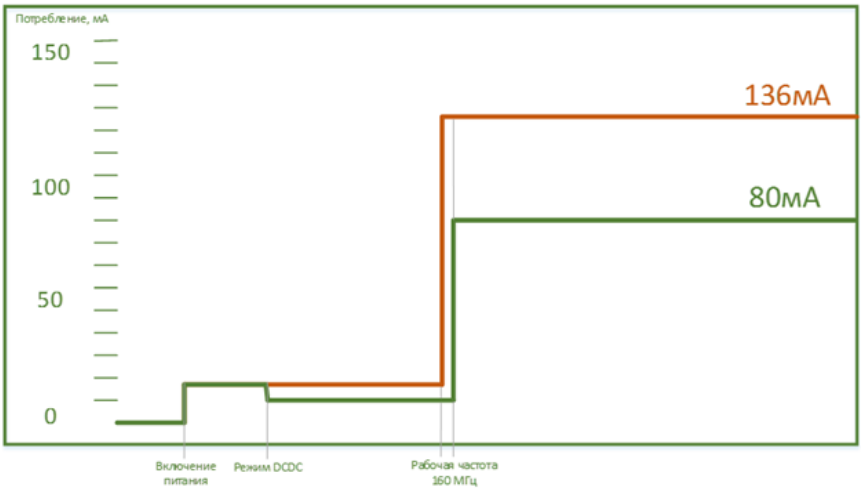



Рисунок 3 – Пример диаграммы потребления тока в различных режимах работы

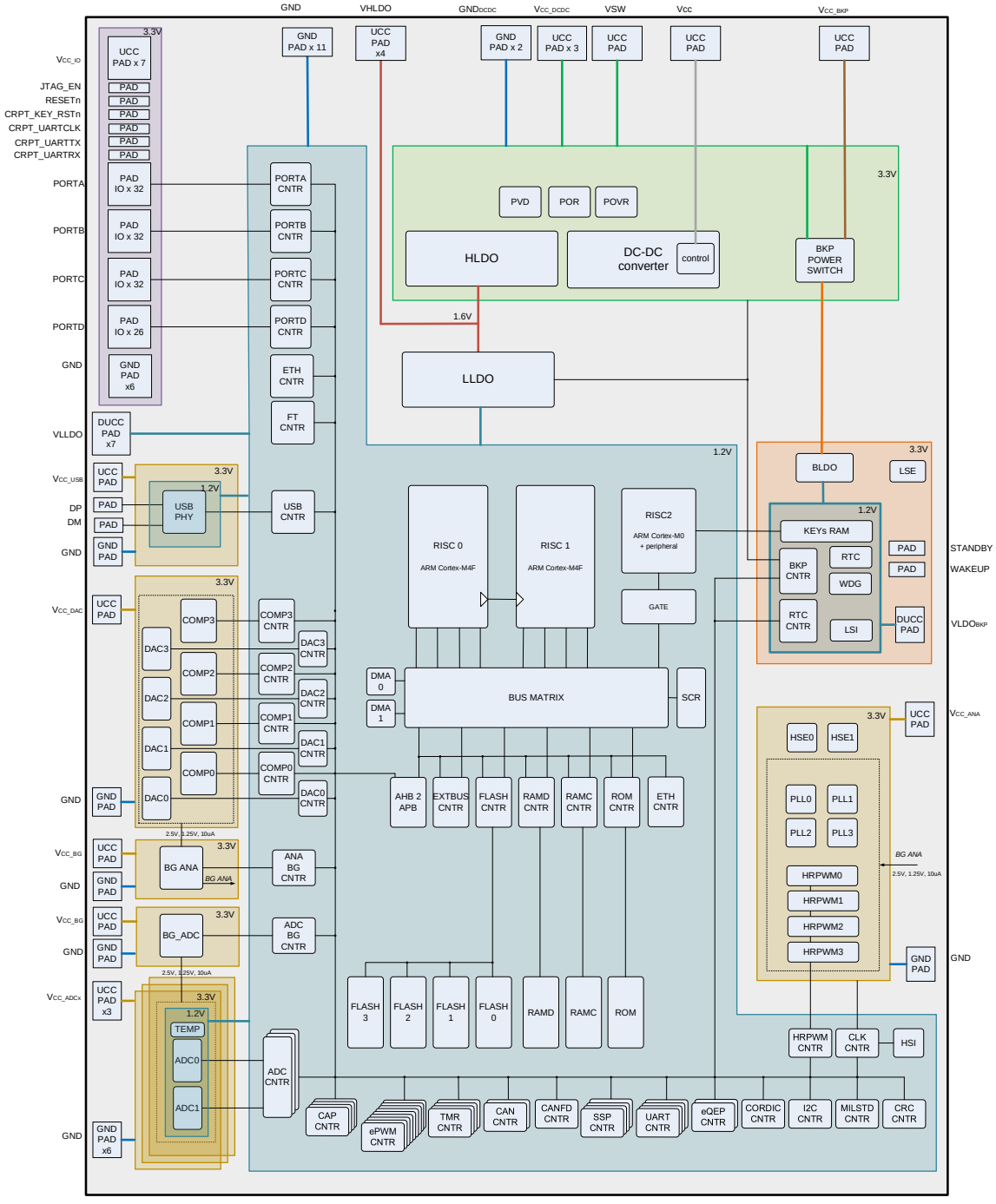



Рисунок 4 – Структурная схема питания микроконтроллера

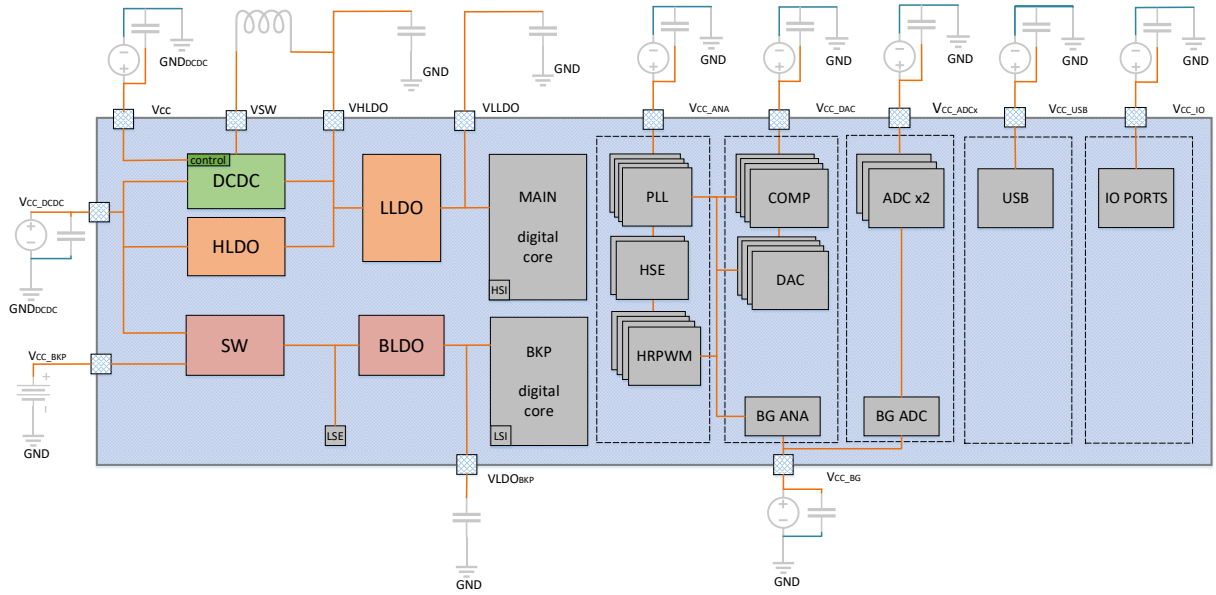



Рисунок 5 – Домены питания микроконтроллера

## 5.3 Организация питания батарейного домена

Схема батарейного домена может работать как от основного, так и от батарейного питания. Для переподключения питания батарейного домена ($V_{CC_BPS}$) между основным и батарейным питанием в системе реализован блок переключения питаний BKP POWER SWITCH. Сигнал готовности основного питания PORRSTn является ключом для переключения питания батарейного домена.

Таким образом, при отсутствии основного питания $V_{CC_DCDC}$ батарейный домен запитан от питания $V_{CC_BKP}$. При включении основного питания и превышении им уровня $U_{PORON}$ через время $T_{PORRSTn}$ питание батарейного домена переключится на основное питание. При снижении основного питания ниже уровня $U_{PORON}$ питание батарейного домена автоматически переключится на $V_{CC_BKP}$.

При выборе источника частоты в блоке переключения частот необходимо, чтобы предыдущий источник частоты был включён.

Примечание – Отсутствие напряжения питания батарейного домена после сброса по питанию или в момент первого запуска микроконтроллера может привести к неправильной работе.

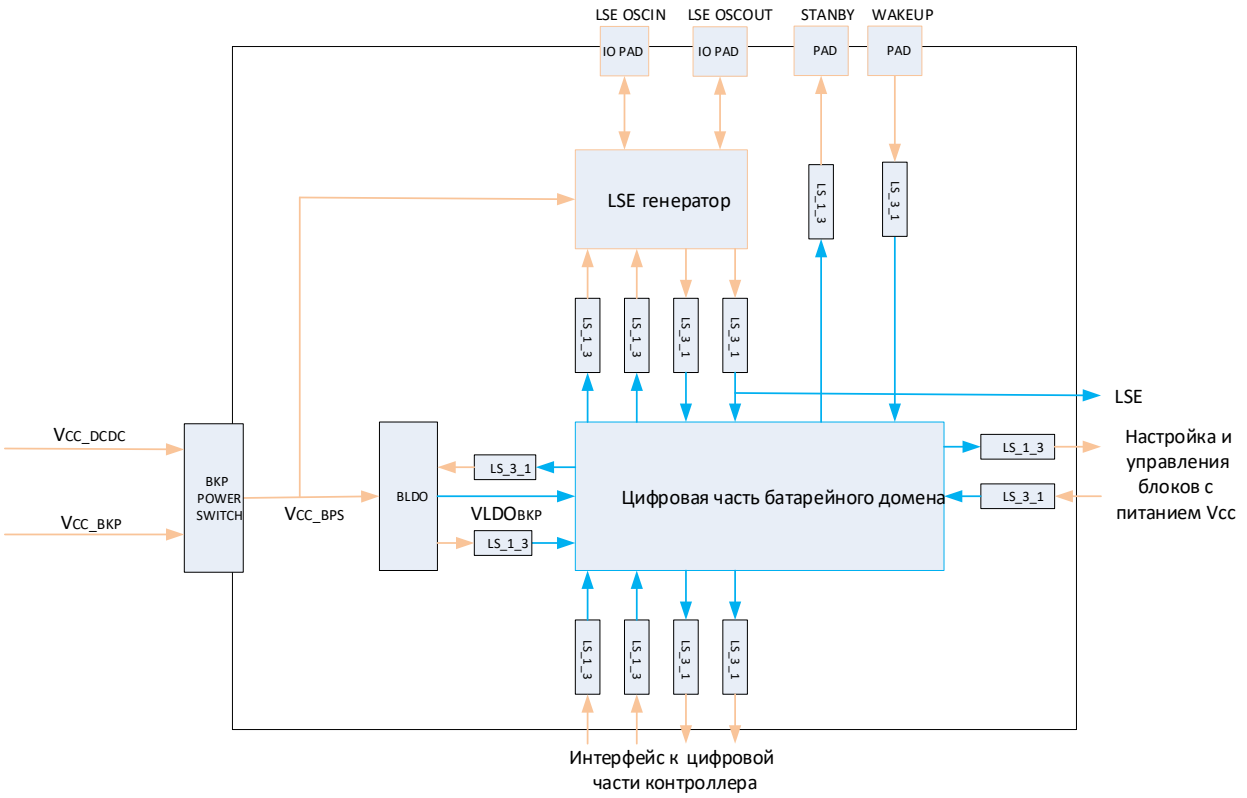



Рисунок 6 – Структура схема организации питания батарейного домена микроконтроллера

## 5.4 Схема сброса по питанию (POR)

Схема POR предназначена для выработки аппаратного сброса при включении или просадке ниже критического уровня основного питания. При включении питания пока уровень $V_{CC_DCDC}$ не превысил значение $U_{PORON}$, сигнал сброса PORRSTn будет нулевым (сброс). После превышения уровня $U_{PORON}$ сигнал PORRSTn еще в течение $T_{PORSTn}$ удерживается в нулевом состоянии и затем переключится в единицу. Задержка $T_{PORSTn}$ формируется на счетчике, работающим с тактовой частотой от встроенного генератора, который работает в домене питания $V_{CC}$. Если по каким-либо причинам произошел отказ встроенного генератора, схема сформирует сигнал PORRSTn с задержкой много меньшей чем $T_{PORSTn}$. При снижении уровня $V_{CC_DCDC}$ ниже уровня $U_{POROFF}$, сигнал PORRSTn сбрасывается в ноль с минимальной задержкой.

Факт срабатывания схемы POR фиксируется в батарейном домене и может быть проанализирован при последующем запуске системы.

При снижении питания ниже уровня $U_{POROFF}$ происходит активация сброса микросхемы.

Блок POR активен при включении питания микросхемы. Выключение блока не рекомендуется и может привести к непредсказуемым последствиям.

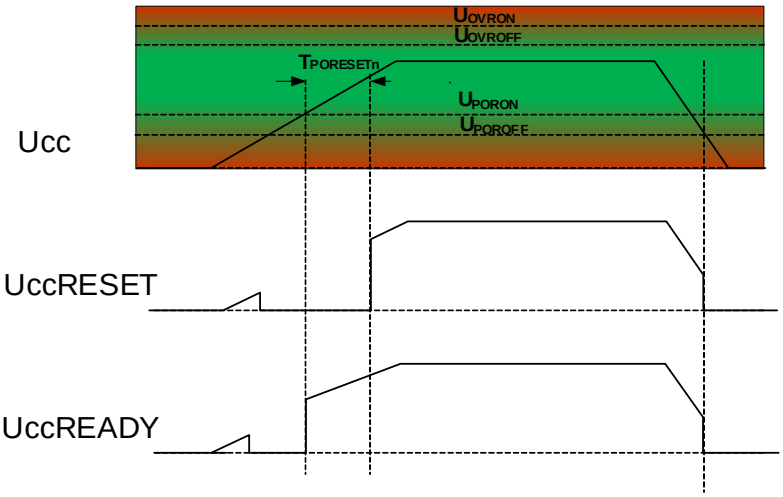



Рисунок 7 – Схема сброса по включению питания

### 5.5 Схема защиты от перенапряжения POVR

В случае, когда напряжение питания $V_{CC_DCDC}$ превышает уровень $U_{OVRON}$, вырабатывается сигнал сброса OVRSTn. Таким образом, когда значение питающего напряжения превышает предельно-допустимые границы, микроконтроллер будет аппаратно сброшен.

Факт срабатывания схемы защиты фиксируется в регистре батарейного домена и может быть проанализирован при последующем запуске системы. Отключать схему защиты в рабочем режиме не рекомендуется.

Сигнал OVRSTn становится неактивным при снижении питания ниже уровня $U_{OVROFF}$.

Блок POVR в момент включения питания активен. Выключение блока осуществляется программно битом ovr3p3_dis регистра блока REG_61_PWR батарейного домена.

### 5.6 Блок монитора основного и батарейного питаний Ucc и BUcc

Для определения абсолютных значений уровней напряжения питания используется блок PVD, обеспечивающий определение уровня напряжения с точностью ± 100 мВ.

Результат определения уровней напряжения отображается в блоке батарейного домена. При этом можно производить мониторинг флагов для программной обработки данного события.

Алгоритм работы с блоком:

1 Задать уровень монитора требуемого питания (шаг 100 мВ) при помощи полей PWRM_Vcc_LVL, PWRM_B_Vcc_LVL или PWRM_IO_Vcc_LVL регистра REG_62_PWR.

2 Разрешить работу требуемого монитора при помощи полей PWRM_Vcc_EN, PWRM_B_Vcc_EN или PWRM_IO_Vcc_EN регистра REG_62_PWR.

3 Вести обработку полученных результатов отработки блока, которая заключается в выставлении флагов события превышения напряжения, задаваемого первым пунктом. Флаги в регистре REG_62_PWR объявлены как PWRM_Vcc_EVENT, PWRM_B_Vcc_EVENT и PWRM_IO_Vcc_EVENT.

При включении монитора питания по напряжению $V_{CC_BKP}$ происходит постоянное потребление от источника $V_{CC_BKP}$, независимо от наличия или отсутствия $V_{CC_DCDC}$. Таким образом, в случае использования батарейки для $V_{CC_BKP}$, анализ $V_{CC_BKP}$ желательно не делать непрерывным, а осуществлять периодически с коротко-временным включением/отключением анализа уровня напряжения $V_{CC_BKP}$.

# 6 Сигналы сброса микроконтроллера

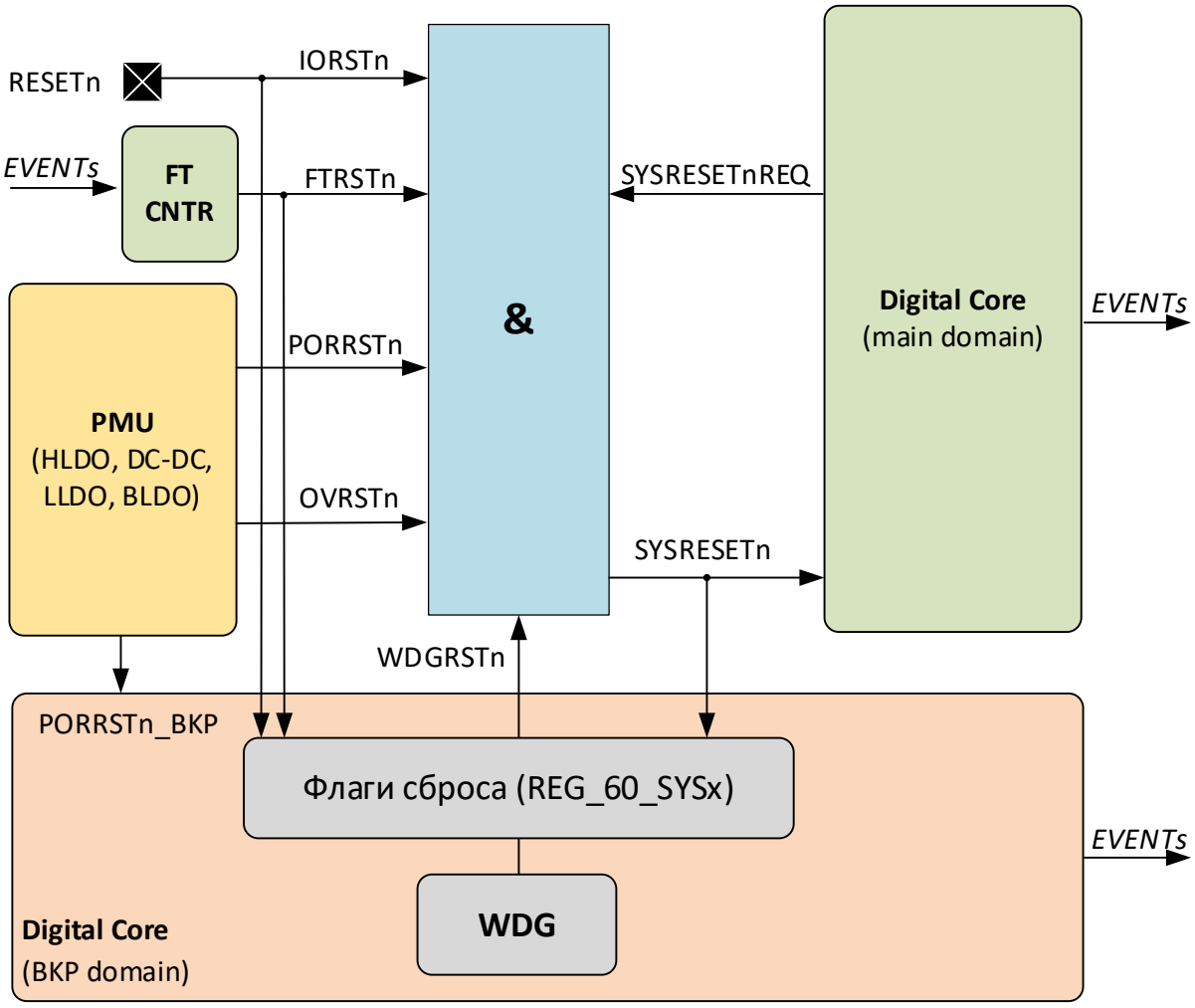



Рисунок 8 – Структура схема организации сброса микроконтроллера

Сигнал внешнего сброса **RESETn** осуществляется подачей низкого уровня на вывод RESETn. Микросхема может быть сброшена внешним сигналом с вывода RESETn.

Сигнал сброса по питанию **PORRSTn** активируется при снижении уровня питания $V_{CC_DCDC}$ ниже $U_{POROFF}$. При включении питания $V_{CC_DCDC}$ (превышения уровня $U_{PORON}$) сигнал остается активным в течение времени $T_{PORSTn}$.

Сигнал сброса по перенапряжению **OVRSTn** – возникает при превышении питанием $V_{CC_DCDC}$ уровня $U_{OVRON}$ обеспечивающий сброс контроллера для предотвращения

Сигнал программного запроса сброса **SYS_RESET_REQ** – вырабатывается программно ядром микроконтроллера. Запрос сброса будет выполнен для обоих процессорных ядер, даже в том случае, если он сформирован одним из ядер в режиме DUALCORE.

Сигналы аварийного сброса **FT_RESET** – активируется блоком контроллера обработки событий отказов, сбоев и ошибок при возникновении одного из поддерживаемых нештатных или ошибочных состояний. После сброса микросхемы все сбросы по аварийным событиям отключены. Для настройки сбросов по аварийным сигналам необходимо задать соответствующие настройки блока контроллера ошибок.

Сигнал сброса сторожевого таймера **IWDG_RESET** – настраивается заданием регистров контроллера сторожевого таймера. Блок сторожевого таймера расположен в батарейном домене.

Факты срабатывания любого из сбросов цифрового ядра микроконтроллера фиксируются в регистрах батарейного домена и могут быть проанализирован при последующем запуске системы.

Сигнал сброса по питанию батарейного домена **PORRSTn_BKP** – активируется при отключении всех источников питания, включая батарейный. Устанавливают все регистры батарейного домена в начальное состояние.

# 7 Синхросигналы микроконтроллера

## 7.1 Источники синхросигналов микроконтроллера

### 7.1.1 *Встроенный высокоскоростной генератор HSI*

RC-генератор HSI с частотой в диапазоне от 5 до 11 МГц является основным встроенным синхросигналом микроконтроллера, который запускается при включении питания. На данной частоте запускается процессорное ядро, системные шины и синхронные периферийные блоки. Физический отказ схемы HSI является критическим отказом. При неработающем HSI-генераторе запуск микроконтроллера не возможен. Генератор работает в домене питания VLLDO.

Генератор HSI является базовым для блока монитора частоты. С помощью блока монитора тактовых частот может быть определен факт отказа одного из генераторов, выполнен аварийный переход на HSI-генератор и программное восстановление работы системы на оставшемся работающем генераторе.

### 7.1.2 *Внешние высокоскоростные генераторы HSEn*

Для формирования внешней стабильной высокой частоты применяются генераторы HSE0 и HSE1. Наличие нескольких генераторов обеспечивает возможность использования независимых источников частоты для работы периферийных блоков и ядра микроконтроллера.

Генераторы HSE могут работать в одном из следующих режимов:

- режим работы с внешним резонатором (HSEn_BYP = «0»), допустимая частота $f_{C_HSE}$ составляет от 2 до 30 МГц;
- режим работы с внешним генератором (HSEn_BYP = «1»), допустимая частота $f_{C_HSE}$ составляет от 1 до 160 МГц.

Выбор режима работы (HSEn_BYP), а также включение генераторов (HSEn_ON), осуществляется в регистрах REG_63_CLKx. В режиме работы с внешним генератором входная тактовая частота с входа HSEn_OSCIN проходит напрямую на выход генератора HSE, при этом выход HSEn_OSCOUT находится в третьем состоянии.

Генераторы HSE имеют встроенные фильтры помех, исключающие прохождение коротких импульсов или «просечек». Данный фильтр предназначен для исключения кратковременных единичных высокоскоростных импульсов, которые могут быть не обнаружены схемой монитора частоты, которая имеет большее время реакции. Разрешение работы фильтра помех осуществляется с помощью бита HSEn_FILTER_EN в регистрах REG_63_CLKx.

Каждый генератор HSE имеет флаг готовности (HSEn_RDY в регистре HSEn_STAT), который вырабатывается при включении генератора и выходе его в нормальный режим работы. При переходе на данную частоту необходимо дождаться сигнала готовности, чтобы предотвратить вероятность перехода на частоту в случае неработающего генератора.

Генераторы HSE работают в домене питания $V_{CC_ANA}$.

### 7.1.3 *Встроенный низкоскоростной генератор LSI*

Встроенный внутренний RC-генератор LSI имеет рабочую от 20 до 60 кГц. Генератор расположен в батарейном домене и может быть использован в момент отключения основного питания микроконтроллера. Генератор работает в домене питания $VLDO_{BKP}$, что позволяет использовать его в момент отключения основного питания.

### 7.1.4 *Внешний низкоскоростной часовой генератор LSE*

Для формирования внешней стабильной низкой частоты применяется генератор LSE. Данный генератор может работать в одном из следующих режимов:

- режим работы с внешним резонатором (LSE_BYP = «0»), допустимая частота $f_{C_LSE}$ составляет от 32 до 33 кГц;
- режим работы с внешним генератором (LSE_BYP = «1»), допустимая частота $f_{C_LSE}$ составляет не более 1000 кГц.

Выбор режима работы (LSE_BYP), а также включение генератора (LSE_ON), осуществляется в регистрах REG_63_CLKx. В режиме работы с внешним генератором входная тактовая частота с входа LSE_OSCIN проходит напрямую на выход генератора LSE, при этом выход LSE_OSCOUT находится в третьем состоянии.

В микроконтроллере нет второго (дублирующего) генератора LSE. Отказ генератора LSE может быть обнаружен с помощью блока монитора тактовых частот, при этом дополнительно может быть выполнен аварийный переход на генератор HSI. Далее работа системы может быть восстановлена путём формирования низкой частоты с помощью оставшихся работающих генераторов.

Генератор LSE имеет встроенный фильтр помех, исключающий прохождение коротких импульсов или «просечек». Разрешение работы фильтра помех осуществляется с помощью бита LSE_FILTER_EN в регистрах REG_63_CLKx. Также генератор LSE имеет флаг готовности (LSE_RDY в регистрах REG_63_CLKx), который вырабатывается при включении генератора и выходе его в нормальный режим работы.

Генератор LSE работает в домене питания $V_{CC_BKP}$ и может использоваться для тактирования часов реального времени.

Примечание – Рекомендуется использовать часовые кварцевые резонаторы с номинальной ёмкостью нагрузки $C_L$ = 12 пФ, для резонаторов с малой номинальной емкостью нагрузки может потребоваться включение дополнительного резистора $R_S$ (номинал подбирается экспериментально из диапазона 500 кОм – 1 МОм) по выводу LSE OSCOUT, как показано на рисунке 9:

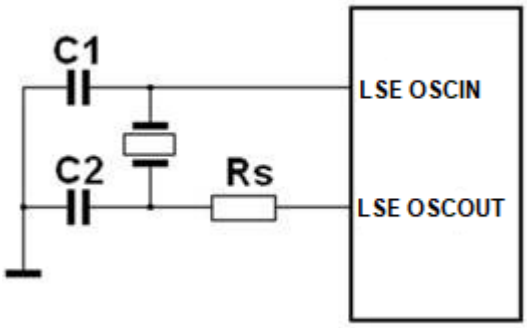



Рисунок 9 – Схема включения LSE с внешними элементами

### 7.1.5 *Блоки умножения тактовой частоты PLLn*

Для формирования высоких тактовых частот используются схемы умножения тактовых частот PLL0, PLL1, PLL2, PLL3. В качестве источников опорной частоты для PLL выступают генераторы HSI и HSE. С помощью PLL формируется частота, которая рассчитывается по формуле (1)

$$F_{O_PLL} = \frac{F_{C_PLL} \cdot (N+1)}{(Q+1) \cdot 2^{DV}}, \quad (1)$$

где N, Q, DV задаются соответствующими значениями в регистрах управления блока.

Допустимые соотношения выходной частоты и делителя DV представлены в таблице 12.

Таблица 12 – Соотношения выходной частоты PLL и делителя DV.

| **Выходная частота $F_{O_PLL}$, МГц** | **Делитель DV** |
|---|---|
| 150 – 160 | 0 |
| 75 – 150 | 1 |
| 40 – 75 | 2 |

Примечание – Допускается выбрать любой делитель DV при совпадении выходной частоты с границей интервала.

При этом для выбранной входной частоты корректная работа блока возможна, только если внутренние частоты (частота фазового детектора $F_{PFD_PLL}$ и внутренняя частота $F_{INT_PLL}$) находятся в диапазонах, указанных в таблице «Предельно-допустимые и предельные параметры». Эти частоты определяются по формулам (2) и (3)

$$F_{PFD_PLL} = \frac{F_{C_PLL}}{Q+1}, \quad (2)$$

$$F_{INT_PLL} = \frac{F_{C_PLL} \cdot (N+1)}{Q+1}. \quad (3)$$

На выходе PLL имеются встроенные фильтры помех, исключающие прохождение коротких импульсов или «просечек». PLL имеет флаг готовности, вырабатывается при включении схемы умножителя и выходе её в нормальный режим работы либо при перезапуске PLL с новыми коэффициентами умножения частоты. С помощью блока монитора частоты можно определить значение частоты по отношению к другим источникам тактирования, обнаружить отказ PLL (снижение или отсутствие частоты, и превышение выходной частоты), выполнить аварийный переход на HSI-генератор и программно восстановить работу системы на других PLL или генераторах.

## 7.2 Домены синхронизации микроконтроллера

Домен синхронизации **MAXCLK** – является основным доменом синхронизации, на основе которого формируются частота процессорного ядра **CPUCLK**, системных шин AHB **HCLK**, и APB **PCLK**. Настройки частоты задаются в регистре MAX_CLK блока управления частотами микроконтроллера CLK_CNTR, в котором можно мультиплицировать любой из источников частоты микроконтроллера, в т.ч. выходы блоков умножения частоты. При включении питания в качестве частоты MAXCLK используется выходная частота генератора HSI.

Частота **CPUCLK** является опорной частотой для микроконтроллерных ядер. Частота формируется на основе MAXCLK делением на выбранное в регистре CPU_CLK значение.

Частота **HCLK** является частотой системных шин и основных системных контроллеров микросхемы. Частота не отключается, не управляется и всегда равна частоте **CPUCLK**.

Частоты **PCLK[63:0]** являются синхросигналами шин APB для периферийных блоков контроллера. Отключение каждого из синхросигналов PCLK возможно для экономии энергии в регистрах PER_CLK1,0. Если синхросигнал разрешен, он всегда синфазен и равен частоте **CPUCLK**.

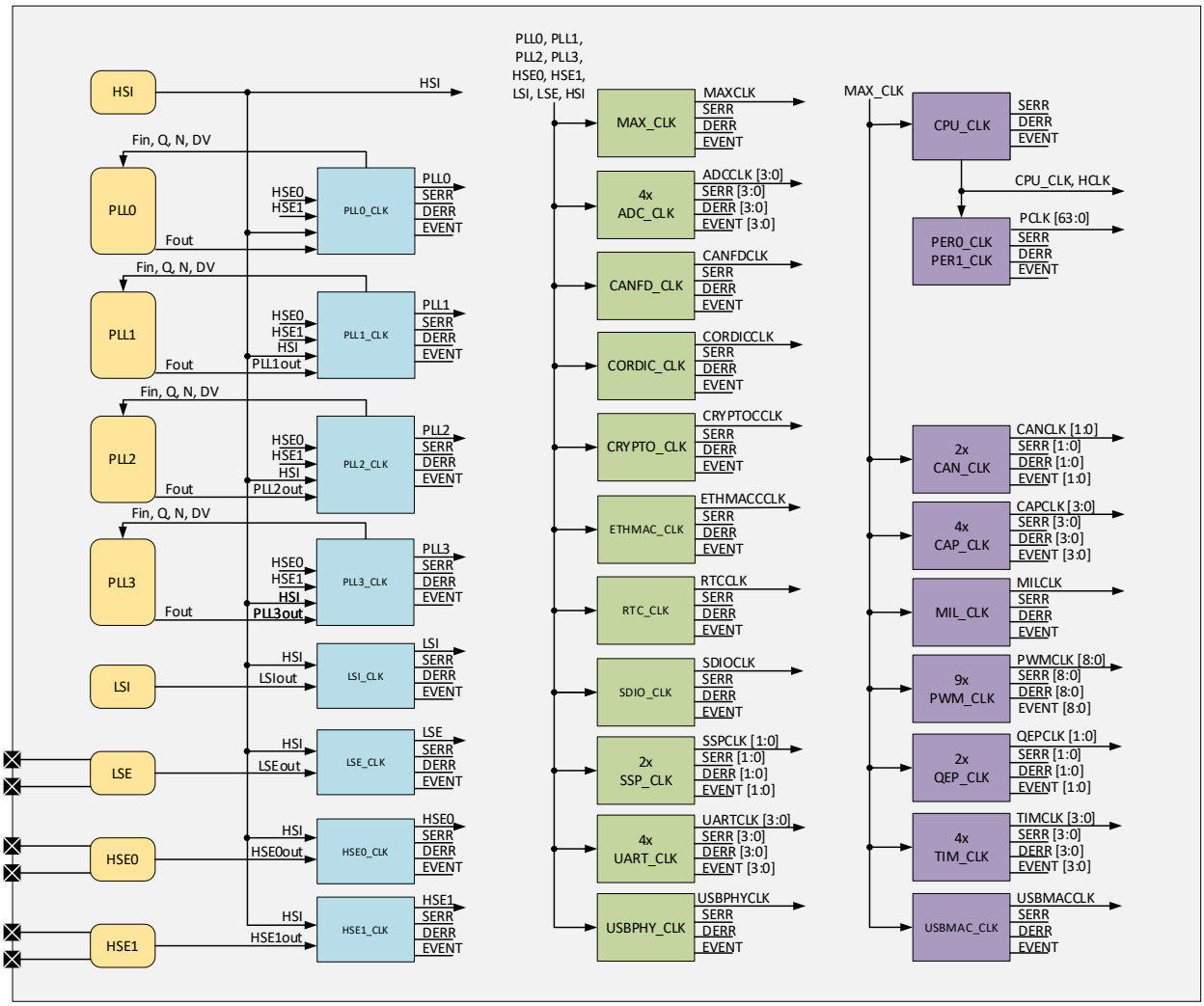



Рисунок 10 – Структурная схема организации синхронизации микроконтроллера

По типу тактирования периферийные блоки микроконтроллера могут быть поделены на группы:

- тактируемые системной частотой (блоки CRC_CNTR, CAP_CNTR, DAC_CNTR, COMP_CNTR);
- тактируемые синхронной с системной частотой (домены синхронизации PWMCLK[8:0], EMACCLK (в ревизии 1), CAPCLK[3:0], CANCLK[1:0], MILCLK, TIMCLK[3:0], QEPCLK[1:0], USBMACCLK). Управление доменами осуществляется в соответствующих регистрах блока CLK_CNTR, где можно включить соответствующую частоту и задать коэффициент деления, при этом частота останется синхронной относительно системной.
- имеющие асинхронную тактовую частоту (домены синхронизации ADCCLK[2:0], CANFDCLK, CRDCCLK, CRPTCLK, EMACCLK, RTCCLK, SDIOCLK, SSPCLK[1:0], UARTCLK[3:0], USBCLK). Управление доменами осуществляется в соответствующих регистрах блока CLK_CNTR, где частота может быть включена, может быть выбран один из источников синхросигнала микроконтроллера, в т.ч. умножитель частоты, а также коэффициент деления выбранной частоты. Полученные частоты являются асинхронными относительно системной частоты. Передача данных между ядром и блоками занимает дополнительное время на пересинхронизацию.

## 7.3 Управление тактовыми частотами (CLKCNTR)

Блок управление тактовыми сигналами является важным элементом обеспечения работоспособности микроконтроллера. При отсутствии тактовых сигналов микроконтроллер не может выполнять возложенные на него функции. При выполнении загрузочной программы в стадии boot (за исключением режимов запуска с использованием автоматического запуска программ самотестирования) и при запуске пользовательской программы в стадии operation, микроконтроллер тактируется встроенным HSI-генератором. HSI-генератор считается надежным и стабильным – отказ HSI-генератора приводит к полному отказу микроконтроллера. Генератор HSI используется для определения значения частоты других источников тактирования.

Блок CLK_CNTR формирует тактовые частоты для процессорных ядер и периферии. Система тактовой синхронизации построена по принципу полной синхронности основных элементов, связанных с процессорными ядрами (блоки памяти ОЗУ, DMA контроллер, мосты, и интерфейсные по отношению к процессору части всех периферийных блоков). То есть эти части микросхемы всегда работают на одной тактовой частоте. Ряд периферийных блоков может быть синхронными процессорному ядру, но при этом блоки могут работать как на большей, так и на меньшей тактовой частоте. В этом случае не требуется дополнительной логики по пересинхронизации. Ряд периферийных блоков может работать на несинхронных частотах. Для обеспечения корректной работы всех синхронных модулей формируется частота MAXCLK. Далее синхронные блоки могут работать либо на частоте MAXCLK, либо на прореженной частоте MAXCLK. При этом процессорные ядра и связанные с ним блоки также могут работать на прореженной частоте (все на одной и той же).

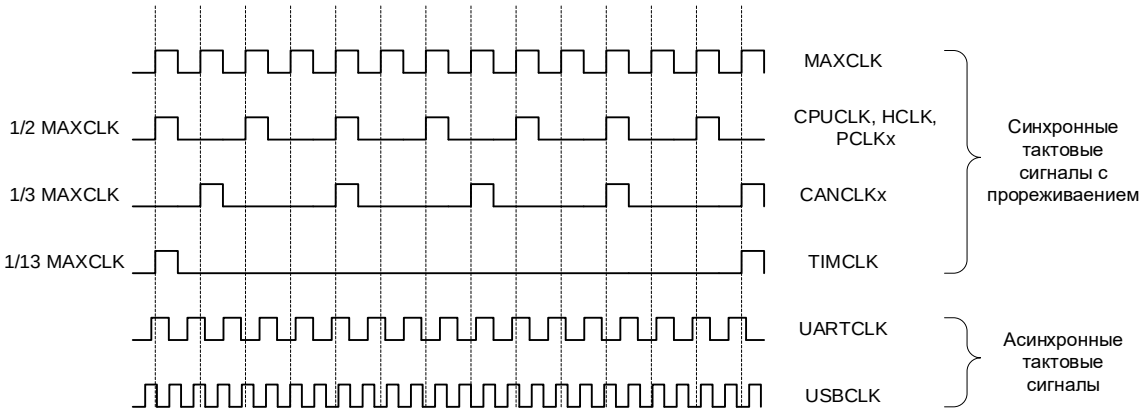



Рисунок 11 – Пример задания тактовых частот в микроконтроллере

При этом в ряде периферийных блоков вводится ограничения на соотношения рабочих частот этих блоков и частоты PCLK интерфейса с процессорным ядром.

## 7.4 Блок контроля частоты внешних генераторов и PLL

Блок монитора и контроля частоты внешних генераторов и PLL Clock Checker входят в состав контроллера тактовых частот CLK_CNTR и предназначены для определения значения тактовых частот по отношению друг к другу и определения четырех событий сбоев в работе генераторов и блоков PLL в зависимости от настроенных критериев:

- незначительное увеличение тактовой частоты;
- значительное увеличение тактовой частоты;
- незначительное снижение тактовой частоты;
- значительно снижение (исчезновение) тактовой частоты.

Блок Clock Checker, в зависимости от настроек, может по-разному реагировать на различные события сбоев тактовых частот:

- вырабатывать сигналы предупреждения (флаги событий);

- аппаратно переключать схему тактирования на другой источник тактирования (HSI) по каждому из четырех событий сбоев;
- формировать прерывания (CLK_IF от блока Clock Checker и FT_IF2 от блока FT_CNTR).

Блоки Clock Checker установлены на входах тактовых частот от внешних источников HSE0, HSE1, LSE, LSI, на всех PLL, а также на выходе схемы формирования тактовых сигналов процессорных ядер CPU_CLK.

В блоке Clock Checker предусмотрена программно настраиваемая возможность аппаратного переключения выхода источника частоты (в том числе CPU_CLK) на тактовую частоту генератора HSI при возникновении событий сбоев.

Определение соотношения частот между собой происходит путем сравнения значений счетчиков, считающих на разных частотах – проверяемой частоте и частоте генератора HSI. Сравнивая это соотношение с различными пределами, определяется событие сбоя. Для обнаружения факта сбоя, в зависимости от реального соотношения частот и заданных пределов их допустимого соотношения, может потребоваться некоторое время, таким образом, при возникновении сбоя или отказа блоку монитора потребуется некоторое время для реакции на это событие. Блок монитора рассчитан на работу в том числе на частоте, значительно превышающей максимальную тактовую частоту микроконтроллера, так как при возникновении сбоя возможно значительное увеличение тактовой частоты.

Монитор проверяемой частоты осуществляется при наличии частоты генератора HSI, поэтому программное отключение или аппаратное исчезновение частоты HSI ведет к возникновению событий сначала превышения, а затем значительного превышения тактовой частоты контролируемых источников (отключение генератора HSI не контролируется блоком Checker). Если по какому-либо из этих событий настроено аппаратное переключение на HSI-генератор, то это приведет к остановке тактирования, а если настроен аппаратный сброс, то и сбросу микроконтроллера.

Блоки Clock Checker проверяют наличие и частоты тактовых сигналов, поступающих только в цифровую часть микроконтроллера. Блок часов реального времени – RTC, может быть сконфигурирован на использование тактовой частоты от LSE или LSI-генератора, при этом частота на RTC поступает напрямую с генератора и не контролируется блоком, в результате исчезновение частоты LSE приведет к остановке часов реального времени RTC. Этот аспект особенно важен при работе микросхемы в режиме STANDBY с пробуждением от RTC. Так как в этом случае при отказе генератора (LSE или LSI) и остановке часов пробуждение будет невозможно.

Возникающие события от блоков Clock Checker формируют прерывание CLK_IF. Возникающие события, для которых настроено аппаратное переключение на HSI, также обрабатываются в блоке FT_CNTR и на эти события может быть настроено прерывание.

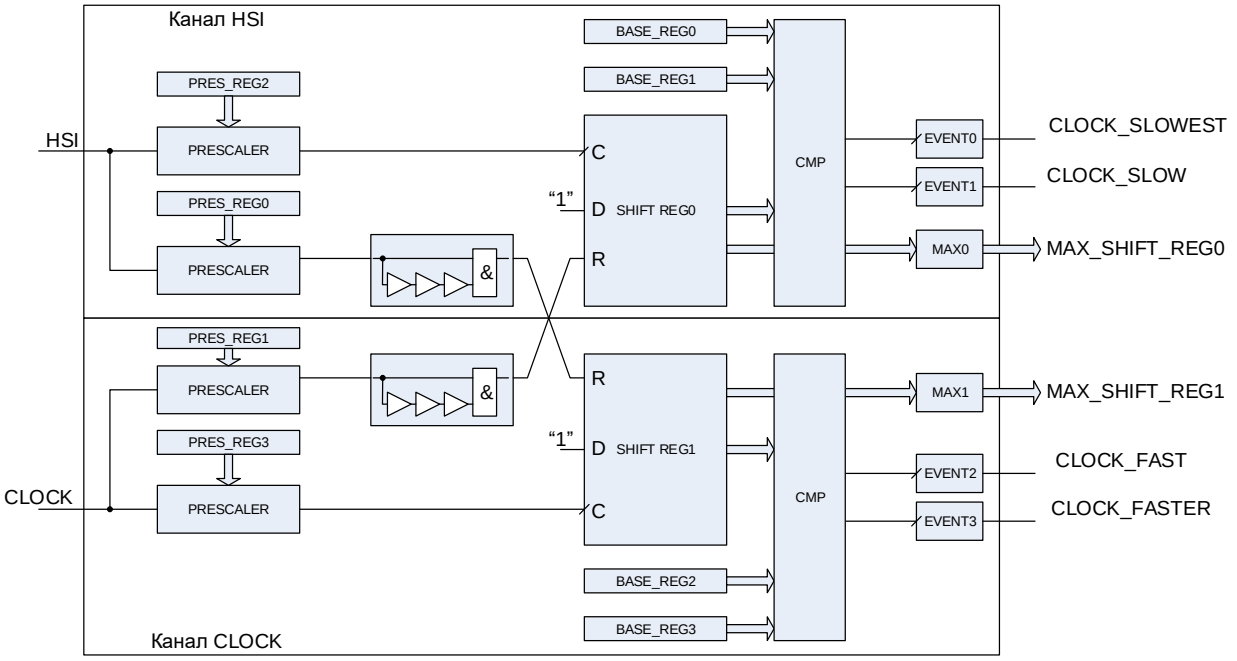



Рисунок 12 – Схема блока контроля тактовых частот

При настройке блока, исходя из ожидаемых значений частот сигналов HSI и CLOCK, необходимо задать значения предварительных делителей в регистрах PRES_REGx, таким образом, чтобы соотношение полученных частот укладывалось в диапазон от 1 до 15.

$$\text{если } f_{HSI} < f_{CLOCK}, \begin{cases} \dfrac{\dfrac{f_{HSI}}{PRES_REG2+1}}{\dfrac{f_{CLOCK}}{PRES_REG1+1}} = \text{K} \\ \dfrac{\dfrac{f_{CLOCK}}{PRES_REG3+1}}{\dfrac{f_{HSI}}{PRES_REG0}} = \text{M} \end{cases}, \tag{4}$$

$$\text{если } f_{HSI} > f_{CLOCK}, \begin{cases} \dfrac{\dfrac{f_{HSI}}{PRES_REG2+1}}{\dfrac{f_{CLOCK}}{PRES_REG1}} = K \\ \dfrac{\dfrac{f_{CLOCK}}{PRES_REG3+1}}{\dfrac{f_{HSI}}{PRES_REG0+1}} = \text{M} \end{cases}, \tag{5}$$

где $f_{HSI}$ – частота сигналов HSI;

$f_{CLOCK}$ – частота сигналов CLOCK;

K є [1; 16] – фактическое соотношение частот, вычисляемое в блоке HSI. На схеме обозначено как SHIFT_REG0;

M є [1; 16] – фактическое соотношение частот, вычисляемое в блоке CLOCK. На схеме обозначено как SHIFT_REG1.

K0 – расчетное соотношение частот K в блоке HSI.

M0 – расчетное соотношение частот M в блоке CLOCK.

При отклонении частоты CLOCK от расчетного значения коэффициенты K и M отклоняются от своих расчетных значений K0 и M0.

При снижении частоты CLOCK значение K изменяется от K0 до 16. Максимальное значение K сохраняется в поле MAX_CHK_SHIFT_REG0 соответствующего регистра ..._STAT.

При достижении K пороговых значений, заданных полями BASE_REG0 и BASE_REG1, вырабатываются события EVENT0 и EVENT1 соответственно.

$$\begin{cases} f_{CLOCK} < \dfrac{f_{HSI} \cdot PRES_REG1^{*}}{(PRES_REG2 + 1) \cdot BASE_REG0} \Rightarrow EVENT0 \\ f_{CLOCK} < \dfrac{f_{HSI} \cdot PRES_REG1^{*}}{(PRES_REG2 + 1) \cdot BASE_REG1} \Rightarrow EVENT1 \end{cases} \quad (6)$$

* $PRES_REG1$ – при $f_{HSI} > f_{CLOCK}$,

$PRES_REG1 + 1$ – при $f_{HSI} < f_{CLOCK}$.

При увеличении частоты CLOCK значение M изменяется от M0 до 16. Максимальное значение M сохраняется в поле MAX_CHK_SHIFT_REG1 соответствующего регистра ..._STAT.

При достижении M пороговых значений, заданных полями BASE_REG2 и BASE_REG3, вырабатываются события EVENT2 и EVENT3 соответственно.

$$\begin{cases} f_{CLOCK} > \dfrac{f_{HSI} \cdot (PRES_REG3 + 1) \cdot BASE_REG2}{PRES_REG0^{*}} \Rightarrow EVENT2 \\ f_{CLOCK} > \dfrac{f_{HSI} \cdot (PRES_REG3 + 1) \cdot BASE_REG3}{PRES_REG0^{*}} \Rightarrow EVENT3 \end{cases} \quad (7)$$

* $PRES_REG0$ – при $f_{HSI} < f_{CLOCK}$,

$PRES_REG0 + 1$ – при $f_{HSI} > f_{CLOCK}$.

Максимальное достигнутое в процессе работы значение регистра SHIFT_REG каждого из каналов можно прочитать программно. При чтении регистра SHIFT_REG считывается теневой регистр, в котором отражается максимальное достигнутое регистром SHIFT_REG значение за период с последнего сброса теневого регистра. Сброс теневого регистра осуществляется программно, записью управляющего бита CLR_CHK_SHIFT_REG.

События EVENT0 и EVENT1 формируются и сбрасываются при записи соответствующих бит на частоте HSI – на этой же частоте формируется значение теневого регистра MAX_CHK_SHIFT_REG0. Аналогично, EVENT2 и EVENT3, а также MAX_CHK_SHIFT_REG1, формируются на проверяемой частоте CLOCK.

После разрешения работы блока необходимо осуществить сброс теневых регистров, а также флагов возникновения событий, с задержкой после разрешения работы. Задержка после разрешения работы блока (бит EN_CHK) до сброса его флагов и теневых регистров должна быть не менее 2 периодов наименьшей из двух частот, поступающих на блок. Это обусловлено наличием двух частотных доменов. При этом сигнал разрешения пересинхронизируется в оба частотных домена, длительность процесса синхронизации блока - не более 2 периодов наименьшей из двух поступающих на блок частот. Блок входит в рабочий режим только после того, как сигнал разрешения будет установлен в обоих доменах, поскольку сигналы из одного тактового домена используются в качестве асинхронных сбросов регистров в другом. Для избежания ложных срабатываний блока Clock Checker настройка аппаратного переключения на частоту HSI и формирования прерывания блоком FT_CNTR должна осуществляться только после вхождения в рабочий режим. Результаты работы блока до его вхождения в рабочий режим следует игнорировать.

Для иллюстрации принципов работы блока возможные комбинации настроек приведены в таблице 23. Фактические значения коэффициентов K и M могут отличаться в связи с подсчетом соотношений частот блоков на двух асинхронных частотах.

Таблица 13 – Примеры работы блока Checker

| Частота HSI | Ожидаемая частота CLOCK | Реальная частота CLOCK | PRES REG2 | PRES REG0 | PRES REG1 | PRES REG3 | BASE REG0 | BASE REG1 | BASE REG2 | BASE REG3 | Примеры |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 МГц | 80 МГц | 80 МГц | 1 | 2 | 199 | 1 | 12 | 15 | 12 | 15 | K0 = 10<br>M0 = 10<br>Нет событий |
| 8 МГц | 80 МГц | 100 МГц | 1 | 2 | 199 | 1 | 12 | 15 | 12 | 15 | K0 = 10<br>M0 = 10<br>K = 8<br>M = 13<br>EVENT2 (Fast Clock) |
| 8 МГц | 80 МГц | 120 МГц | 1 | 2 | 199 | 1 | 12 | 15 | 12 | 15 | K0 = 10<br>M0 = 10<br>K = 6<br>M = 16<br>EVENT2 (Fast Clock)<br>EVENT3 (Faster Clock) |
| 8 МГц | 80 МГц | 60 МГц | 1 | 2 | 199 | 1 | 12 | 15 | 12 | 15 | K0 = 10<br>M0 = 10<br>K = 13<br>M = 8<br>EVENT0 (Slow Clock) |
| 8 МГц | 80 МГц | 0 МГц (нет частоты) | 1 | 2 | 199 | 1 | 12 | 15 | 12 | 15 | K0 = 10<br>M0 = 10<br>K = 16 (∞)<br>M = 0<br>EVENT0 (Slow Clock)<br>EVENT1 (Slowest Clock) |
| 8 МГц | 2,5 МГц | 2,5 МГц | 5 | 89 | 20 | 2 | 12 | 15 | 12 | 15 | K0 = 11<br>M0 = 10<br>Нет событий |
| 8 МГц | 2,5 МГц | 3,33 МГц | 5 | 89 | 20 | 2 | 12 | 15 | 12 | 15 | K0 = 11<br>M0 = 10<br>K = 8<br>M = 13<br>EVENT2 (Fast Clock) |
| 8 МГц | 2,5 МГц | 5 МГц | 5 | 89 | 20 | 2 | 12 | 15 | 12 | 15 | K0 = 11<br>M0 = 10<br>K = 6<br>M = 16<br>EVENT2 (Fast Clock)<br>EVENT3 (Faster Clock) |
| 8 МГц | 2,5 МГц | 2 МГц | 5 | 89 | 20 | 2 | 12 | 15 | 12 | 15 | K0 = 11<br>M0 = 10<br>K = 10<br>M = 8<br>EVENT0 (Slow Clock) |
| 8 МГц | 2,5 МГц | 0 МГц (нет частоты) | 5 | 89 | 20 | 2 | 12 | 15 | 12 | 15 | K0 = 11<br>M0 = 10<br>K = 16 (∞)<br>M = 0 |

| Частота HSI | Ожидаемая частота CLOCK | Реальная частота CLOCK | PRES REG2 | PRES REG0 | PRES REG1 | PRES REG3 | BASE REG0 | BASE REG1 | BASE REG2 | BASE REG3 | Примеры |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | EVENT0 (Slow Clock)<br>EVENT1 (Slowest Clock) |

# 8 Режимы загрузки микроконтроллера

Выбор режима запуска микроконтроллера определяется при включении микроконтроллера или после любого типа сброса. Режим запуска задается уровнями сигналов на входах MODE[4:0] (PA[20:16]), при этом, когда выполняется считывание режима запуска MODE[4:0], вывод MODE[5] (PA[21]) работает на выход, формируя логическую единицу.

Значение MODE[4] является битом четности и дополняет количество единиц в значении режима MODE[3:0] до чётного числа. При обнаружении ошибки по четности микроконтроллер остается в режиме ожидания TEST_MODE+JB в загрузочной программе.

Для определения режима запуска загрузчик выполняет следующие действия:

1. конфигурирует вывод MODE[5] в цифровой режим на выход и устанавливает не нем логическкую единицу;
2. конфигурирует выводы MODE[4:0] в цифровой режим на вход с включенными внутренними резисторами подтяжки ~50 кОм на землю;
3. выдерживает паузу ~300 мкс для переопределения потенциалов на выводах MODE[4:0];
4. производит тройное считывание значений выводов MODE[4:0] для выполнения мажоритарного контроля;
5. конфигурирует выводы MODE[5:0] в аналоговый режим;
6. выполняет мажоритарный контроль считанных значений MODE[4:0] и определяет результирующее значение MODE[4:0];
7. выполняет проверку на четность результирующего значения MODE[4:0]. Если проверка прошла успешно, то значение MODE[3:0] записывается в поле MODE[3:0] регистров REG_60_SYSx. При неудачной проверке на четность в поле MODE[3:0] регистров REG_60_SYSx записывается режим TEST_MODE+JB;
8. на основании конечного значения MODE[3:0] осуществляется переход в выбранный режим.

Таблица 14 – Режимы загрузки микроконтроллера

| **Бит чётности MODE[4]** | **Биты режима MODE[3:0]** | **Режим** | **Краткое описание** |
|---|---|---|---|
| 0 | 0000 | WAIT_BOOT_JA | Ожидание в бесконечном цикле с включенным интерфейсом отладки через выводы JTAG_A |
| 1 | 0001 | FLASH+JB | Запуск из FLASH (0x0100_0000) с включенным интерфейсом отладки через выводы JTAG_B |
| 1 | 0010 | FLASH+JA | Запуск из FLASH (0x0100_0000) с включенным интерфейсом отладки через выводы JTAG_A |
| 0 | 0011 | EXTBUS_8_ECC+JB* | Запуск из внешней памяти с (0x1000_0000), сконфигурированной в минимальный режим:<br>– шина данных D[7:0] (PC[7:0])<br>– шина адреса A[15:0] (PA[16:31])<br>– сигнал WEn (PB[14])<br>– сигнал OEn (PB[15])<br>с последовательной организацией ECC с базового адреса 0x1000_9000 и с включенным интерфейсом отладки через выводы JTAG_B |

| Бит чётности MODE[4] | Биты режима MODE[3:0] | Режим | Краткое описание |
|---|---|---|---|
| 1 | 0100 | EXTBUS _8_ECC+JA* | Запуск из внешней памяти с (0x1000_0000), сконфигурированной в минимальный режим:<br>– шина данных D[7:0] (PC[7:0])<br>– шина адреса A[15:0] (PA[16:31])<br>– сигнал WEn (PB[14])<br>– сигнал OEn (PB[15])<br>с последовательной организацией ECC с базового адреса 0x1000_9000 и с включенным интерфейсом отладки через выводы JTAG_A |
| 0 | 0101 | EXTBUS_CFG+JB* | Запуск из внешней памяти с (0x1000_0000) с чтением конфигурации в режиме:<br>– шина данных D[7:0] (PC[7:0])<br>– шина адреса A[10:3] (PA[21:28])<br>– сигнал WEn (PB[14])<br>– сигнал OEn (PB[15])<br><br>Читается DATA[7:0] = { ECC[3:0], CFGx[3:0] }, где:<br>– CFG0 = Режим запуска (8, 16, 32)<br>– CFG1 = Режим ECC (нет, послед., параллельный)<br>– CFG2 = ECCBASE[3:0]<br>– CFG3 = ECCBASE[7:4]<br>…<br>– CFG9 = ECCBASE[31:28]<br>с включенным интерфейсом отладки через выводы JTAG_B |
| 0 | 0110 | EXTBUS_CFG+JA* | Запуск из внешней памяти с (0x1000_0000) с чтением конфигурации в режиме:<br>– шина данных D[7:0] (PC[7:0])<br>– шина адреса A[10:3] (PA[21:28])<br>– сигнал WEn (PB[14])<br>– сигнал OEn (PB[15])<br><br>Читается DATA[7:0] = { ECC[3:0], CFGx[3:0] }, где:<br>– CFG0 = Режим запуска (8, 16, 32)<br>– CFG1 = Режим ECC (нет, послед., параллельный)<br>– CFG2 = ECCBASE[3:0]<br>– CFG3 = ECCBASE[7:4]<br>…<br>– CFG9 = ECCBASE[31:28]<br>с включенным интерфейсом отладки через выводы JTAG_A |
| 1 | 0111 | SPI0+JB | Загрузка из внешней памяти (серия 5576 и аналогичные) по последовательному интерфейсу:<br>– сигнал DCLK (PB[20])<br>– сигнал CONF_DONE (PB[21])<br>– сигнал DATA (PB[22])<br>– сигнал nSTATUS (PB[23])<br>с включенным интерфейсом отладки через выводы JTAG_B |

| Бит чётности MODE[4] | Биты режима MODE[3:0] | Режим | Краткое описание |
|---|---|---|---|
| 1 | 1000 | SPI0+JA | Загрузка из внешней памяти (серия 5576 и аналогичные) по последовательному интерфейсу:<br>– сигнал DCLK (PB[20])<br>– сигнал CONF_DONE (PB[21])<br>– сигнал DATA (PB[22])<br>– сигнал nSTATUS (PB[23])<br>с включенным интерфейсом отладки через выводы JTAG_A |
| 0 | 1001 | SPI1+JB | Загрузка из внешней памяти по последовательному интерфейсу SPI:<br>– сигнал CLK (PB[16])<br>– сигнал FS (PB[17])<br>– сигнал RX (PB[18])<br>– сигнал TX (PB[19])<br>с включенным интерфейсом отладки через выводы JTAG_B |
| 0 | 1010 | SPI1+JA | Загрузка из внешней памяти по последовательному интерфейсу SPI:<br>– сигнал CLK (PB[16])<br>– сигнал FS (PB[17])<br>– сигнал RX (PB[18])<br>– сигнал TX (PB[19])<br>с включенным интерфейсом отладки через выводы JTAG_A |
| 1 | 1011 | UART0+JB | Загрузка по последовательному интерфейсу UART:<br>– сигнал RX (PC[0])<br>– сигнал TX (PC[1])<br>с включенным интерфейсом отладки через выводы JTAG_B |
| 0 | 1100 | UART0+JA | Загрузка по последовательному интерфейсу UART:<br>– сигнал RX (PB[0])<br>– сигнал TX (PB[1])<br>с включенным интерфейсом отладки через выводы JTAG_A |
| 1 | 1101 | - | Резерв |
| 1 | 1110 | SRAMC+JA | Запуск из SRAMC (0x0200_0000) с включенным интерфейсом отладки через выводы JTAG_A |
| 0 | 1111 | TEST_MODE+JB | Тестовый режим микросхемы для отбраковочного тестирования с включенным интерфейсом отладки через выводы JTAG_B |
| | Ошибка четности | TEST_MODE+JB | Ожидание в бесконечном цикле с включенным интерфейсом отладки через выводы JTAG_B |

* В данном режимах загрузки адресные выводы EXTBUS конфликтуют с выводами генераторов HSEx.

Примечание – Диапазон адресов ОЗУ 0x2001_FE00-0x2002_0000 в микроконтроллерах используется загрузочной программой для работы со стеком и кучей. Необходимо учитывать данную особенность при работе в разных режимах работы микроконтроллера, особенно, в режимах SPI.

В зависимости от выбранного режима работы выводы интерфейса JTAG/SW переопределяются на различные выводы микроконтроллера: JTAG_A (порт A) или JTAG_B (порт B). Выбор выводов для отладки определяется в троированных регистрах батарейного домена REG_60_SYSx битами MODE[0] и DISABLE_JTAG. В ходе работы микроконтроллера выводы для отладки могут быть изменены путём настройки регистров REG_60_SYSx. При выборе того или иного интерфейса данные выводы жестко переопределяются на выполнение функции JTAG.

### 8.1 Режим WAIT_BOOT_JA

Ожидание в бесконечном цикле с включенным интерфейсом отладки через выводы JTAG_A.

### 8.2 Режим TEST_MODE+JB

В данном режиме на выводы JTAG_B вместо TAP контроллера процессорных ядер подключается технологический TAP контроллер, позволяющий перевести микросхему в тестовые режимы.

### 8.3 Режим FLASH+JA

После определения данного режима загрузчик выполняет следующие действия:

- устанавливает адрес указателя стека в значение, считанное из ячейки с адресом 0x0100_0000 (SP);
- устанавливает адрес таблицы векторов в значение 0x0100_0000 (VTOR);
- осуществляет безусловный переход по адресу, считанному из ячейки 0x0100_0004 (RESET_HANDLER).

В специальной структуре в ОЗУ передаются флаги об ошибках в процессе работы загрузчика.

Отладка через выводы JTAG_A.

### 8.4 Режим FLASH +JB

Аналогично FLASH_JA, за исключением того, что выполняется отладка через выводы JTAG_B.

### 8.5 Режим EXTBUS_8_ECC+JA

После определения данного режима загрузчик конфигурирует внешнюю системную шину в режим:

- шина данных D[7:0] (PC[7:0]);
- шина адреса A[15:0] (PA[16:31]);
- сигнал WEn (PB[14]);
- сигнал OEn (PB[15]);
- режим ECC: последовательная организация 8 бит ECC, проверочные биты расположены с базового адреса 0x1000_9000;
- длительность фаз во время транзакции на внешней системной шине:
  - фаза предустановки WS_SETUP = 4;
  - активная фаза WS_ACTIVE = 127;
  - фаза удержания WS_HOLD = 4.

После конфигурации внешней системной шины загрузчик выполняет следующие действия:

- устанавливает адрес указателя стека в значение, считанное из ячейки с адресом 0x1000_0000 (SP);
- устанавливает адрес таблицы векторов в значение 0x1000_0000 (VTOR);

- осуществляет безусловный переход по адресу, считанному из ячейки 0x1000_0004 (RESET_HANDLER).

В специальной структуре в ОЗУ передаются флаги об ошибках в процессе работы загрузчика.

Отладка выполняется через выводы JTAG_A.

### 8.6 Режим EXTBUS_8_ECC+JB

Аналогично EXTBUS_8_ECC+JA, за исключением того, что выполняется отладка через выводы JTAG_B.

### 8.7 Режим EXTBUS_CFG+JA

После определения данного режима загрузчик конфигурирует внешнюю системную шину в два этапа.

Сначала внешняя системная шина конфигурируется для чтения из внешней памяти конфигурационных бит CFGx, при этом устанавливается режим работы:

- шина данных D[7:0] (PC[7:0]);
- шина адреса A[10:3] (PA[21:28]);
- сигнал WEn (PB[14]);
- сигнал OEn (PB[15]);
- режим ECC: контроль ECC не выполняется;
- длительность фаз во время транзакции на внешней системной шине:
  - фаза предустановки WS_SETUP = 4;
  - активная фаза WS_ACTIVE = 127;
  - фаза удержания WS_HOLD = 4.

Из ячеек по адресам 0x1000_0400, 0x1000_0408, … 0x1000_0448 читается DATA[7:0] = {ECC[3:0], CFGx[3:0]} и выполняется программный контроль ECC. При обнаружении одиночной ошибки загрузчик исправляет её, при обнаружении двойной ошибки – формирует сигнал программного сброса SYS_RESET_REQ.

После проверки загрузчик конфигурирует внешнюю системную шину в зависимости от считанных значений CFGx:

- шина данных (в зависимости от CFG0):
  - CFG0 = 1 - 8-ми битная шина данных D[7:0] (PC[7:0]);
  - CFG0 = 2 - 16-ти битная шина данных D[15:0] (PC[15:0]);
  - CFG0 = 3 - 32-х битная шина данных D[31:0] (PC[31:0]);
- шина адреса A[20:0] (PA[11:31]);
- сигнал WEn (PB[14]);
- сигнал OEn (PB[15]);
- режим ECC (в зависимости от CFG1):
  - CFG1 = 1 – контроль ECC не выполняется;
  - CFG1 = 2 – последовательная организация 8 бит ECC. Базовый адрес ECC устанавливается из CFG9 - CFG2:
    - CFG2 = ECCBASE[3:0];
    - CFG3 = ECCBASE[7:4];
    - …
    - CFG9 = ECCBASE[31:28];
  - CFG1 = 3 – параллельная организация 8 бит ECC, используются разряды шины данных D[39:32] (PA[7:0]), при этом интерфейс JTAG_A становится недоступен;
- длительность фаз во время транзакции на внешней системной шине:
  - фаза предустановки WS_SETUP = 4;
  - активная фаза WS_ACTIVE = 127;

- фаза удержания WS_HOLD = 4.

Параллельная организация ECC (CFG1 = 3) может быть задана только при выборе 32-битной шины данных (CFG0 = 3). При некорректных значениях CFG0 или CFG1 загрузочная программа формирует сигнал программного сброса SYS_RESET_REQ.

После переопределения внешней системной шины загрузчик выполняет следующие действия:

- устанавливает адрес указателя стека в значение, считанное из ячейки с адресом 0x1000_0000 (SP);
- устанавливает адрес таблицы векторов в значение 0x1000_0000 (VTOR);
- осуществляет безусловный переход по адресу, считанному из ячейки 0x1000_0004 (RESET_HANDLER).

В специальной структуре в ОЗУ передаются флаги об ошибках в процессе работы загрузчика.

Отладка выполняется через выводы JTAG_A.

### 8.7.1 *Расчет ECC для поля CFG*

Таблица 15 – Значение CFG и соответствующее ему ECC.

| CFG | ECC |
|---|---|
| 0x0 | 0x0 |
| 0x1 | 0x7 |
| 0x2 | 0xB |
| 0x3 | 0xC |
| 0x4 | 0xD |

## 8.8 Режим EXTBUS_CFG+JB

Аналогично EXTBUS_8_ECC+JA, за исключением того, что выполняется отладка через выводы JTAG_B.

## 8.9 Режим SPI0+JA

Данный режим необходим для записи в ОЗУ какой-либо программы (в частности программатора Flash-памяти), верификации ее и запуска на выполнение.

После определения данного режима загрузчик конфигурирует контроллер SSP1 для работы с микросхемами памяти для конфигурирования ПЛИС (5576РТ1У, 5576РС1У и аналогичные) по последовательному интерфейсу:

- PB[20] – DCLK
- PB[21] – CONF_DONE
- PB[22] – DATA
- PB[23] – nSTATUS

и отладкой через выводы JTAG_A. В специальной структуре в ОЗУ передаются флаги об ошибках в процессе работы загрузчика.

В качестве источника тактовой частоты SSP1 используется внутренний RC-генератор HSI с частотой ~8 МГц.

После конфигурирования выводов загрузочная программа ожидает высокого уровня на выводе nSTATUS, что свидетельствует о готовности к работе внешней FLASH. После обнаружения высокого логического уровня из внешней FLASH последовательно считывается информационный заголовок (см. таблицу 16), начиная с **Target address**. Далее в соответствии со считанными параметрами выполняется загрузка образа программы во внутреннюю память ОЗУ. Все данные во внешней FLASH должны иметь порядок байт «little-endian», при этом все байты должны иметь обратный порядок бит.

После загрузки образа программы во внутреннюю память ОЗУ загрузочная программа осуществляет безусловный переход по адресу **Entry address**. Адрес указателя

стека (SP) и адрес таблицы векторов (VTOR) не конфигурируются и должны быть установлены в пользовательской программе.

### 8.9.1 *Параметры связи по SPI*

Для связи по SPI выбраны следующие параметры канала связи:
- Частота сигнала DCLK – 100 кГц;
- Количество бит данных – 8;
- Формат синхронного обмена – SPI фирмы Motorola, SPO=1, SPH=0.

### 8.9.2 *Протокол обмена по SPI*

Таблица 16 – Структура информационного заголовка

| Ext. Flash Address | Ext. Flash data | |
|---|---|---|
| 0x000000 | 32 bit | INF offset |
| ... | *Flash config data** | |
| offset | 32 bit | Target address |
| offset+0x4 | 32 bit | Entry address |
| offset+0x8 | 32 bit | IMG length |
| offset+0xC | 16 bit | CRC16 over data cover |
| offset+0xE | 16 bit | CRC16 over INF |
| offset+0x10<br>... | IMG data + CRC16 | |

*В некоторых FLASH начальные адреса зарезервированы для конфигурации.

- **INF offset** – адрес начала информационного заголовка, может быть равно нулю;
- **Target address** – адрес, по которому будет расположен образ копируемой программы во внутренней памяти ОЗУ;
- **Entry address** – 32-разрядный адрес, которому будет передано управление после копирования программы. Как правило, это адрес обработчика RESET_HANDLER, расположенный в таблице векторов прерываний по смещению 0x4. Младший бит **Entry address** должен быть установлен в «1»;
- **IMG lenght** – 32-разрядная длина образа программы в словах (32 бит) без учета CRC16;
- **CRC16 over data cover** – длина блоков в словах (32 бит), на которые разбивается образ программы для вычисления CRC16. Для каждого блока программы рассчитывается индивидуальное значение CRC16. В образе программы значение CRC16 располагается сразу после блока программы, на основании которого оно было рассчитано, как показано в таблице 17. При указании этого поля равным нулю значение CRC16 от образа программы не рассчитывается;
- **CRC16 over INF** – значение контрольной суммы CRC16 информационного заголовка без учета поля CRC16.

После чтения информационного заголовка верифицируются считанное и рассчитанное значение CRC16 по информационному заголовку. При несовпадении формируется сигнал программного сброса SYS_RESET_REQ. После успешной проверки CRC16 загрузочная программа последовательно считывает образ программы.

Таблица 17 – Структура образа программы

| Ext. Flash Address | Ext. Flash data | |
|---|---|---|
| … | | |
| offset+0x10 | 32 bit | IMG data word_0 |
| offset+0x14 | 16 bit | CRC16 over word_0 |
| offset+0x16 | 32 bit | IMG data word_1 |
| offset+0x20 | 16 bit | CRC16 over word_1 |

Последовательность загрузки образа программы во внутреннюю ОЗУ:

- Блок данных последовательно считывается и записывается во внутреннюю память ОЗУ, начиная с адреса **Target Address.** При этом увеличивается внутренний счетчик слов данных.
- Считывается CRC16 и верифицируется с рассчитанным значением по блоку данных (при несовпадении CRC16 формируется сигнал программного сброса SYS_RESET_REQ);
- Блок данных последовательно считывается и записывается во внутреннюю память ОЗУ. При этом увеличивается внутренний счетчик слов данных.
- Считывается CRC16 и верифицируется с рассчитанным значением по блоку данных (при несовпадении CRC16 формируется сигнал программного сброса SYS_RESET_REQ);
- и.т.д., пока внутренний счетчик данных не станет равным **IMG lenght**.

В данном случае образ программы соответствует ситуации, когда для расчёта каждого значения CRC16 используется одно слово данных программы (**CRC16 over data cover** = 1). Таким образом, в зависимости от значения поля **CRC16 over data cover** устанавливается необходимая избыточность проверочных данных (CRC16).

### 8.9.3 *Вычисление CRC16*

Для вычисления CRC16 используется полином вида: $x^{16} + x^{15} + x^2 + 1$ (0xA001) с начальным значением CRC16 равным 0xFFFF:

```
uint16_t crc16_update (uint16_t crc, uint8_t a)
{
   int i;

   crc ^= a;
   for (i = 0; i < 8; ++i)
   {
      if (crc & 1)
         crc = (crc >> 1) ^ 0xA001;
      else
         crc = (crc >> 1);
   }
   return crc;
}
```

## 8.10 Режим SPI0+JB

Аналогично SPI0+JA, за исключением того, что отладка выполняется через выводы JTAG_B.

### 8.11 Режим SPI1+JA

Данный режим необходим для записи в ОЗУ какой-либо программы (в частности программатора Flash-памяти), верификации ее и запуска на выполнение.

После определения данного режима загрузчик конфигурирует контроллер SSP0 для работы с микросхемами памяти по интерфейсу SPI:

- PB[16] – CLK
- PB[17] – FS
- PB[18] – RX
- PB[19] – TX

и отладкой через выводы JTAG_A. В специальной структуре в ОЗУ передаются флаги об ошибках в процессе работы загрузчика.

В качестве источника тактовой частоты SSP0 используется внутренний RC-генератор HSI с частотой ~8 МГц.

После конфигурирования выводов загрузочная программа считывает из внешней FLASH по адресу 0x000000 32-разрядный адрес смещения информационного заголовка **INF offset**. Если смещение отлично от нуля, то операция чтения прерывается и возобновляется по адресу **INF offset**. Далее последовательно считывается информационный заголовок, приведённый в таблице 16, после чего в соответствии со считанными параметрами выполняется загрузка образа программы во внутреннюю память ОЗУ. Все данные во внешней FLASH должны иметь порядок байт «little-endian».

После загрузки образа программы во внутреннюю память ОЗУ загрузочная программа осуществляет безусловный переход по адресу **Entry address**. Адрес указателя стека (SP) и адрес таблицы векторов (VTOR) не конфигурируются и должны быть установлены в пользовательской программе.

#### 8.11.1 *Параметры связи по SPI*

Для связи по SPI выбраны следующие параметры канала связи:

- Частота сигнала CLK – 100 кГц;
- Количество бит данных – 8;
- Формат синхронного обмена – SPI фирмы Motorola, SPO = 0, SPH = 0.

Для чтения данных используется команда Read Array: код команды – 0x03, количество байт адреса – 3.

#### 8.11.2 *Протокол обмена по SPI*

Аналогично режиму SPI0+JA (см.8.9.2 «Протокол обмена по SPI»).

### 8.12 Режим SPI1+JB

Аналогично SPI1+JA, за исключением того, что отладка выполняется через выводы JTAG_B.

### 8.13 Режим UART0+JA

Данный режим предоставляет достаточный набор операций, необходимых для записи в ОЗУ какой-либо программы (в частности программатора Flash-памяти), верификации ее и запуска на выполнение. Кроме того, существует возможность изменения внешним устройством скорости обмена. Помимо доступа к ОЗУ может быть осуществлен доступ и к другим адресным диапазонам.

После определения данного режима загрузчик конфигурирует контроллер UART0:

- PB[0] – RX
- PB[1] – TX

с отладкой через выводы JTAG_A. В специальной структуре в ОЗУ передаются флаги об ошибках в процессе работы загрузчика.

В качестве источника тактовой частоты UART0 используется внутренний RC-генератор HSI с частотой ~8 МГц. Так как имеется разброс значений частоты HSI, то требуется этап подбора значения делителя частоты UART0 для синхронизации с внешним устройством.

### 8.13.1 *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 UART0;
- загрузчик всегда выступает в качестве Slave, а внешнее устройство, подающее команды – в качестве Master.

### 8.13.2 *Протокол обмена по UART*

После синхронизации с Master загрузчик переходит в диспетчер команд. Для управления устройству Master доступны команды, приведённые в таблице 18.

Таблица 18 – Список команды диспечера команд 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 мастеру (Master) |
| 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 |

### 8.13.3 *Синхронизация с внешним устройством*

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

### 8.13.4 *Команда CMD_SYNC*

Пустая команда.

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

Таблица 19 – 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. |

### 8.13.5 *Команда CMD_CR*

Выдача приглашения.

Таблица 20 –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.<br>Иначе выдаёт 3 байта:<br>- код команды CMD_CR;<br>- код 0x0A;<br>- код 0x3E (ASCII символ '>'). |

### 8.13.6 *Команда CMD_BAUD*

Установка скорости обмена.

Таблица 21 – Команда CMD_BAUD

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

### 8.13.7 *Команда CMD_LOAD*

Загрузка массива байт в память микроконтроллера.

Таблица 22 – Команда CMD_LOAD

| **Код команды** | **CMD_LOAD = 0x4C** |
|---|---|
| ASCII символ, соответствующий коду команды | 'L' |
| Количество параметров команды | 2 |
| Параметр 1 | Адрес памяти приемника данных |
| Параметр 2 | Размер массива в байтах |
| **Формат команды:** | |
| Master: Выдает код команды CMD_LOAD | Slave: Если команда принята с ошибками, то выдает код ошибки ERR_CHN или ERR_CMD и завершает обработку текущей команды. |
| Master: Выдает параметр 1 | Slave: Ожидает получения всех параметров. |
| Master: Выдает параметр 2 | Slave: Если хотя бы один из параметров принят с ошибками, то выдает код ошибки ERR_CHN и завершает обработку текущей команды.<br>Иначе, выдает код команды CMD_LOAD. |

| | |
|---|---|
| Master: Выдает массив байт младшим байтом вперед | Slave: Принимает массив байт. Если хотя бы один байт принят с ошибками, то выдает код ошибки ERR_CHN и завершает обработку текущей команды, не дожидаясь окончания принятия всего массива.<br>По окончании приёма массива выдает код ответа REPLY_OK = 0x4B ('K'). |

### 8.13.8 *Команда CMD_VFY*

Выдача массива байт из памяти микроконтроллера.

Таблица 23 – Команда CMD_VFY

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

### 8.13.9 *Команда CMD_RUN*

Запуск программы на выполнение.

Таблица 24 – Команда CMD_RUN

| **Код команды** | **CMD_RUN = 0x52** |
|---|---|
| ASCII символ, соответствующий коду команды | 'R' |
| Количество параметров команды | 1 |
| Параметр | Адрес первой команды загруженной программы (младший бит адреса необходимо установить в 1). Как правило, это адрес обработчика RESET_HANDLER, расположенный в таблице векторов прерываний по смещению 0x4 |
| **Формат команды:** | |
| Master: Выдает код команды CMD_RUN | Slave: Если команда принята с ошибками, то выдает код ошибки ERR_CHN или ERR_CMD и завершает обработку текущей команды. |
| Master: Выдает параметр | Slave: Если параметр принят с ошибками, то выдает код ошибки ERR_CHN и завершает обработку текущей команды.<br>Иначе:<br>- выдает код команды CMD_RUN;<br>- устанавливает значение PC согласно принятому адресу и, таким образом, Slave завершает свое выполнение. Адрес указателя стека (SP) и адрес таблицы векторов (VTOR) не конфигурируются и должны быть установлены в пользовательской программе. |

| | |
|---|---|
| | Передача управления загруженной программе происходит, не дожидаясь окончания отправки кода команды CMD_RUN. |

### 8.13.10 *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.

### 8.13.11 *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**

Код ошибки 0x62 ('b').

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

## 8.14 Режим UART0+JB

Аналогично UART0+JA, за исключением того, что конфигурируются выводов для обмена по UART RX(PC[0]) и TX(PC[1]), а также отладка выполняется через выводы JTAG_B.

## 8.15 Статус загрузчика

Последние четыре слова в ОЗУ (RAMD) используются для передачи ошибок, которые могли возникнуть в процессе загрузки контроллера. Таким образом, после завершения работы загрузочной программы и передачи управления, ошибки могут быть считаны из ОЗУ для корректирования режима и минимизации ошибок при последующих запусках.

Таблица 25 – Структура загрузчика ОЗУ.

| Адрес ОЗУ | Размер, бит | Название | Описание |
|---|---|---|---|
| 0x2001_7FF0 | 8 | INIT_ERR_CNTR | Число повторных инициализаций выводов MODE[5:0].<br>Значение INIT_ERR_CNTR >2 свидетельствует об ошибке в контроллере порта. |
| 0x2001_7FF1 | 8 | MODE_0 | Первое считанное значение MODE[4:0] |
| 0x2001_7FF2 | 8 | MODE_1 | Второе считанное значение MODE[4:0] |
| 0x2001_7FF3 | 8 | MODE_2 | Третье считанное значение MODE[4:0] |
| 0x2001_7FF4 | 8 | MODE_MAJ | Значение MODE[4:0] после мажоритарного контроля MODE_0, MODE_1 и MODE_2 |
| 0x2001_7FF5 | 8 | MODE_MAJ_ERR | Ошибка мажоритарного контроля:<br>0 – нет различий между MODE_0, MODE_1 и MODE_2<br>1 – есть различия между MODE_0, MODE_1 и MODE_2 |
| 0x2001_7FF6 | 16 | - | Зарезервировано |
| 0x2001_7FF8 | 8 | PARITY_ERR | Ошибка чётности MODE[4:0] (MODE_MAJ):<br>0 – нет ошибки, в MODE[4:0] чётное число единиц<br>1 – есть ошибка, в MODE[4:0] нечётное число единиц |
| 0x2001_7FF9 | 8 | MODE_CORR | Скорректированное значение MODE[4:0], которое используется для определения режима загрузки.<br>Если ошибка чётности PARITY_ERR не обнаружена, то MODE_CORR совпадает с MODE_MAJ. Иначе в MODE_CORR записывается значение TEST_MODE+JB. |
| 0x2001_7FFA | 8 | MODE_EXTBUS_CFG0 | Исправленное значение конфигурации CFG0 в режиме EXTBUS_CFG+JB или EXTBUS_CFG+JA |
| 0x2001_7FFB | 8 | MODE_ECC_EXTBUS_CFG1 | Исправленное значение конфигурации CFG1 в режиме EXTBUS_CFG+JB или EXTBUS_CFG+JA |
| 0x2001_7FFC | 32 | ECC_BASE_EXTBUS_CFG2-CFG9 | Исправленное значение базового адреса ECC CFG2-CFG9 в режиме EXTBUS_CFG+JB или EXTBUS_CFG+JA |

## 8.16 Тестовый режим работы

Микросхема переводится в тестовый режим:

- при установке высокого уровня на выводе JTAG_EN;
- при определении загрузочной программой режима MODE[4:0] = 01111;
- при установке поля MODE[3:0] = 1111 в регистрах REG_60_SYSx контроллера батарейного домена.

При переходе в тестовый режим работа микросхемы в функциональном режиме останавливается.

В тестовом режиме к выводам JTAG_A подключается тестовый TAP-контроллер. Регистр инструкций (IR) тестового TAP-контроллера имеет длину 6 бит. Загрузка инструкций в регистр IR переводит микросхему в тестовые режимы, указанные в таблице 26.

Таблица 26 – Инструкции тестового TAP-контроллера

| IR[5:0] | Название инструкции | Описание |
|---|---|---|
| 0x37 | IDCODE | В качестве регистра данных выбирается регистр ID, который имеет значение 0x03E00FDF. |
| 0x3C | EXTEST | Переводит микросхему в режим внешнего граничного тестирования. В качестве регистра данных выбирается регистр граничного сканирования. |
| 0x3D | SAMPLE | Позволяет считать регистр граничного сканирования, который выбирается в качестве регистра данных. |
| 0x3E | PRELOAD | Позволяет загрузить регистр граничного сканирования, который выбирается в качестве регистра данных. |
| 0x3F | BYPASS | В качестве регистра данных выбирается 1-битный регистр. Используется для обеспечения последовательного пути минимальной длины между выводами TDI и TDO. |

# 9 Организация памяти

Микроконтроллер имеет Фон-Неймановскую архитектуру памяти, то есть все адресное пространство (память программ и память данных) отображены в общее адресное пространство размером 4 Гбайт. Но при этом память разделена на две секции: CODESECTION (от 0x00000000 до 0x1FFFFFFF) для кода программ и DATASECTION (от 0x20000000 до 0xDFFFFFF) для данных.

Доступ к CODESECTION осуществляется через шины IBUS и DBUS ядра микроконтроллера. Доступ к DATASECTION осуществляется через шину SBUS. Блок прямого доступа к памяти DMA имеет выделенную шину MBUS и может обращаться к памяти в диапазоне DATASECTION.

Код программы может располагаться в DATASECTION, но при его выполнении снижается производительность из-за возникающих конфликтов по доступу за инструкциями и данными. Данные могут располагаться в CODESECTION в областях RAMC или внешней памяти ОЗУ, отображаемой в эту секцию.

Диапазоны адресов для различных областей памяти и периферийных блоков смотрите в соответствующих пунктах раздела «Программная модель микроконтроллера».

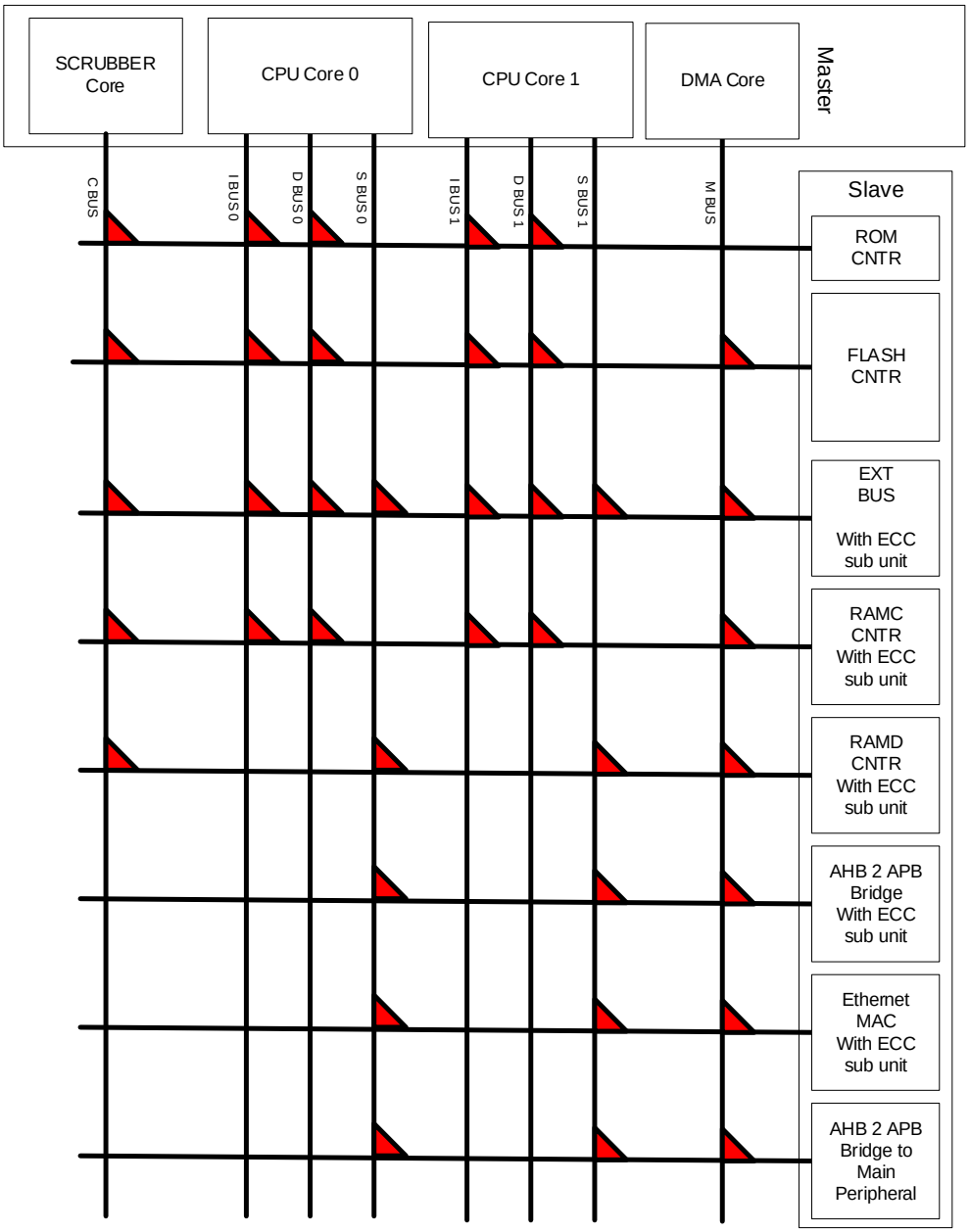



Рисунок 13 – Схема организации доступа системных шин к различным областям ИМС

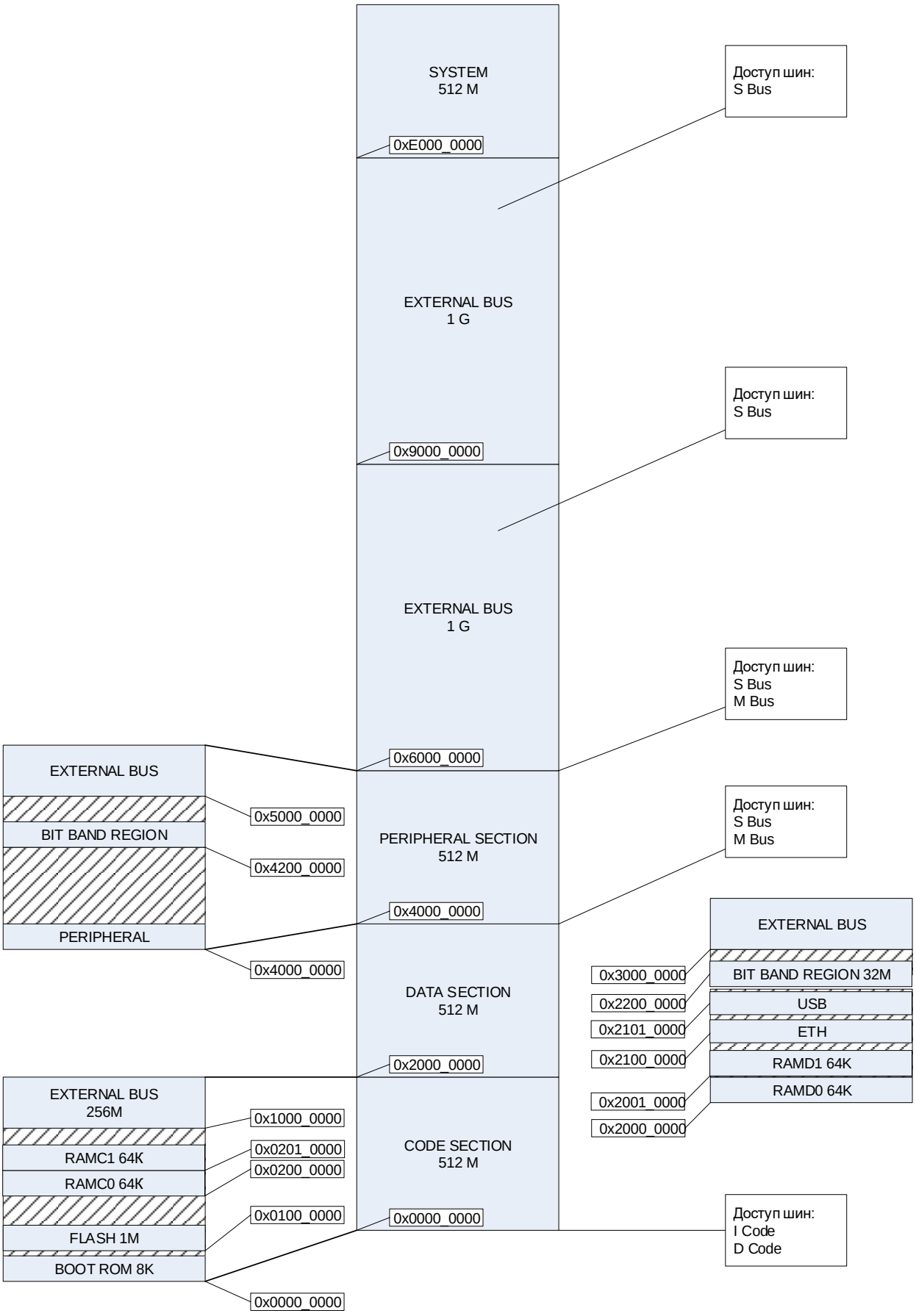



Рисунок 14 – Адресное пространство микроконтроллера

## 9.1 Контроллер кэш-памяти (CACHE_CNTR)

В микросхеме для каждого из процессорных ядер реализованы кэш инструкций (I-CACHE) и кэш данных (D-CACHE) размером 128 байт каждый. При включении питания оба блока выключены и при необходимости могут быть включены программно в регистре CNTR контроллера CACHE_CNTR.

При выключении питания кэш-память обнуляется. Также кэш память может быть сброшена установкой бита CLR_CACHE регистра CNTR. При включении кэш-памяти заполнение происходит автоматически по ходу выполнения программы. Алгоритм обновления данных в кэш-памяти построен по принципу FIFO в I-Cache и PLRU в D-Cache.

В кэш-памяти кэшируются данные из FLASH и диапазона внешней шины EXT_BUS, обращения к которым осуществляется через шины IBus и DBus. При этом не обеспечивается аппаратная поддержка когерентности закэшированных данных, которые могут быть изменены в ходе работы. Обеспечения когерентности этого региона реализуется пользовательской программой путем перезапуска кэш-памяти. Кэш память состоит из записей тега адреса и четырех слов данных с ECC кодами.

| **Тэг Адреса 32 бита** | **Данные 32 + 8ECC бита** | | **Данные 32 + 8ECC бита** | | **Данные 32 + 8ECC бита** | | **Данные 32 + 8ECC бита** | |
|---|---|---|---|---|---|---|---|---|
| ADDR0 | DATA[ADDR0+0x0] | ECC | DATA[ADDR0+0x4] | ECC | DATA[ADDR0+0x8] | ECC | DATA[ADDR0+0xC] | ECC |
| ADDR1 | DATA[ADDR1+0x0] | ECC | DATA[ADDR1+0x4] | ECC | DATA[ADDR1+0x8] | ECC | DATA[ADDR1+0xC] | ECC |
| ADDR2 | DATA[ADDR2+0x0] | ECC | DATA[ADDR2+0x4] | ECC | DATA[ADDR2+0x8] | ECC | DATA[ADDR2+0xC] | ECC |
| … | … | | … | | … | | … | |
| ADDRm | DATA[ADDRm+0x0] | ECC | DATA[ADDRm+0x4] | ECC | DATA[ADDRm+0x8] | ECC | DATA[ADDRm+0xC] | ECC |

Рисунок 15 – *Организация CACHE-памяти*

Значение ECC, сохраняемое в кэш-памяти, позволяет исправить одиночную ошибку и обнаружить двойную в поле данных и обнаружить одиночную и двойную ошибку в поле адреса.

При исполнении кода программы из FLASH в кэш-память записываются строки целиком, то есть из FLASH-памяти извлекается 160-битное слово (4 × 32 бита данных и 4 × 8 бита ECC), которые по выделенному каналу записываются в кэш-память. В зависимости от того, по какой шине (I или D) происходит обращение, в I- или D- кэш будут записаны данные.

При этом, в контроллере FLASH проверку проходят только те слова, которые выдавались по запросам контроллера, в cache заносится вся строка данных, таким образом в случае возникновения сбоев в памяти, возможно заполнение cache ошибочными данными. При выдаче из cache микропроцессорному ядру данные проходят проверку на ECC, таким образом в случае попадания в cache данных с невыявленными ранее ошибками, ошибка будет выдана на этапе выборки из cache. Возникновение ошибки означает, что данные в cache повреждены. Таким образом, для предотвращения ошибки в дальнейшем необходимо программно проанализировать причину возникновения ошибки, после чего осуществить очистку всего буфера cache.

При исполнении кода или чтении данных из внешней шины EXT_BUS через шины IBus и DBus, строки кэш-памяти заполняются последовательно через сами шины IBus и DBus. При этом для необновленных слов в строках кэш-памяти не установлен флаг готовности, и даже при обращении по адресу, совпадающему с активным тегом в кэш, но у необходимого слова не установлен флаг готовности, то обращение будет пропущено в память FLASH или внешнюю шину, и при возврате данных они автоматически запишутся в нужную строку кэш-памяти и установят бит готовности. Таким образом, при последующем обращении по этому адресу ответ будет получен уже от кэш-памяти.

## 9.2 Контроллер ПЗУ (ROM_CNTR)

Контроллер ПЗУ не имеет каких-либо функций по управлению блоком ПЗУ. Данный блок используются всегда, и контроллер предоставляет центральному процессору только информацию об обнаруженных ошибках при работе с ПЗУ, которая отображается в регистрах контроллера ПЗУ.

В микроконтроллере реализовано два контроллера ПЗУ, работающих в режиме LockStep с задержкой в два такта рабочей частоты для проверки корректности исполнения запросов. При возникновении расхождения в работе контроллеров устанавливается флаг события ошибки контроллера ПЗУ. Событие фиксируется в контроллере обработке событий отказов, по нему могут быть настроены прерывание или сброс контроллера. Встроенная загрузочная ПЗУ имеет параллельную организацию ECC-кодирования и имеет размерность 2048 x (32+8ECC).

## 9.3 Контроллер FLASH (FLASH_CNTR)

В контроллер интегрирована блочная энергонезависимая FLASH-память размером:

- 1 Мбайт памяти программ/данных с доступом по шинам AHB или регистровым доступом;
- 256 Кбайт памяти данных с регистровым доступом;
- 256 Кбайт памяти проверочных кодов ECC;
- 48 Кбайт информационной памяти.

Структурно состоит из четырех банков данных по 64K × 32 каждый и двух банков ECC.

Адресация данных при обращении ядра осуществляется последовательно в каждый из банков, то есть данные, соответствующие установленным на шине байтовым адресам с 0x0 по 3, помещаются в нулевой банк, далее с 0x4 по 0x7 - в первый, далее (с 0x8 по 0xB) - во второй, далее (с 0xC по 0xA) - в третий. Таким образом, разряды [1:0] системной шины адреса указывают байт внутри слова и не используются, так как блоки FLASH не поддерживают байтовый доступ. Разряды [3:2] указывают номер банка, к которому производится обращение. Данные биты указывают, данные с какого банка необходимо выдавать на шину, при этом внутри блока обращение производится ко всем четырем банкам одновременно. Биты [5:4] – адрес строки 128 бит, указывают, к какой строке производится обращение.

Для каждых 32-битных слов формируется 8 бит ECC. Для 128-битной строки получается 32 бита ECC, которые размещаются в нулевой банк ECC для четных строк и в первый банк ECC для нечетных. Физически, банки ECC имеют такой же размер, как все банки данных (256 Кбайт), при этом половина каждого из банков не используется для хранения ECC и может быть использована для хранения констант, но поддерживает только регистровый доступ. Данная область доступна на чтение/запись независимо от установленного режима защиты.

Данные со всех четырех банков формируют 128-битную строку, которая вычитывается контроллером одновременно за одну процедуру чтения, что позволяет ускорить доступ к данным со стороны ядра.

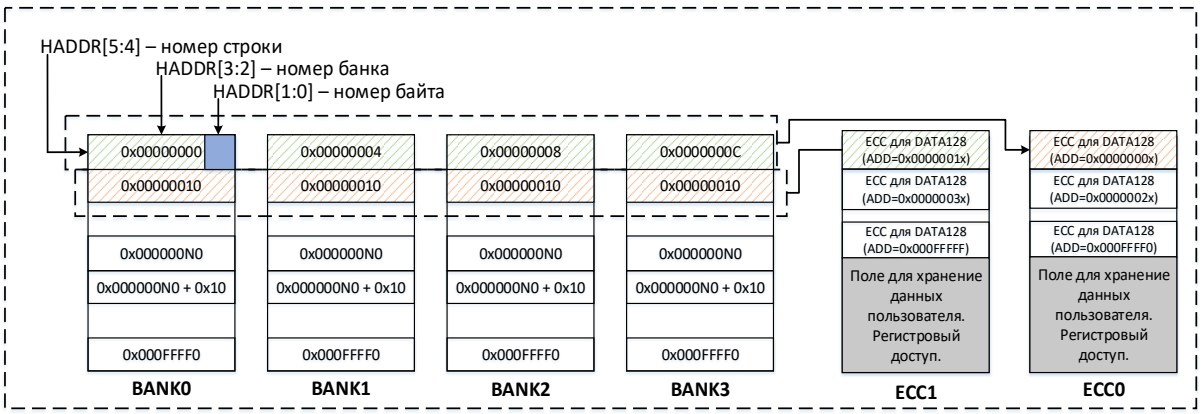



Рисунок 16 – Структурная схема контроллера FLASH

В микроконтроллере реализовано два контроллера FLASH – основной, осуществляющий физический доступ в память, и дублирующий, повторяющий действия основного контроллера с задержкой в два такта системной тактовой частоты. При возникновении расхождения в работе контроллеров устанавливается флаг события ошибки контроллера FLASH. Событие фиксируется в контроллере обработке событий отказов, по нему могут быть настроены прерывание или сброс контроллера.

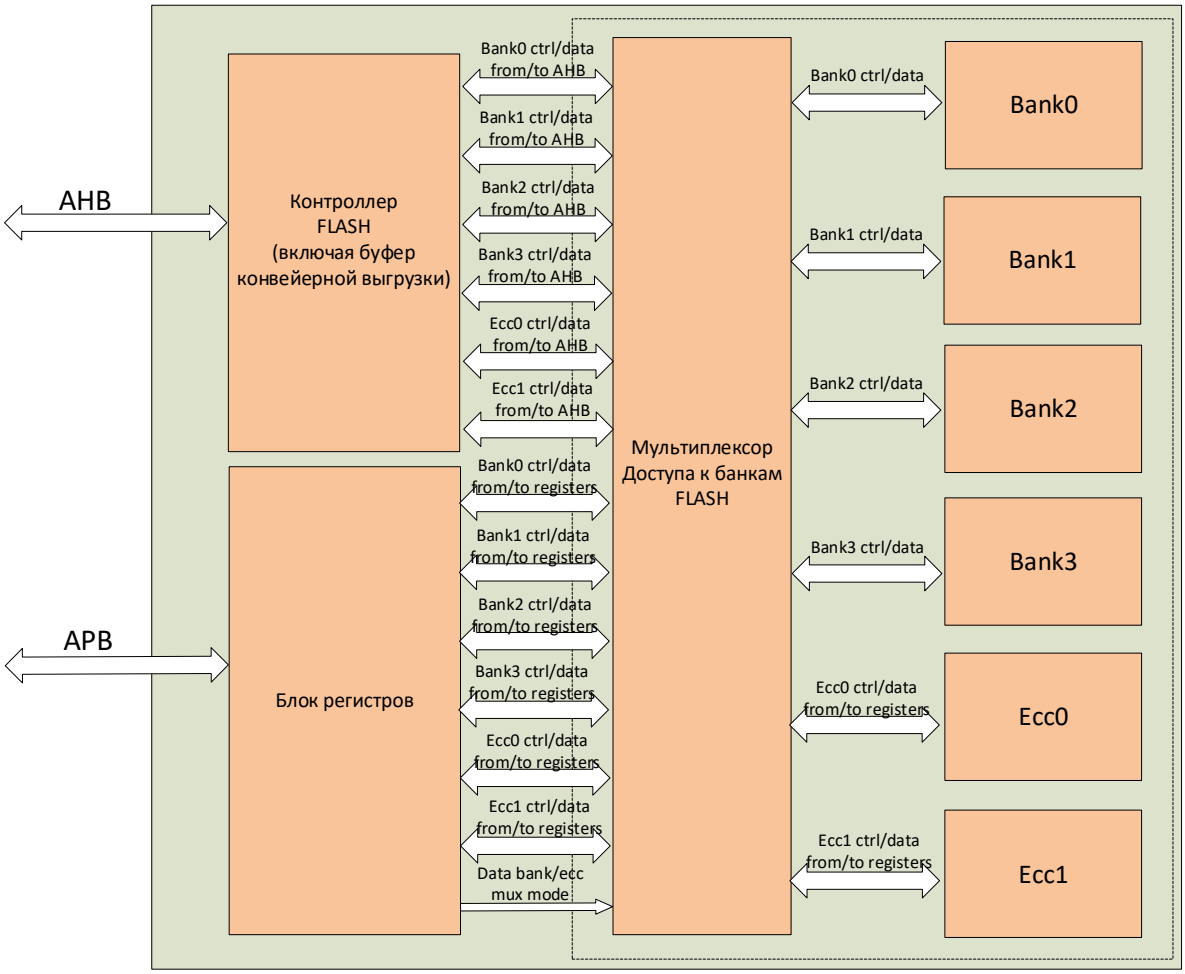



Рисунок 17 – Структурная схема контроллера FLASH

### 9.3.1 *Доступ к памяти FLASH*

#### 9.3.1.1 Доступ по AHB-шине

Базовый тип доступа при выполнении программы из памяти ядром микроконтроллера. При обращении по AHB-шине внутренний автомат доступа автоматически разворачивает диаграмму чтения из всех четырех банков FLASH и соответствующего банка ECC, при этом ядру контроллера будет выдано только запрашиваемое слово, а вся строка будет помещена в буфер для ускорения доступа к последующим адресам. Режим доступа по шине AHB является состоянием по умолчанию, и позволяет микроконтроллеру после сброса выполнять инструкции из FLASH-памяти.

Время доступа к FLASH ($T_{READ}$) зависит от параметров PVT, но не превышает 32 нс. Поле WAITCYCLE регистра CNTR позволяет регулировать время обращения к памяти по шинам AHB в зависимости от рабочей частоты.

Из FLASH извлекаются четыре слова с адресами ADDR+0x0, ADDR+0x4, ADDR+0x8 и ADDR+0xC. Процессору возвращаются считанные по запрошенному по шине адресу данные с ECC. Контроллер FLASH различает обращения по шине I и D и, в зависимости от того, по какой шине произошло обращение, записывает четыре считанных слова соответственно в I-кэш или D-кэш. После завершения обращения от процессора и отсутствии нового, контроллер начинает упреждающее считывание из адреса ADDR+0x10, ADDR+0x14, ADDR+0x18, ADDR+0x1C. При этом если в момент считывания приходит обращение в данные адреса, то чтение продолжается до полного выполнения. Если за время упреждающего чтения обращений со стороны процессора не было, то считанные данные фиксируются во внутреннем буфере контроллера FLASH, и память FLASH переводится в режим пониженного потребления. Если при наличии упреждающе считанных данных приходит обращение по этим адресам, то данные без чтения из FLASH выставляются процессору, и начинается новое упреждающее чтение. В зависимости от частоты процессора и времени чтения из памяти, обращение может занимать от одного до восьми тактов.

При возникновении ECC-ошибок при проверке данных, полученных из памяти, а также при попытке записи в FLASH возникает исключение BUSFAULT.

В режиме доступа по AHB-шине возможно только чтение памяти FLASH. Возможен доступ только к четырем банкам данных и скрытый доступ к ECC для верификации выбранных данных. К информационной памяти и пользовательской области в ECC-банках в данном режиме доступ закрыт.

#### 9.3.1.2 Регистровый доступ

Для осуществления операций стирания, программирования, а также для организации доступа к информационным и скрытым областям памяти и прямого доступа к банкам ECC в микроконтроллере реализован режим регистрового доступа во FLASH.

Для перехода в режим регистрового доступа необходимо установить общий для всех банков бит разрешения регистрового доступа mode и установить биты тех банков, с которыми предполагается регистровый обмен (FLASH5_CR, FLASH4_CR, FLASH3_CR, FLASH2_CR, FLASH1_CR, FLASH0_CR) в регистре FLASH_CNTR. Эти сигналы, объединенные по «И», переводят мультиплексоры сигналов управления банками FLASH на управление из блока регистров.

Сигналы с триггеров IFREN, NVSTR, PROG, MAS1, ERASE, YE, XE, SE[5:0], TMR регистра FLASH_CNTR, регистров ADDR, WDATA3…0 и WECC1..0 передаются напрямую на вход физических блоков FLASH памяти, если выбран соответствующий режим мультиплексоров. Таким образом, регистровое управление блоками FLASH осуществляется программным формированием последовательности сигналов обращения к блокам в соответствии с диаграммами работы. Время доступа определяется программно, значение поля WAITCYCLE в регистровом режиме доступа значения не имеет.

Для записи ячеек FLASH необходимо инициализировать регистр ADDR адресом строки, в которую будет произведена запись в этом цикле. Регистры WDATA 3..0, WECC1..0 инициализировать данными для записи в банки данных и ECC соответственно. Далее установить оставшиеся биты управления в соответствии с диаграммой записи.

Последними всегда устанавливаются сигналы SE, они определяют в какой из банков будет производится обращение в текущем цикле.

Для чтения памяти необходимо сначала также инициализировать регистр ADDR адресом читаемой строки, далее задать управляющие сигналы в соответствии с диаграммой работы FLASH, последними также задаются сигналы SE. Выждав необходимое для осуществления чтения количество тактов (32нс), установить сигнал rdata_ready, по которому будет осуществлена запись в регистры RDATA3…0 и RECC1..0 значений из соответствующих банков памяти, но только в том случае, если для них были установлены сигналы SE.

Операции постраничного удаления и полной очистки банков FLASH выполняются в соответствии с диаграммами работы. Для постраничного стирания памяти необходимо в регистре ADDR указать адрес страницы для осуществления стирания. Полное стирание банков памяти требует лишь задания соответствующих бит регистра FLASH_CNTR, значение регистров адреса и данных на выполнение операции влияния не оказывают.

Бит TMR регистра FLASH_CR рекомендуется держать установленным при проведении любых операций регистрового доступа во FLASH.

Бит IFREN определяет выбор памяти, к которой производится обращение (основная или информационная). Если бит IFREN установлен, все операции проводятся для информационной памяти, и не затрагивают основную, за исключением операции полного стирания. Если требуется стереть только информационную область, то нужно использовать операцию постраничного стирания. Таким образом поле информационной памяти может быть использовано для хранения пользовательской информации. Старшие половины банков ECC никак не задействованы при выполнении программы микропроцессорным ядром и могут также использоваться для энергонезависимого хранения данных.

Корректное выполнение программы микроконтроллерным ядром из FLASH памяти в режиме регистрового доступа невозможно. Перед переводом мультиплексоров управления блоками памяти в режим регистрового доступа необходимо осуществить программный переход в ОЗУ или внешнюю память.

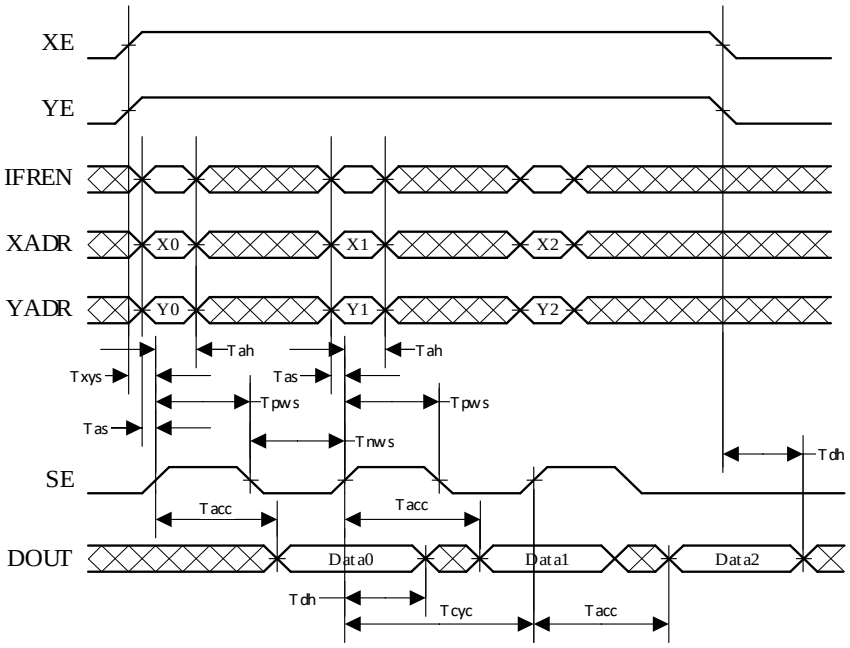



Рисунок 18 – Диаграмма подачи сигналов для осуществления операции чтения

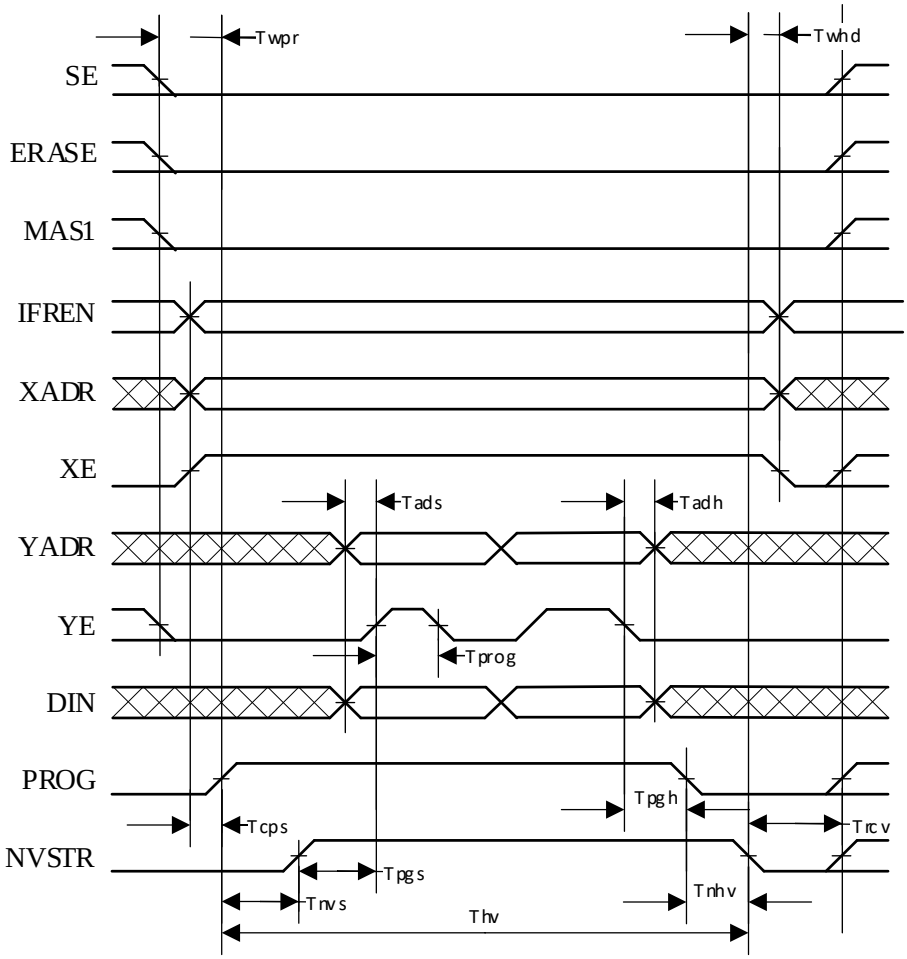



Рисунок 19 – Диаграмма подачи сигналов для осуществления операции программирования

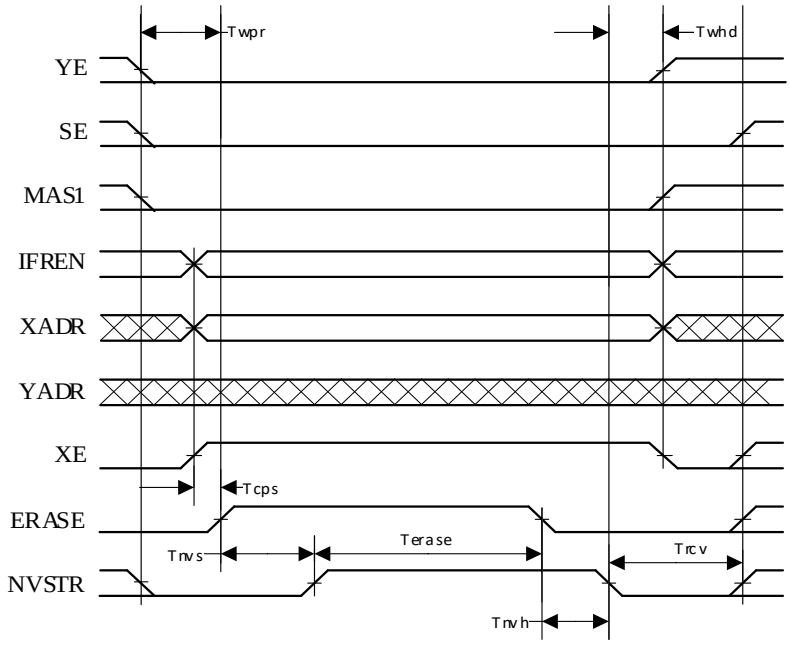



Рисунок 20 – Диаграмма подачи сигналов для осуществления операции постраничного стирания

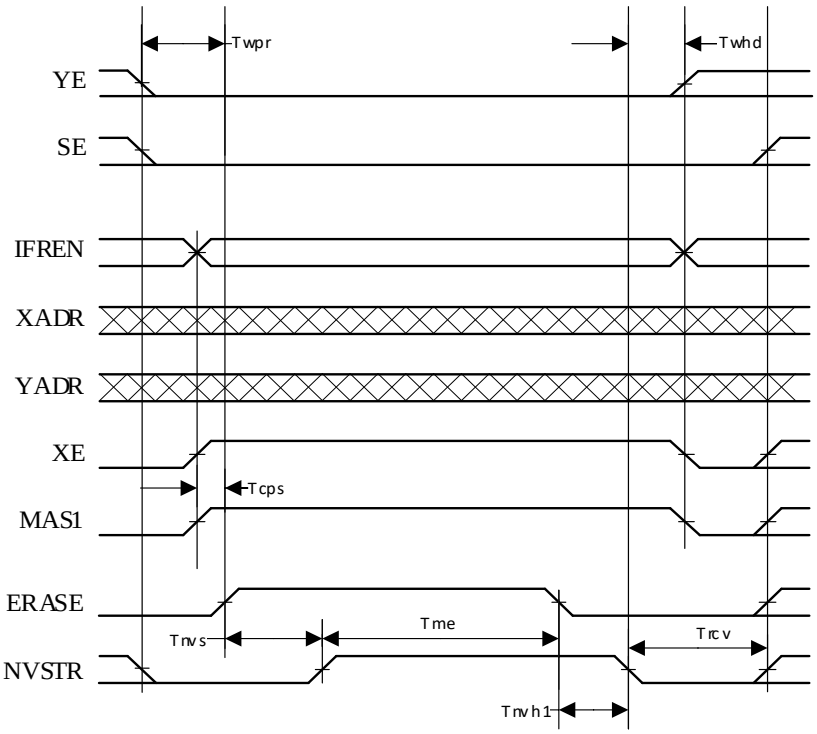



Рисунок 21 – Диаграмма подачи сигналов для осуществления операции полного стирания

| Parameter | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| Access time | Tacc | – | 23 | ns |
| Cycle time | Tcyc | 25 | – | ns |
| Setup time for address to SE | Tas | 1.0000 | – | ns |
| Hold time for address to SE | Tah | 1.0000 | – | ns |
| Data hold time | Tdh | 0.5000 | – | ns |
| Setup time for XE/YE to SE | Txys | 1.0000 | – | ns |
| Positive pulse width of SE | Tpws | 8 | – | ns |
| Negative pulse width of SE | Tnws | 5 | – | ns |
| NVSTR set up time | Tnvs | 5 | – | us |
| NVSTR hold time | Tnvh | 5 | – | us |
| NVSTR hold time (mass erase) | Tnvh1 | 100 | – | us |
| NVSTR to program set up time | Tpgs | 10 | – | us |
| Program hold time | Tpgh | 20 | – | ns |
| Program time | Tprog | 20 | 40 | us |
| Write prepare time | Twpr | >0 | – | ns |
| Write hold time | Twhd | -10 | – | ns |
| Control to PROG/ERASE set up time | Tcps | -10 | – | ns |
| Address/data set up time | Tads | 20 | – | ns |
| Address/data hold time | Tadh | 20 | – | ns |
| Recovery time | Trcv | 10 | – | us |
| Cumulative program HV period | Thv | – | 8 | ms |
| Erase time | Terase | 20 | 40 | ms |
| Mass erase time | Tme | 20 | 40 | ms |
| Make-up time of sleep or power-down to standby | Twk | 5 | – | us |
| Standby hold time | Tsbh | 100 | – | ns |
| Vcc setup time | Tps | 0 | – | ns |
| VD33 hold time | Tph | 0 | – | ns |
| PDM33 hold time | Tpdh | 100 | – | ns |
| PDM33 setup time | Tpds | 100 | – | ns |

### 9.3.2 *Защищенный режим памяти*

Область FLASH памяти, начиная с адреса 0x010E0000, является закрытой областью памяти для хранения ответственных пользовательских данных и кодов программ. Реализованные алгоритмы защиты позволяют обеспечить доступ процессорного ядра к хранящимся в защищенной области данным, при этом обеспечить защиту данных от несанкционированного внешнего доступа.

После сброса микроконтроллера защищенная область является закрытой, в т.ч. и для процессорного ядра. Попытки чтения области по системным шинам приводят к чтению нулевых значений и нулевых кодов ECC, что вызывает событие ECC-ошибки и приводит к BUSFAULT исключению ядра микроконтроллера.

В микроконтроллере реализовано два механизма доступа к защищенному участку памяти:

**Механизм стирания защищенной области памяти.** Запускается механизм установкой бита ERASE_START регистра FLASH_CNTR. Флагом окончания выполнения процедуры является бит ERASE_DONE регистра FLASH_CNTR. В процессе выполнения процедуры выполняется стирание всей памяти FLASH (открытой и защищенной части). После выполнения процедуры открывается полный доступ к памяти на чтение и запись, но все данные, хранившиеся в памяти до начала выполнения процедуры, будут удалены. Отладочные интерфейсы контроллера (JTAG, SWD, TRACE) не блокируются. Доступ к памяти будет открыт до момента следующего сброса контроллера. Запись данных во FLASH не закрывает доступ.

**Механизм получения доступа к защищенной области памяти.** Для выполнения программного кода из FLASH в контроллере реализован механизм защищенного доступа к закрытой области памяти на основе проверки целостности образа кода, помещенного в памяти.

Перед запуском процедуры проверки необходимо записать разрешающий открытие Flash ключ **0xB3C3B3C3** по адресу **0x4000604C** контроллера FLASH. В регистр возможна однократная запись, следующая запись возможна только после сброса микроконтроллера. Любое другое значение ключа блокирует работу автомата доступа к защищенной области.

Старт работы автомата доступа к защищенной области осуществляется записью бита BRKTHRU_START. Далее встроенным в контроллер FLASH автоматом выполняется следующая процедура получения доступа:

1 Автомат получения доступа к защищенной области вычитывает информационные области всех банков FLASH, анализируя при этом значения из Таблица 27.

Таблица 27 – Содержание информационной памяти flash

| | **Адрес** | **Что лежит** |
|---|---|---|
| 1 | 0x0000001C | Адрес ключа 1 KEY1_ADDR |
| 2 | 0x0000002C | Адрес ключа 2 KEY2_ADDR |
| 3 | 0x0000003C | Адрес CRC образа CRC_ADDR |

2 Автомат считывает всю память FLASH, вычисляя CRC для всех получаемых данных, кроме данных, лежащих по адресу CRC_ADDR.

3 Чтение данных происходит сразу из четырех банков, то есть адрес инкрементируется на значение 0x10 каждый раз. Значения адресов ключей и CRC также должны быть выровнены 0x10.

4 По адресу CRC_ADDR производится чтение CRC и сравнение его со значением, рассчитанным для всего оставшегося образа памяти.
CRC хранится во всех четырех банка по адресам:
CRC_ADDR, CRC_ADDR+0x04, CRC_ADDR+0x08, CRC_ADDR+0x0C.

5 По адресам KEY1_ADDR и KEY2_ADDR производится чтение ключей.

6 Анализируются полученные данные. Получение доступа к защищенной области FLASH происходит на основе проверки CRC и кода открытия, хранящегося в ключах.

Если CRC не совпадает, доступ к защищенной области остается заблокированным.

Если CRC совпадает, доступ к защищенной области возможен в трех режимах открытия:

1 В открытом режиме процессорные ядра получают доступ ко всем банкам FLASH на чтение и запись. Отладочные интерфесы не блокируются. Выполнение кода может производится по любому адресу. Режим используется при разработке кода, позволяет производить отладку микрокнтроллера, в т.ч. в случае выполнении программных функций из защищенной области.

2 В защищенном режиме 1 процессорные ядра получают доступ ко всем банкам FLASH на чтение и запись. Программный код является верифицированным на основе CRC проверки, таким образом, гарантируется поведение вычислительных ядер в соответствии с заложенными разработчиков функциями. При этом отладочные интерфесы блокируются, отладка выполнения кода становится невозможной. Выполнение кода по любому адресу (ОЗУ, внешняя память) не приводит к закрытию доступа.

3 Защищенный режим 2 аналогичен защищенному режиму 1, с исключением, что выполнение кода любым из ядер из любой области кроме FLASH приводит к закрытию защищенной области памяти. Данный режим позволяет защитится от некоторых видов атак, в то же время накладывает ограничения на программный код.

Таблица 28 – Режимы доступа во flash

| **CRC** | **Значение KEY1 + KEY2 по модулю** | **Режим доступа** | **Значение breakthrough_mode** |
|---|---|---|---|
| Ok | 0x12345678 | доступ открыт, открытый режим | 2b’11 |
| Ok | 0xABCDEF12 | доступ открыт, защищенный режим 1 | 2b’01 |
| Ok | 0x562C17D4 | доступ открыт, защищенный режим 2 | 2b’10 |
| Ok | другое значение | доступ закрыт | 2b’00 |
| Fail | безразлично | доступ закрыт | 2b’00 |

При breakthrough_close == 1 доступ закрывается.

Повторный доступ возможен только после сброса.

## 9.4 Контроллер ОЗУ RAMC (RAMC_CNTR)

Встроенная статическое ОЗУ RAMC имеет параллельную организацию ECC-кодирования и имеет размерность 2 банка x 16 384 x (32+8ECC) бит. Для ускорения параллельного доступа со стороны микропроцессорных ядер в режиме DUALCORE память разделена на два банка. Доступ к памяти выполняется по шинам инструкций и программных данных I, D каждого из ядер системы, что позволяет с минимальными задержками выполнять программный код, и по шинам DMA.

Для организации доступа в память в микроконтроллере реализовано два контроллера RAMC, которые работают в режиме LockStep для проверки корректности исполнения запросов. В случае возникновения расхождения в работе контроллеров возникает событие LOCKSTEP ERROR, которое отображается в регистрах блока обработки событий отказов и по которому возможна настойка прерываний и сброса микроконтроллера.

Считанные данные из RAMC не кэшируются. При обращении по записи байта и полуслова в память RAMC в контроллере на основании полученных адреса обращения, данных для записи и поля HWECC[7:0] проверяется их корректность. Затем выполняется чтение (32+8ECC) разрядной ячейки расположенной по адресу обращения, выполняется проверка ECC считанных из памяти данных, в корректные считанные данные

«вставляется» записываемый байт или полуслово, выполняется генерация ECC для модифицированного 32-разрядного слова и выполняется его запись по адресу обращения. Таким образом, обращения по записи байт и полуслов выполняется в два этапа.

При обращении по записи 32-разрядного слова в контроллере проверяется ECC записываемого слова, генерируется новое ECC и записывается в память. Таким образом, при обращении по записи 32-разрядного слова выполняются в один этап с проверкой/генерацией ECC.

При возникновении ECC-ошибки в контроллере формируется событие SRAM ECC ERROR или SRAM ECC DOUBLE ERROR. Данные события не приводят к исключению BUSFAULT, отображаются в блоке контроля событий отказов, где по ним может быть настроено прерывание или сброс микроконтроллера.

## 9.5 Контроллер ОЗУ RAMD (RAMD_CNTR)

Организация выполнена аналогично блоку памяти RAMC. Область также, как область RAMC, разбита на два независимых блока памяти. Доступ к памяти осуществляется по шинам данных S каждого из ядер и по шинам блоков прямого доступа к памяти. Выполнение кода из этих областей возможно, но будет происходит медленнее, чем из области RAMC.

## 9.6 Контроллер внешней шины EXTBUS (EXTBUS_CNTR)

Для организации доступа к внешней памяти в микроконтроллере реализован блок контроллера внешней системной шины. Использование блока позволяет обращаться к внешним устройствам памяти напрямую через системные шины IBUS, DBUS, SBUS и шины DMA контроллеров MBUS.

Внешняя шина может иметь 8-, 16- и 32-битную организацию шины данных (без учета проверочных бит ECC при параллельной организации).

Таблица 29 – Выводы интерфейса внешней шины

| Имя входа | Направление | Описание |
|---|---|---|
| ADDR[31:0] | Out | Выход шины адреса |
| DATA[31:0] | In \| Out | Двунаправленная шина данных |
| DATA[47:32] | In \| Out | Двунаправленная шина ECC-данных |
| BEn[5:0] | Out | Выход ByteEnable:<br>0 – разрешение байта<br>1 – запрет байта<br>BEn[0] – DATA[7:0]<br>BEn[1] – DATA[15:8]<br>BEn[2] – DATA[23:16]<br>BEn[3] – DATA[31:24]<br>BEn[4] – DATA[39:32]<br>BEn[5] – DATA[47:40] |
| CS[7:0] | Out | Выход разрешения региона:<br>0 – регион не выбран<br>1 – регион выбран |
| CSn[7:0] | Out | Выход разрешения региона:<br>0 – регион выбран<br>1 – регион не выбран |
| WEn | Out | Сигнал записи:<br>0 – запись<br>1 – нет записи |
| OEn | Out | Сигнал обращения по чтению данных:<br>0 – есть чтение<br>1 – нет чтения |
| BWEn[5:0] | Out | Выход сигнала записи с сигналом Byte Enable:<br>BWEn[0] = WEn or BEn[0]<br>BWEn[1] = WEn or BEn[1]<br>BWEn[2] = WEn or BEn[2]<br>BWEn[3] = WEn or BEn[3] |

| | | |
|---|---|---|
| | | BWEn[4] = WEn or BEn[4]<br>BWEn[5] = WEn or BEn[5] |
| Ready[7:0] | In | Вход готовности от блоков памяти с сигналом готовности:<br>0 – не готов<br>1 – готов |
| CLOCK | Out | Выход сигнала синхронизации внутри транзакции |
| OCLK | Out | Выход тактового сигнала, формируемого контроллером внешней системной шины |

Внешняя шина имеет восемь программно управляемых диапазонов. Для каждого диапазона настраивается собственный режим работы, включая временные характеристики транзакции на шине, разрядность шины, способ обращения с ожиданием или без сигнала готовности, и способ организации ECC. Организация ECC для внешней системной шины

Для всех диапазонов внешней системной шины может быть задан режим работы с проверкой ECC и без проверки.

Диаграммы работы контроллера при записи и чтении представлены на рисунках 22 и 23 соответственно.

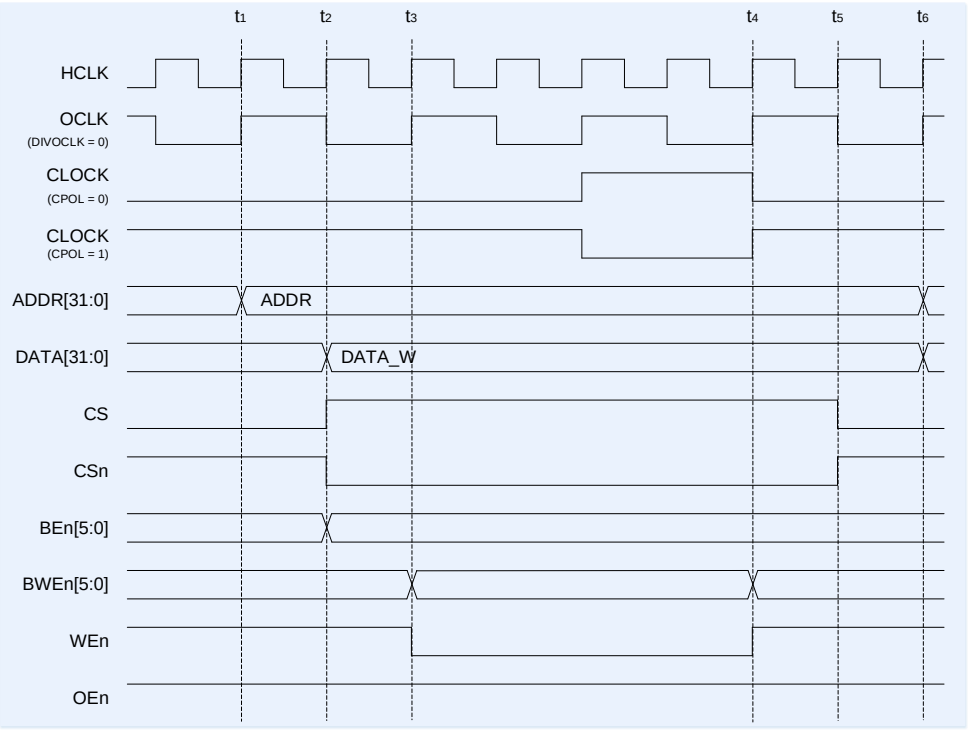



Рисунок 22 – Диаграмма записи на внешней системной шине (WS_SETUP=WS_HOLD=0, WS_ACTIVE = 3)

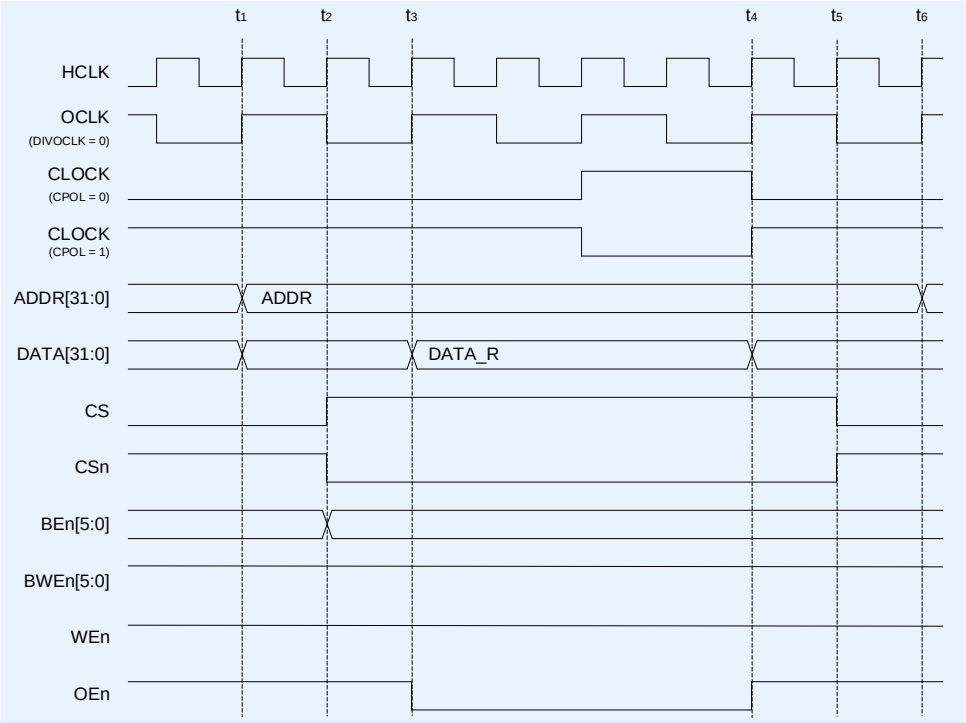



Рисунок 23 – Диаграмма чтения на внешней системной шине
(WS_SETUP=WS_HOLD=0, WS_ACTIVE = 3)

В момент времени $t_1$ начинается транзакция.
$t_2 - t_1 = t_{HCLK}$;
$t_3 - t_2 = t_{HCLK} \cdot (WS_SETUP + 1)$;
$t_4 - t_3 = t_{HCLK} \cdot (WS_ACTIVE + 1)$;
$t_5 - t_4 = t_{HCLK} \cdot (WS_HOLD + 1)$;
$t_6 - t_5 = t_{HCLK}$;
В момент времени $t_6$ может быть начата следующая транзакция.

### 9.6.1 *Помехозащищенное кодирование для шины EXTBUS*

Для всех диапазонов внешней системной шины может быть задан режим работы с проверкой ECC или без проверки. Если проводится проверка ECC, она может быть организована по коду ECC8 (8 проверочных бит на 32 информационных бита) или ECC16 (16 проверочных бит на 32 информационных бита). При этом проверочные биты ECC могут располагаться как в дополнительных разрядах шины данных (32 данные + 8 или 16 ECC – параллельная организация PECC8 или PECC16), так и непосредственно в самой 32-разрядной памяти, начиная с программно задаваемого базового адреса расположения ECC для каждого региона (SECC8 или SECC16). Режим ECC8 для 32-битного слова позволяет обнаруживать и исправлять произвольные одиночные ошибки и обнаруживать любые двойные ошибки. Режим ECC16 для 32-битного слова позволяет обнаруживать и исправлять произвольные одиночные и двойные ошибки и обнаруживать любые тройные ошибки.

### 9.6.2 *Параллельная организация ECC*

Параллельная организация ECC для внешней системной шины возможна только при организации 32-разрядной шины данных. При этом дополнительно к каждым 32-м разрядам данных добавляется 8 или 16 разрядов ECC, то есть общая разрядность шины данных будет 40 или 48 бит. При настройке разрядности шины данных в 16- или 8- битный режим и выбор параллельной организацией ECC обращение к внешней шине будет приводить к ошибке. При записи 8- и 16-разрядных слов во внешнюю системную шину обращение выполняется в два этапа:

1 Выполняется считывание из внешней системной шины по адресу обращения данных с ECC, проверка ECC считанных данных;
2 Модификация 8- или 16-разрядного слова в считанном и проверенном по ECC 32-разрядном слове, генерация ECC для модифицированного слова и запись в память.

При записи 32-разрядного слова за один этап проводится генерация ECC и запись в память. При чтении 8- или 16-разрядных слов из памяти всегда извлекается 32-разрядное слово с ECC. В контроллере внешней шины проверяется ECC, генерируется новое ECC и возвращается процессору, на входе которого ECC проверяется еще раз.

### 9.6.3 *Последовательная организация ECC*

Последовательная организация ECC для внешней системной шины возможна для всех разрядностей шины данных, но при этом контроллер внешней системной шины все равно манипулирует 32-разрядными словами. Таким образом, независимо от реализации шины данных памяти на внешней системной шине для процессора отображается 32-разрядная память.

Контрольные ECC-биты для памяти располагаются в старших адресах каждого диапазона, начиная с программно задаваемого базового адреса ECCBASE (младшие 4 бита адреса ECCBASE игнорируются и считаются равными нулю). Для 8-, 16- и 32-битных шин данных принцип расположения данных представлен на рисунках 24 и 25.

Такая организация памяти требует:

- два обращения для считывания 32-разрядных данных с ECC при 32-разрядной шине данных;
- три обращения при 16-разрядной шине данных;
- пять обращений при 8-разрядной шине данных при SECC8;
- шесть обращений при 8-разрядной шине данных при SECC16.

Число обращений для выполнения операций чтения и записи данных различной разрядности при различной организации шины данных представлено в таблице 30.

### 9.6.4 *Адреса расположения кодов ECC*

При параллельной организации, ECC-биты располагаются в расширении шины данных и считываются одновременно со словом.

При последовательной организации, ECC-биты хранятся в том же массиве памяти, что и проверяемые слова, но начиная с адреса ECCBASE. И для их считывания выполняются дополнительные циклы чтения или записи.

При последовательной организации адрес ECC-бит вычисляется согласно формулам:

- Для 8, 16, 32-битной шине данных ECC8:
  $ADDR_{ECC}[31:0] = ECCBASE[31:0] + OFFSET(ADDR[31:2])$;
- Для 8, 16, 32-битной шине данных ECC16:
  $ADDR_{ECC}[31:0] = ECCBASE[31:0] + OFFSET(ADDR[31:2])<<1$,

где OFFSET(A) = A – REGIONBASE(A);

ECCBASE для каждого семейства сегмента должен лежать внутри адресного пространства.

Таблица 30 – Число транзакций на шине EXT_BUS при последовательном ECC

| Тип операции | Разрядность данных | Разрядность шины | Число транзакций на шине |
|---|---|---|---|
| чтение | 32,16,8 | 32 | 2 чтения |
| чтение | 32,16,8 | 16 | 3 чтения |
| чтение | 32,16,8 | 8 | 5 чтений |
| запись | 32 | 32 | 1 чтение+2 записи |
| запись | 32,16,8 | 16 | 3 чтения+3 записи |
| запись | 32,16,8 | 8 | 5 чтений+5 записей |

| | | | |
|---|---|---|---|
| запись | 16,8 | 32 | 2 чтения+2 записи |

Область выше ECCBASE, предназначенная для хранения ECC, может быть считана/записана как обычные данные. Для обращения в эту область необходимо предварительно перейти в режим работы без контроля ECC.

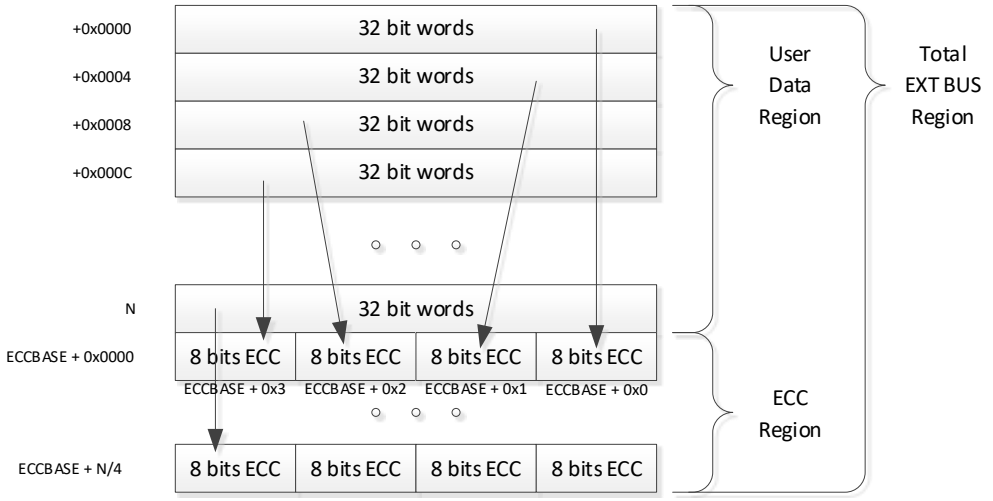



Рисунок 24 – Структура расположения данных в памяти на шине EXTBUS для 32-, 16- и 8-битных шин SECC8

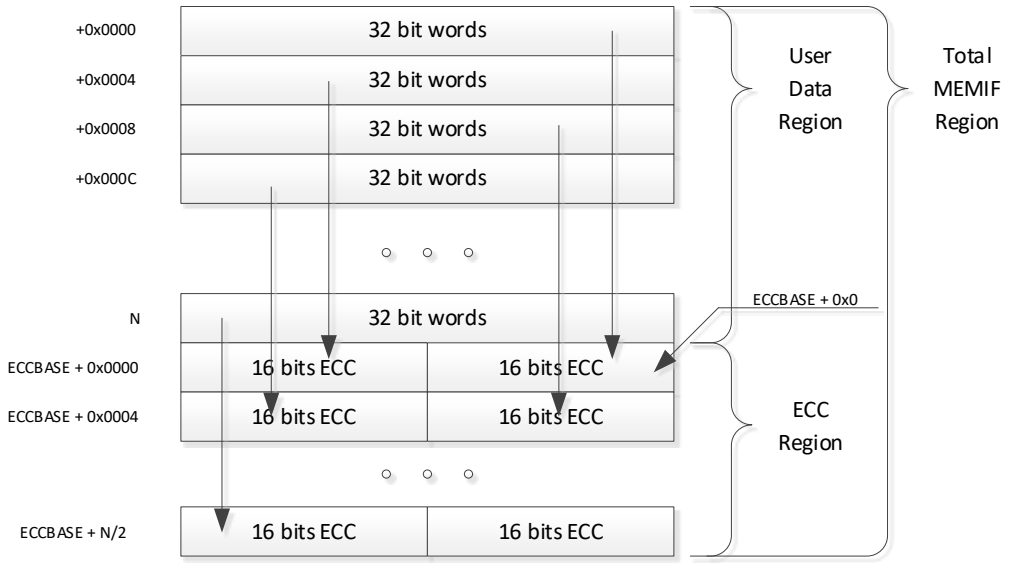



Рисунок 25 – Структура расположения данных в памяти на шине EXTBUS для 32-, 16- и 8-битных шин SECC16

### 9.6.5 *Доступ во внешнюю память без контроля ECC*

При отсутствии возможности или необходимости организации контроля ECC для диапазона внешней памяти может быть установлен режим работы без контроля ECC для определенного диапазона. В этом случае данные считываются из внешней памяти без контроля ECC. При необходимости выполняются дополнительные циклы чтения для формирования 32-разрядных данных, что необходимо учитывать при работе с внешними периферийными блоками, критичными к чтению (FIFO, и т. п.).

Таблица 31 – Количество транзакций при доступе во внешнюю память без использования ECC-защиты

| Тип операции | Разрядность данных | Разрядность шины | Число транзакций на шине |
|---|---|---|---|
| Чтение | 32,16,8 | 32 | 1 чтения |
| Чтение | 32 | 16 | 2 чтения |

| Тип операции | Разрядность данных | Разрядность шины | Число транзакций на шине |
|---|---|---|---|
| Чтение | 32 | 8 | 4 чтения |
| Чтение | 16,8 | 16 | 1 чтение |
| Чтение | 16 | 8 | 2 чтения |
| Чтение | 8 | 8 | 1 чтение |
| Запись | 32 | 32 | 1 запись |
| Запись | 32 | 16 | 2 записи |
| Запись | 32 | 8 | 4 записи |
| Запись | 16,8 | 32 | 1 чтение + 1 запись |
| Запись | 16 | 16 | 1 запись |
| Запись | 16 | 8 | 2 записи |
| Запись | 8 | 16 | 1 чтение +1 запись |
| Запись | 8 | 8 | 1 запись |

### 9.6.6 *Регионы внешней памяти*

С помощью контроллера внешней шины EXTBUS можно организовать до 8-ми регионов с подключением внешних устройств с отображением по адресам, приведенным в таблице 32. Базовые адреса регионов фиксированы и не могут изменяться.

Для каждого региона может быть задан свой режим работы. При отображении выводов контроллера внешней шины на пользовательские выводы этим выводам необходимо задать функцию внешней шины. Если выводу не задана функция внешней шины, с него считывается ноль. Если выводы внешней шины отображаются на выделенные выводы для многокристальной сборки, они выполняют только эту функцию, и их настройка выполняется в регистрах контроллера внешней шины. Для различных регионов могут быть выбраны различные способы отображения – на пользовательские выводы, либо на выводы для микросборки.

Таблица 32 – Базовые адреса внешней системной шины

| Регион | Начальный адрес | Конечный адрес |
|---|---|---|
| REGION0 | 0x1000_0000 | 0x17FF_FFFF |
| REGION1 | 0x1800_0000 | 0x1FFF_FFFF |
| REGION2 | 0x5000_0000 | 0x57FF_FFFF |
| REGION3 | 0x5800_0000 | 0x5FFF_FFFF |
| REGION4 | 0x6000_0000 | 0x67FF_FFFF |
| REGION5 | 0x6800_0000 | 0x6FFF_FFFF |
| REGION6 | 0x7000_0000 | 0x7FFF_FFFF |
| REGION7 | 0x8000_0000 | 0xDFFF_FFFF |

### 9.6.7 *Организация доступа с ожиданием сигнала готовности*

Для каждого диапазона внешней системной шины может быть настроен режим работы с ожиданием сигнала готовности READY. В этом случае транзакция на внешней системной шине может быть начата только при пассивном (шина готова) состоянии сигнала READY. При этом будут выполнены фазы предустановки сигналов выборки, адреса/данных перед сигналами записи/чтения и вход в фазу записи/чтения. Переход из фазы записи/чтения в фазу удержания адреса/данных будет выполнен только при наличии пассивного состояния сигнала READY. При наличии активного состояния сигнала READY (шина не готова) контроллер внешней системной шины будет находиться в фазе записи/чтения. В режиме работы с ожиданием сигнала готовности в любом случае для всех фаз выполняются временные настройки их длительности, сигналом готовности может быть только увеличена фаза записи/чтения. Для всех диапазонов, работающих с ожиданием внешнего сигнала, может быть задана только одна полярность пассивного/активного состояния. Для каждого диапазона, работающего с ожиданием внешнего сигнала готовности, может быть программно задана максимальная длительность времени ожидания сигнала готовности (перед началом транзакции и нахождения в фазе

записи/чтения). Если транзакция не выполняется успешно (без учета корректности ECC) за максимальное время ожидания, транзакция со стороны процессора завершается с флагом неготовности (исключение BusFault). При работе внешней системной шины в режиме без ожидания сигнала готовности сигнал READY всегда считается находящимся в пассивном состоянии.

## 9.7 Контроллер диагностики памяти SCRUBBER (SCR_CNTR)

В микросхеме реализован специальный блок SCRUBBER. Данный блок позволяет без участия процессора проводить первоначальную инициализацию памяти и проводить диагностику памяти в блоках FLASH, RAMC, RAMD и EXTBUS на предмет возникновения в них ошибок. При обращении к FLASH-памяти доступно только чтение, запись не доступна. Блок SCRUBBER может работать в следующих режимах:

- первоначальная инициализация памяти SCR_MODE = 0x1;
- проверка инициализации памяти SCR_MODE = 0x2;
- прямое чтение памяти SCR_MODE = 0x3;
- прямая запись памяти SCR_MODE = 0x4;
- последовательное циклическое чтение SCR_MODE = 0x5;
- последовательное циклическое неразрывное чтение-запись SCR_MODE = 0x6.

В режиме ***первоначальной инициализации***, блоку SCRUBBER задается начальный адрес S_ADDR и конечный адрес F_ADDR инициализируемого массива памяти, данные DATA для первоначальной инициализации (одно слово на весть массив) и задается режим первоначальной инициализации SCR_MODE = 0x1. Блок SCRUBBER, без участия процессора и в фоновом режиме (обращения со стороны процессора или DMA более приоритетны), осуществляет запись данных DATA и проверочные ECC-биты, вычисляемые в соответствии с записываемыми данными и адресом записи, во все ячейки, начиная с адреса S_ADDR и заканчивая адресом F_ADDR. По завершению операции генерируется событие окончания операции SCR_FINISH_IF.

В режиме ***проверки первоначальной инициализации***, блоку SCRUBBER задается начальный адрес ***S_ADDR*** и конечный адрес ***F_ADDR*** массива памяти, данные DATA для проверки первоначальной инициализации (одно слово на весть массив) и задается режим проверки первоначальной инициализации ***SCR_MODE=0x2***. Блок SCRUBBER, без участия процессора и в фоновом режиме (обращения со стороны процессора или DMA более приоритетны), осуществляет чтения данных из памяти и сравнивает их с данными в ***DATA,*** начиная с адреса ***S_ADDR*** и заканчивая адресом ***F_ADDR***. Если при чтении обнаруживается одиночная или двойная ошибка, в контроллере проверяемого диапазона в регистр ***ECCADDR*** записывается адрес ячейки с ошибкой, в регистр ***ECCDATA*** данные без исправления ECC из ячейки с ошибкой, в регистр ***ECCECC*** - проверочные биты ECC без исправления, и генерируется событие SECC или DECC. В контроллере SCRUBBER тест останавливается и генерируется событие SCR_ERROR_IF. По завершению выполнения операции без ошибок блок SCRUBBER генерирует событие окончания операции SCR_FINISH_IF.

В режиме ***прямого чтения памяти*** блоку SCRUBBER задается начальный адрес ***S_ADDR*** и конечный адрес ***F_ADDR*** массива памяти и задается режим прямого чтения ***SCR_MODE=0x3***. При чтении из регистра ***DATA*** считываются данные с проверкой ECC из памяти, из ячейки с адресом ***S_ADDR***. При чтении регистра ***DATA*** происходит автоматическое увеличение значения регистра ***S_ADDR*** на 0x04. Если при чтении обнаруживается одиночная или двойная ошибка, в контроллере проверяемого диапазона, в регистр ***ECCADDR*** записывается адрес ячейки с ошибкой, в регистр ***ECCDATA*** данные без исправления ECC из ячейки с ошибкой, в регистр ***ECCECC*** - проверочные биты ECC без исправления, и генерируется событие SECC или DECC. В контроллере SCRUBBER тест останавливается и генерируется событие SCR_ERROR_IF. После считывания регистра ***DATA***, при ***S_ADDR*** равном ***F_ADDR***, генерируется событие окончания операции SCR_FINISH_IF - но выполнение операции при этом не завершается, возможно дальнейшее чтение.

В режиме ***прямой записи памяти*** блоку SCRUBBER задается начальный адрес ***S_ADDR*** и конечный адрес ***F_ADDR*** массива памяти и задается режим прямой записи ***SCR_MODE=0x4***. При записи регистра ***DATA*** в память записываются данные из регистра ***DATA*** и проверочные ECC-биты из регистра ***ECC***. В память записываются непосредственно те значения, которые записаны в регистры. Таким образом, корректность ECC-битов может быть нарушена. При записи в регистр ***DATA*** происходит автоматическое увеличение значения регистра ***S_ADDR*** на 0x04. После записи в регистр ***DATA***, при ***S_ADDR*** равном ***F_ADDR*** генерируется событие окончания операции SCR_FINISH_IF - но выполнение операции при этом не завершается, возможна дальнейшая запись. Данный режим позволяет проверить битовую целостность массива памяти, а также может быть применен для генерации событий одиночных и двойных сбоев при отладке программного обеспечения. Так как факт записи в память определяется моментом записи в регистр ***DATA***, то регистр ***ECC*** должен быть задан до записи регистра ***DATA***.

В режиме ***циклического чтения***, блоку SCRUBBER задается начальный адрес ***S_ADDR*** и конечный адрес ***F_ADDR*** массива памяти, интенсивность чтения в поле ***SCR_TIME*** и задается режим циклического чтения ***SCR_MODE=0x5***. Блок SCRUBBER, без участия процессора и в фоновом режиме (обращения со стороны процессора или DMA более приоритетны), осуществляет чтение данных из памяти, начиная с адреса ***S_ADDR*** и заканчивая адресом ***F_ADDR,*** с периодом определяемым полем ***SCR_TIME***. Если при чтении обнаруживается одиночная или двойная ошибка, в контроллере проверяемого диапазона, в регистр ***ECCADDR*** записывается адрес ячейки с ошибкой, в регистр ***ECCDATA*** – данные без исправления ECC из ячейки с ошибкой, в регистр ***ECCECC*** – проверочные биты ECC без исправления, и генерируется событие SECC или DECC. В контроллере SCRUBBER тест останавливается и генерируется событие SCR_ERROR_IF. По завершению выполнения операции без ошибок блок SCRUBBER генерирует событие окончания операции SCR_FINISH_IF. Если при этом установлен бит ***SCR_CYCLE***, то циклическое чтение начинается снова.

В режиме ***циклического связанного чтения-записи***, блоку SCRUBBER задается начальный адрес ***S_ADDR*** и конечный адрес ***F_ADDR*** массива памяти, интенсивность чтения в поле ***SCR_TIME*** и задается режим циклического связанного чтения-записи ***SCR_MODE=0x6***. Блок SCRUBBER, без участия процессора и в фоновом режиме (обращения со стороны процессора или DMA более приоритетны), осуществляет чтения данных из памяти и запись их обратно, начиная с адреса ***S_ADDR*** и заканчивая адресом ***F_ADDR*** с периодом определяемым полем ***SCR_TIME***. Если в промежутке между чтением и записью возникло более приоритетное обращение со стороны процессора или DMA, то чтение повторяется для того чтобы избежать возможную перезапись новых данных старым значением. Если при чтении обнаруживается одиночная или двойная ошибка, в контроллере проверяемого диапазона в регистр ***ECCADDR*** записывается адрес ячейки с ошибкой, в регистр ***ECCDATA*** - данные без исправления ECC из ячейки с ошибкой, в регистр ***ECCECC*** - проверочные биты ECC без исправления, и генерируется событие SECC или DECC. В контроллере SCRUBBER тест останавливается и генерируется событие SCR_ERROR_IF. По завершению выполнения операции без ошибок блок SCRUBBER генерирует событие окончания операции SCR_FINISH_IF. Если при чтении обнаруживается одиночная ошибка и бит ***SCR_SEC*** не установлен, или обнаруживается двойная ошибка, то обратно в память запись не осуществляется. В регистр ***ECCADDR*** записывается адрес ячейки с ошибкой, в регистр ***ECCDATA*** - данные без исправления ECC из ячейки с ошибкой, в регистр ***ECCECC*** - проверочные биты ECC без исправления, и генерируется событие SCR_ERROR_IF, и тест останавливается. Если при проверке ошибок не обнаружено, то по завершению операции генерируется событие окончания операции SCR_FINISH_IF. Если при этом установлен бит ***SCR_CYCLE*** равным единице, то циклическое чтение начинается снова. Данный режим позволяет регенерировать только полностью корректное состояние ячеек памяти при сброшенном в ноль бите ***SCR_SEC***, либо выполнять автоматическое исправление одиночных ошибок при установленном бите ***SCR_SEC***.

Блок SCRUBBER начинает выполнение операции при записи в регистр ***CNTRL*** соответствующего требуемой операции кода ***SCR_MODE***.

До записи кода операции следует сбросить флаги событий окончания операции SCR_FINISH_IF и обнаружения ошибки SCR_ERROR_IF. Переключение между режимами рекомендуется осуществлять через SCR_MODE=0x0. При задании режима установленное значение S_ADDR должно быть меньше F_ADDR. Блок SCRUBBER сбрасывается только сигналом сброса по питанию.

### 9.8 Организация доступа к памяти контроллера DMA (DMA_CNTR)

Контроллер прямого доступа в память (DMA) является инициатором транзакций на шине AHB, аналогичный процессорному ядру. Контроллер DMA формирует шину MBUS. На входе/выходе контроллера DMA генерируется и проверяется ECC, аналогично шинам IBUS, DBUS и SBUS процессорного ядра. При работе с другими блоками системы по чтению и записи данных, DMA контроллер функционирует аналогично процессорному ядру. Настройка и управление контроллера DMA осуществляется в регистрах периферии. Описание функционирования контроллера DMA приведено в разделе «Контроллер прямого доступа в память»

### 9.9 Организация доступа к памяти контроллера Ethernet (ETHERNETMAC_CNTR)

Встроенная память контроллера Ethernet MAC отображается в адресное пространство на уровне арбитража шины AHB, при этом регистры управление контроллером Ethernet отображены в адресное пространство периферии. Отображаемая память Ethernet MAC снабжена защитой ECC аналогично памяти RAMC и RAMD. Передаваемые и принимаемые данные по сети Ethernet защищены контрольными суммами CRC согласно протоколу стандарта Ethernet. При приеме данных по сети Ethernet, данные формируются в 32-разрядные слова, для которых генерируется ECC и записываются в память. При считывании со стороны процессора проверка ECC производится на входе в процессор. При записи со стороны процессора данные в память записываются с проверкой ECC и генерацией новой ECC, аналогично работе памяти RAMC и RAMD.

### 9.10 Организация доступа к памяти периферии

При работе с периферией на шине APB осуществляется проверка ECC, но не все периферийные блоки ее поддерживают в полном объеме. Проверка ECC проводится для блоков с большой внутренней памятью, таких как CAN, UART, SSP, EMAC. Другие блоки при обращении формируют флаг NOECC, таким образом, в мосте AHB2APB не проводится проверка ECC при чтении, а для процессора формируется ECC по тем данным, которые считаны.

# 10 Контроллер прямого доступа к памяти (DMA_CNTR)

## 10.1 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.

## 10.2 Термины и определения

При описании контроллера DMA используются следующие термины:

Таблица 33 – Термины и определения

| Термины/определения | Описание |
|---|---|
| Альтернативная | Альтернативная структура управляющих данных канала.<br>Для изменения типа структуры данных можно установить соответствующий регистр |
| C | Идентификатор номера канала прямого доступа. Например,<br>C=1 – канал DMA 1<br>C=23 – канал DMA 23 |
| Канал | Возможны конфигурации контроллера с числом каналов до 32.<br>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. Можно программировать структуру данных так, чтобы контроллер выполнял передачу данных по каналу DMA в желаемом режиме. Контроллер должен иметь доступ к области системной памяти, где находится эта информация.<br>*Примечание* – Любое упоминание в данном разделе 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.<br>Общее количество передач DMA, которые контроллер выполняет для канала |

| Термины/определения | Описание |
|---|---|
| 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, устанавливающая 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 окончания цикла, в случае, 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. Затем контроллер начинает цикл DMA, используя альтернативную структуру данных.<br>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 |

## 10.3 Функциональное описание

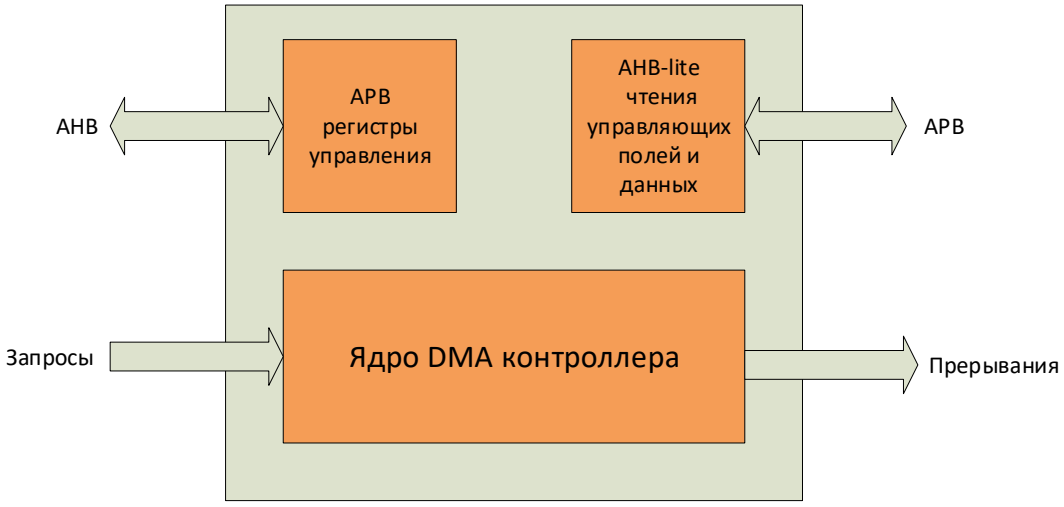



Рисунок 26 – Структурная схема контроллера DMA

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

### 10.3.1 *Типы передач*

Контроллер интерфейса не поддерживает пакетные передачи, выполняет только 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.

### 10.3.2 *Разрядность передач данных*

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

Таблица 34 – 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;<br>**) – запрещенная комбинация | | | |

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

### 10.3.3 *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 35 – 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 (подробнее см. в подразделе «Структура управляющих данных канала DMA»).

Для каждого канала DMA также возможен выбор режима защиты данных (подробнее см. в подразделе «Управление DMA»).

### 10.3.4 *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. Таблица 36 перечисляет возможные комбинации.

Таблица 36 – Величина инкремента адреса в зависимости от разрядности данных

| 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 подразделе «Структура управляющих данных канала DMA». 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. подраздел «Структура управляющих данных канала DMA»).

## 10.4 Управление DMA

### 10.4.1 *Правила обмена данными*

Контроллер использует правила обмена данными, перечисленные в таблице 37, 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 37 – 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 |

| Правило | Описание |
|---|---|
| 4 | Для типов циклов DMA, отличных от периферийного «Исполнение с изменением конфигурации», dma_active[C] остается в состоянии 1 до тех пор, пока контроллер не окончит передачи с номерами меньше, чем значение $2^R$ или чем число передач, указанное в регистре n_minus_1.<br>В периферийном режиме «Исполнение с изменением конфигурации», dma_active[C] остается в состоянии 1 в течение каждой пары DMA передач, с использованием первичной и альтернативной структур управляющих данных. Таким образом, контроллер выполняет $2^R$ передач, используя первичную структуру управляющих данных, затем без осуществления арбитража выполняет передачи с номерами меньше, чем значение $2^R$ (или чем число передач, указанное в регистре n_minus_1), используя альтернативную структуру управляющих данных. По окончании последней передачи dma_active[C] сбрасывается в 0 |
| 5 | Контроллер устанавливает dma_active[C] в 0 на, как минимум, один такт сигнала hclk перед тем, как снова установит dma_active[C] или dma_active[ ] в 1 |
| 6 | Для каналов, по которым разрешена передача, контроллер осуществляет установку в 1 только одного dma_done[ ] |
| 7 | Если dma_req[C] устанавливается в состояние 1 в момент, когда dma_active[C] или dma_stall также в состоянии 1, то это означает, что контроллер обнаружил запрос |
| 8 | Если разряды cycle_cntr 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_cntr позволяют удержание dma_done[C], то по срезу сигнала dma_active[ ] произойдут события:<br>– если dma_stall в состоянии 0, контроллер устанавливает dma_done[ ] в состояние 1 продолжительностью один такт hclk;<br>– 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:<br>– контроллер завершит $2^R$ передач (или число передач, указанное в регистре n_minus_1);<br>– dma_req[C] будет установлен в 0;<br>– 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:<br>контроллер установит dma_active[C] в 0 на следующем такте сигнала hclk;<br>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 $2^R$.<br>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.<br>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.<br>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 2R, то контроллер не очистит разряды chnl_useburst_set, и запросы по dma_sreq[C] будут маскированы. Если периферия не устанавливает dma_req[C] в состояние 1, то контроллер никогда не выполнит необходимых передач.

Таблица 38 – Правила, при которых передача данных по каналам разрешена, и запросы не маскируются

| 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 |

### 10.4.2 *Диаграммы работы контроллера DMA*

Данный раздел описывает примеры функционирования контроллера с использованием правил обмена данными, представленных в таблице 37.

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

Примечание – Все диаграммы, показанные далее на рисунках в этом подразделе, подразумевают следующее:

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

#### 10.4.2.1 Импульсный запрос на обработку

Рисунок 27 показывает временную диаграмму работы контроллера DMA при получении импульсного запроса от периферии.

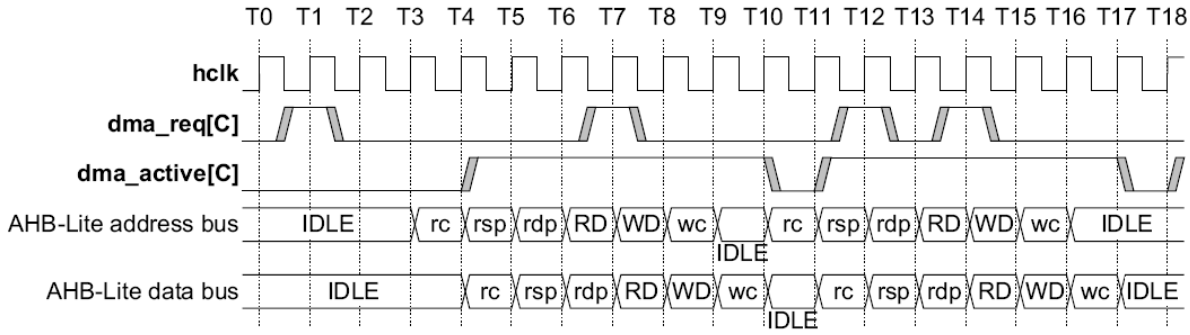



Рисунок 27 – Диаграмма работы при получении импульсного запроса

Пояснения к диаграмме на рисунке 27 приведены ниже.

Таблица 39 – Пояснения к диаграмме работы при получении импульсного запроса

| | |
|---|---|
| T1 | Контроллер обнаружил запрос на обработку по каналу C (Таблица 37, 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:<br>rc – чтение настроек канала, channel_cfg;<br>rsp – чтение указателя адреса окончания данных источника, src_data_end_ptr;<br>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:<br>RD – чтение данных;<br>WD – запись данных |
| T9-T10 | Контроллер осуществляет запись настроек канала, channel_cfg, где<br>wc – запись настроек канала, channel_cfg |
| T10 | Контроллер сбрасывает сигнал dma_active[C], что указывает на окончание передачи DMA (см. правило 4) |
| T10-T11 | Контроллер удерживает dma_active[C] на, как минимум, один такт hclk (см. правило 5) |
| T11 | Если канал C имеет более высокий приоритет, то контроллер устанавливает dma_active[C], так как ранее на такте T7 был получен запрос на обработку (см. правила 2 и 3) |
| T12 | При установленном dma_active[C] в 1 и при условии, что chnl_req_mask_set[C] находится в состоянии 0, контроллер обнаруживает импульс запроса на обработки по каналу C (см. правило 7). Контроллер обработает этот запрос в течение следующего арбитража |
| T14 | Контроллер игнорирует запрос по каналу C из-за отложенного запроса, полученного на такте T12 |
| T17 | Контроллер сбрасывает сигнал dma_active[C], что указывает на окончание передачи DMA (см. правило 4) |
| T17-T18 | Контроллер удерживает dma_active[C], как минимум, на один такт hclk (см. правило 5) |
| T18 | Если канал C имеет более высокий приоритет, то контроллер устанавливает dma_active[C], так как ранее на такте T12 был получен запрос на обработку (см. правила 2 и 3) |

#### 10.4.2.2 Запрос на обработку по уровню

Временная диаграмма работы контроллера DMA при получении от периферии запроса на обработку по уровню.

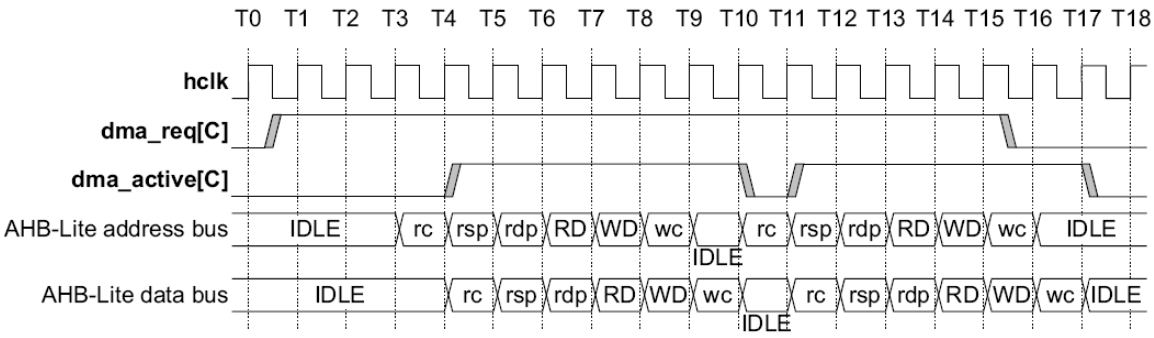



Рисунок 28 – Диаграмма работы при получении запроса на обработку по уровню

Таблица 40 – Пояснения к диаграмме работы при получении запроса на обработку по уровню

| | |
|---|---|
| T1 | Контроллер обнаружил запрос на обработку по каналу C (Таблица 37, 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:<br>rc – чтение настроек канала, channel_cfg;<br>rsp – чтение указателя адреса окончания данных источника, src_data_end_ptr;<br>rdp - чтение указателя адреса окончания данных приемника, dst_data_end_ptr |
| T7-T9 | Контроллер выполняет передачу DMA по каналу C, где:<br>RD – чтение данных<br>WD – запись данных |
| T9-T10 | Контроллер осуществляет запись настроек канала, channel_cfg, где<br>wc – запись настроек канала, channel_cfg |
| T10 | Контроллер сбрасывает сигнал dma_active[C], что указывает на окончание передачи DMA (см. правило 4).<br>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.

#### 10.4.2.3 Флаги завершения

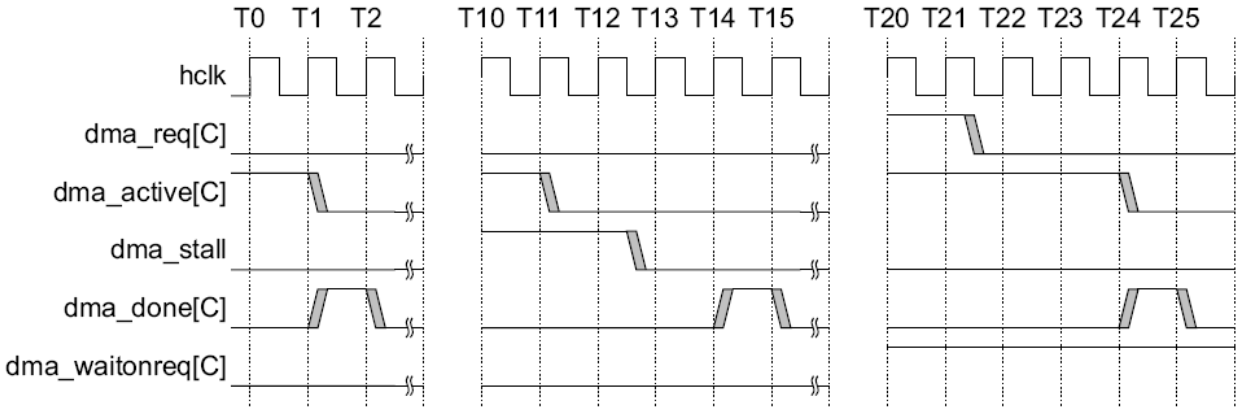



Рисунок 29 – Диаграммы функционирования dma_done

Рисунок 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.

Таблица 41 – Функционирования dma_done, такты от T0 до T2

| | |
|---|---|
| T1 | Контроллер сбрасывает сигнал dma_active[C], что указывает на окончание передачи DMA (Таблица 37, правило 4) |
| T1-T2 | Контроллер завершает цикл DMA и если cycle_cntr[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) |

Таблица 42 – 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_cntr[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).

Таблица 43 – 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_cntr[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) |

#### 10.4.2.4 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.

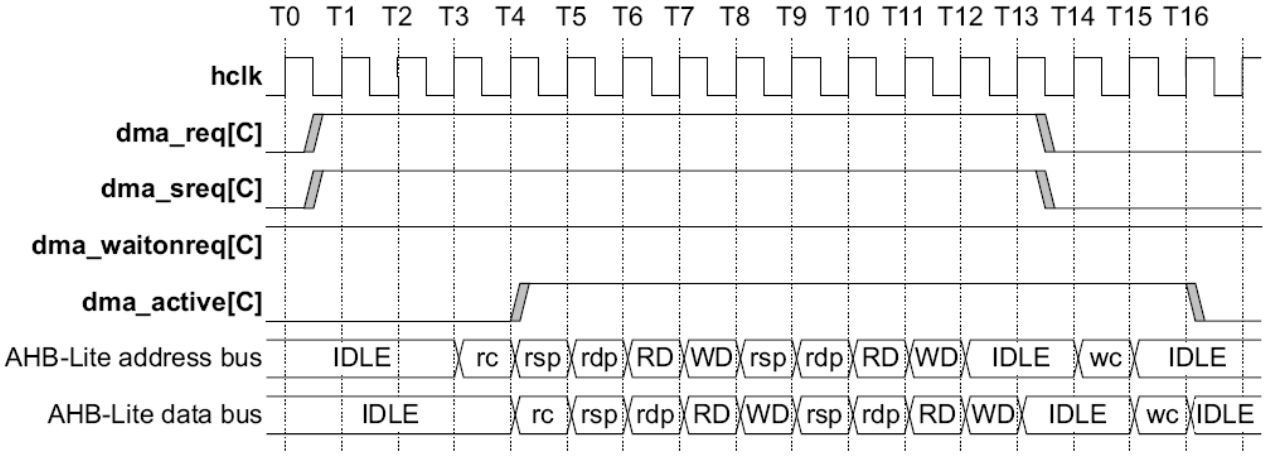



Рисунок 30 – Диаграмма работы контроллера DMA при использовании dma_waitonreq

Таблица 44 – 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.<br>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:<br>rc – чтение настроек канала, channel_cfg;<br>rsp – чтение указателя адреса окончания данных источника, src_data_end_ptr;<br>rdp – чтение указателя адреса окончания данных приемника, dst_data_end_ptr |
| T7-T9 | Контроллер выполняет передачу DMA по каналу C, где:<br>RD – чтение данных;<br>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<br>wc – запись настроек канала, channel_cfg |
| T16 | Контроллер сбрасывает сигнал dma_active[C], что указывает на окончание передачи DMA (см. правило 11).<br>Контроллер устанавливает значение по чтению регистра chnl_useburst_set[C] в 0, если количество оставшихся передач менее 2R (см. правило 15) |

Рисунок 31 показывает работу контроллера DMA при установке dma_waitonreq в 1 и выполнении одиночной DMA передачи.

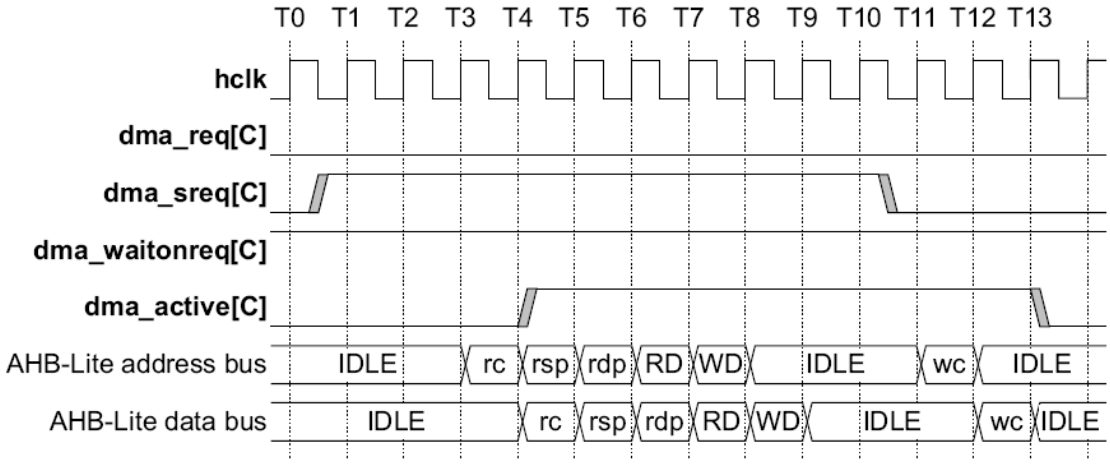



Рисунок 31 – Работа DMA при использовании dma_waitonreq совместно с dma_sreq

Таблица 45 – 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:<br>rc – чтение настроек канала, channel_cfg;<br>rsp – чтение указателя адреса окончания данных источника, src_data_end_ptr;<br>rdp – чтение указателя адреса окончания данных приемника, dst_data_end_ptr |
| T7-T9 | Контроллер выполняет передачу DMA по каналу C, где:<br>RD – чтение данных;<br>WD – запись данных.<br>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<br>wc – запись настроек канала, channel_cfg |
| T13 | Контроллер сбрасывает сигнал dma_active[C], что указывает на окончание передачи DMA (см. правило 11) |

### 10.4.3 *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 46 – Периодичность арбитража в единицах передач по шине AHB

| Значение R | Периодичность арбитража каждые x передач DMA |
|---|---|
| b0000 | 1 |
| b0001 | 2 |

| Значение R | Периодичность арбитража каждые x передач DMA |
|---|---|
| b0010 | 4 |
| b0011 | 8 |
| b0100 | 16 |
| b0101 | 32 |
| b0110 | 64 |
| b0111 | 128 |
| b1000 | 256 |
| b1001 | 512 |
| b1010-b1111 | 1024 |

**Внимание!** 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 в случае, 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».

***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 47 показывает уровень приоритета каналов DMA в порядке его уменьшения.

Таблица 47 – 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 32 иллюстрирует процесс выбора следующего канала для обслуживания.




Рисунок 32 – Алгоритм выбора следующего канала для обслуживания

### 10.4.4 *Типы циклов DMA*

Разряды cycle_cntr определяют, как контроллер будет выполнять циклы DMA. Описание значений этих разрядов приведено в таблице 48.

Таблица 48 – Типы циклов DMA

| cycle_cntr | Описание |
|---|---|
| b000 | Структура управляющих данных канала в запрещенном состоянии |
| b001 | Обычный цикл DMA |
| b010 | Автозапрос |
| 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_cntr 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;
- автозапрос;
- «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:
- Контроллер выполняет $2^R$ передач. Если число оставшихся передач 0, контроллер переходит к шагу 3.
- 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.
- 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:
- Контроллер выполняет $2^R$ передач для канала C. Если число оставшихся передач 0, контроллер переходит к шагу 3.
- Контроллер выполняет арбитраж. Когда канал C обладает более высоким приоритетом, то цикл DMA возвращается к шагу 1.
- 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.

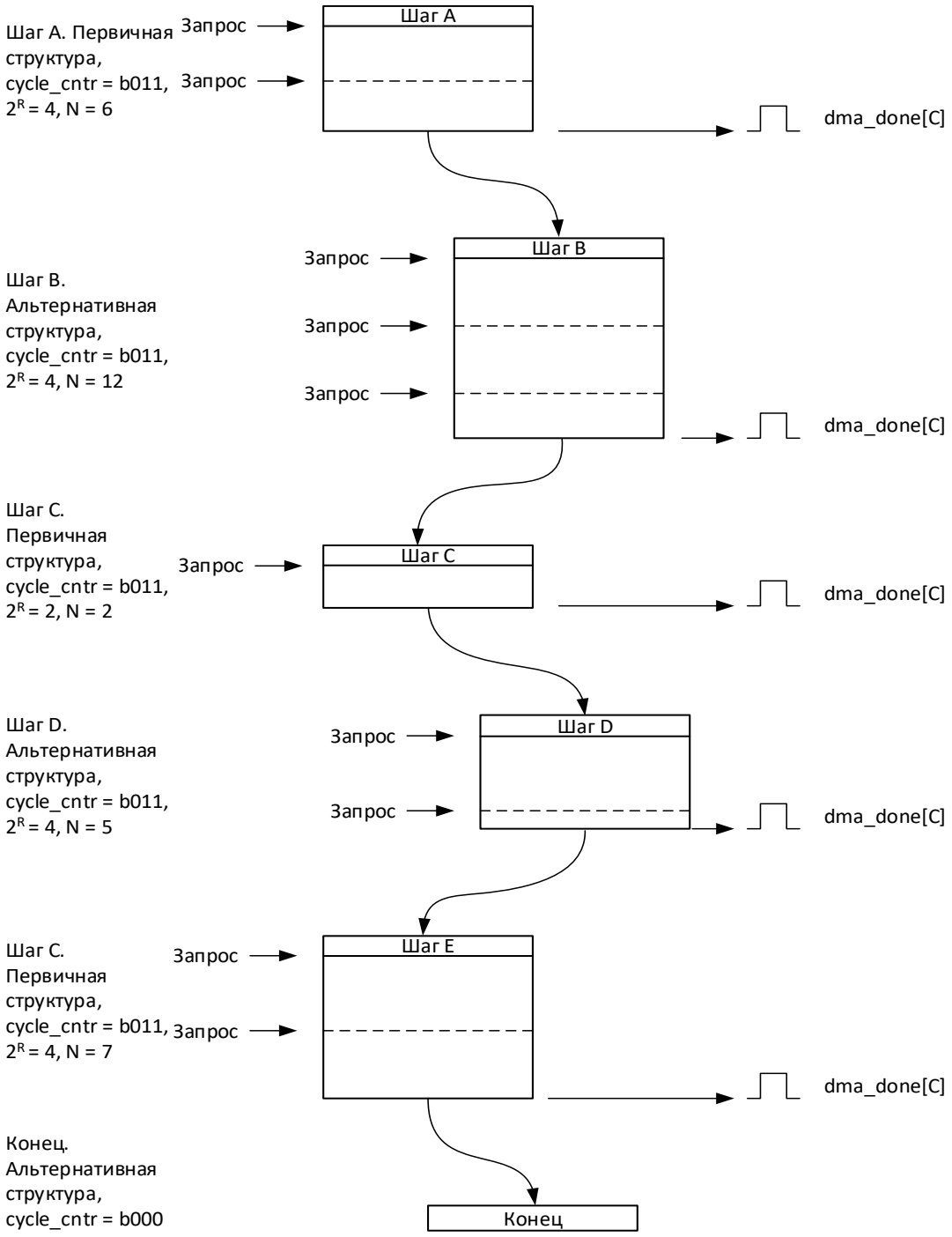



Рисунок 33 – Пример функционирования контроллера в режиме «пинг-понг»

**Шаг А** Процессор устанавливает первичную структуру управляющих данных для шага А.
Процессор устанавливает альтернативную структуру управляющих данных для шага В. Это позволит контроллеру преступить к шагу В незамедлительно после выполнения шага А, при условии, что контроллер не получит запрос на обработку от высокоприоритетного канала.
Контроллер получает запрос и выполняет 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_cntr 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_cntr в 3’b001.

***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_cntr 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 49 – Channel_cfg для первичной структуры управляющих данных в режиме работы с памятью «исполнение с изменением конфигурации»

| Номер бита | Обозначение | Значение | Описание |
|---|---|---|---|
| **Области с константными значениями** | | | |
| 31…30 | dst_inc | b'10 | Контроллер производит инкремент адреса пословно |
| 29…28 | dst_size | b'10 | Контроллер осуществляет передачу пословно |
| 27…26 | src_inc | b'10 | Контроллер производит инкремент адреса пословно |
| 25…24 | src_size | b'10 | Контроллер осуществляет передачу пословно |
| 17…14 | R_power | b'0010 | Контроллер выполняет 4 передачи DMA |
| 3 | next_useburst | b'0 | Для данного режима этот разряд должен быть равен 0 |
| 2…0 | cycle_cntr | b'100 | Контролер работает в режиме работы с периферией «исполнение с изменением конфигурации» |
| **Области со значениями, определяемыми пользователем** | | | |
| 23…21 | dst_prot_cntr | - | Определяет состояние HPROT при записи данных в приемник |
| 20…18 | src_prot_cntr | - | 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 34 демонстрирует пример функционирования в режиме работы с памятью «Исполнение с изменением конфигурации».

Инициализация:

- Настройка первичных управляющих данных для разрешения копирования A, B, C и D: cycle_cntr=b100, $2^R$=4, N=16.
- Запись первичных данных в память с использованием структуры, показанной в таблице ниже.

| | src_data_end_ptr | dst_data_end_ptr | channel_cfg | Unused |
|---|---|---|---|---|
| Data for Task A | 0x0A000000 | 0x0AE00000 | cycle_ctrl = b101, $2^R$ = 4, N = 3 | 0xXXXXXXXX |
| Data for Task B | 0x0B000000 | 0x0BE00000 | cycle_ctrl = b101, $2^R$ = 2, N = 8 | 0xXXXXXXXX |
| Data for Task C | 0x0C000000 | 0x0CE00000 | cycle_ctrl = b101, $2^R$ = 8, N = 5 | 0xXXXXXXXX |
| Data for Task D | 0x0D000000 | 0x0DE00000 | cycle_ctrl = b001, $2^R$ = 4, N = 4 | 0xXXXXXXXX |

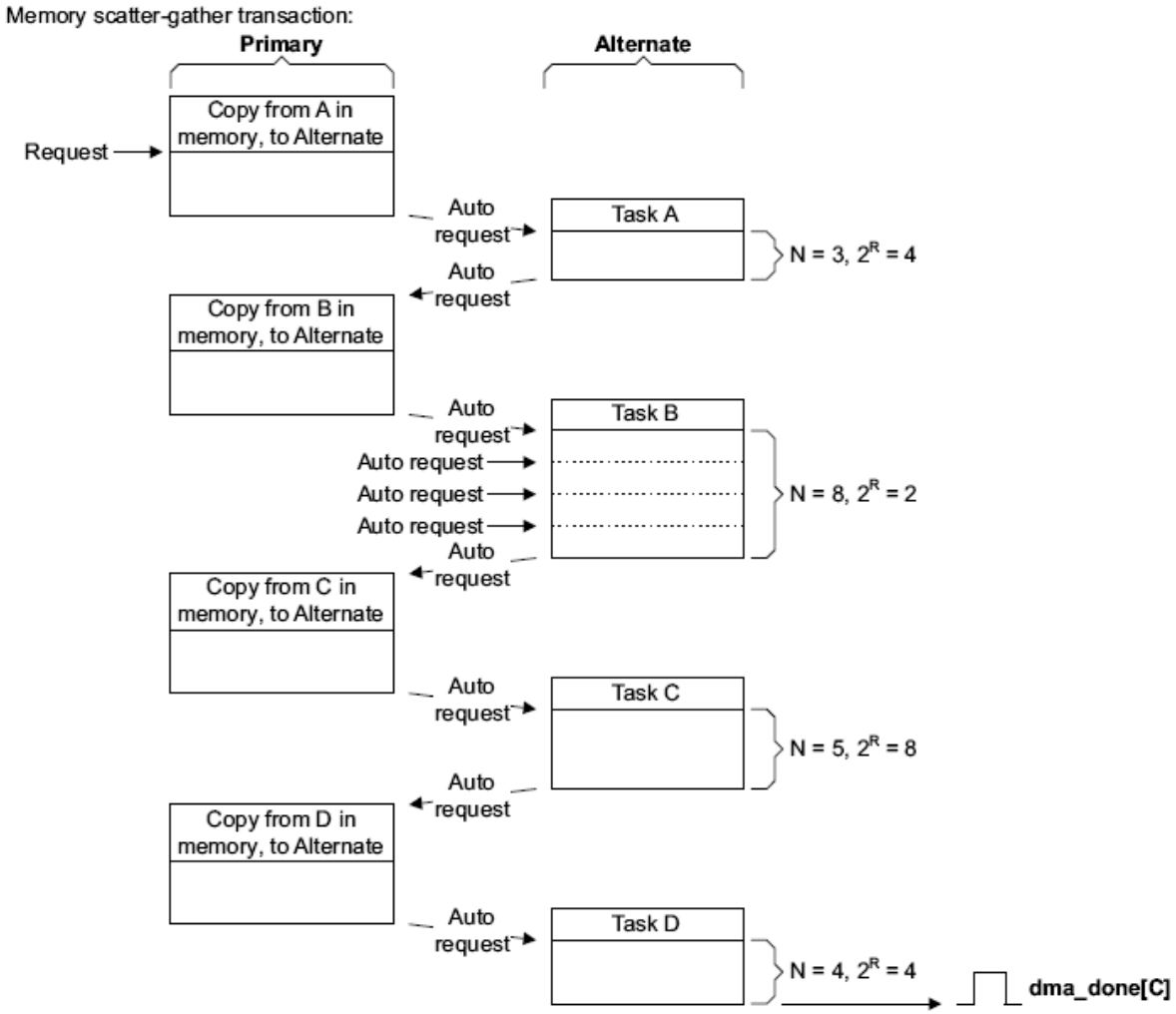



Рисунок 34 – Пример работы DMA в режиме с «Исполнением с изменением конфигурации»

Инициализация:

- Процессор настраивает первичную структуру управляющих данных для работы в режиме работы с памятью «исполнение с изменением конфигурации» путем установки cycle_cntr 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.
- Процессор записывает управляющие данные для шагов A, B, C, D в область памяти с адресом, указанным в src_data_end_ptr.
- 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 получении 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 четыре передачи DMA. Эти передачи записывают альтернативную структуру управляющих данных для шага D. Контроллер устанавливает cycle_cntr 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.

***Режим работы с периферией «исполнение с изменением конфигурации»***

В данном режиме контроллер, получая начальный запрос на обработку, выполняет четыре передачи DMA, используя первичные управляющие данные. По окончании этих передач контроллер начинает цикл DMA, используя альтернативные управляющие данные без осуществления арбитража и не устанавливая сигнал dma_active[C] в 0.

Примечание – Это единственный случай, при котором контроллер не осуществляет процедуру арбитража после выполнения передачи DMA, используя первичные управляющие данные.

После того, как этот цикл завершился, контроллер выполняет арбитраж и по получении запроса на обслуживание от периферии, имеющего наивысший приоритет, он выполняет еще 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_cntr 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 50 – Channel_cfg для первичной структуры управляющих данных в режиме работы с периферией «Исполнение с изменением конфигурации»

| Номер бита | Обозначение | Значение | Описание |
|---|---|---|---|
| **Области с константными значениями** | | | |
| 31…30 | dst_inc | b’10 | Контроллер производит инкремент адреса пословно |
| 29…28 | dst_size | b’10 | Контроллер осуществляет передачу пословно |
| 27…26 | src_inc | b’10 | Контроллер производит инкремент адреса пословно |
| 25…24 | src_size | b’10 | Контроллер осуществляет передачу пословно |
| 17…14 | R_power | b’0010 | Контроллер выполняет 4 передачи DMA |
| 2…0 | cycle_cntr | b’110 | Контролер работает в режиме работы с периферией «исполнение с изменением конфигурации» |
| **Области со значениями, определяемыми пользователем** | | | |
| 23…21 | dst_prot_cntr | - | Определяет состояние HPROT при записи данных в приемник |
| 20…18 | src_prot_cntr | - | 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.

Инициализация:

1 Настройка первичных управляющих данных для разрешения копирования A, B, C и D: cycle_cntr = 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 |

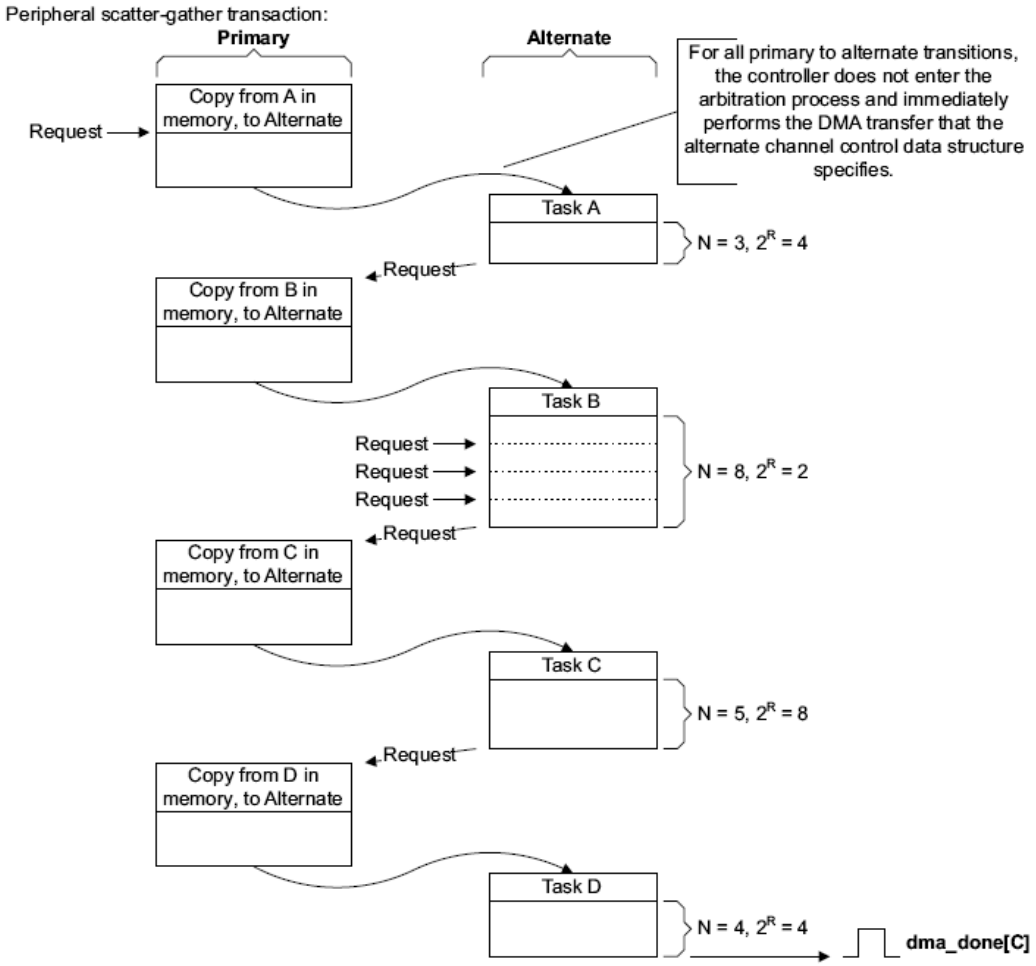



Рисунок 35 – Пример работы DMA в режиме с «Исполнением с изменением конфигурации»

Инициализация:

1 Процессор настраивает первичную структуру управляющих данных для работы в режиме работы с периферией «исполнение с изменением конфигурации» путем установки cycle_cntr 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 четыре 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 четыре

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 три запроса. По окончании контроллер проводит процедуру арбитража. Первичная, копирование из области C памяти. Контроллер выполняет четыре 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 четыре передачи DMA. Эти передачи записывают альтернативную структуру управляющих данных для шага D.

Контроллер устанавливает cycle_cntr 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.

### 10.4.5 *Индикация ошибок*

При получении контроллером по шине AHB ответа об ошибке контроллер 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.

## 10.5 Структура управляющих данных канала DMA

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.

Распределение бит внутри структуры представлены в разделе «Программная модель микроконтроллера» -> «Управляющие структуры и регистры блока прямого доступа в память».

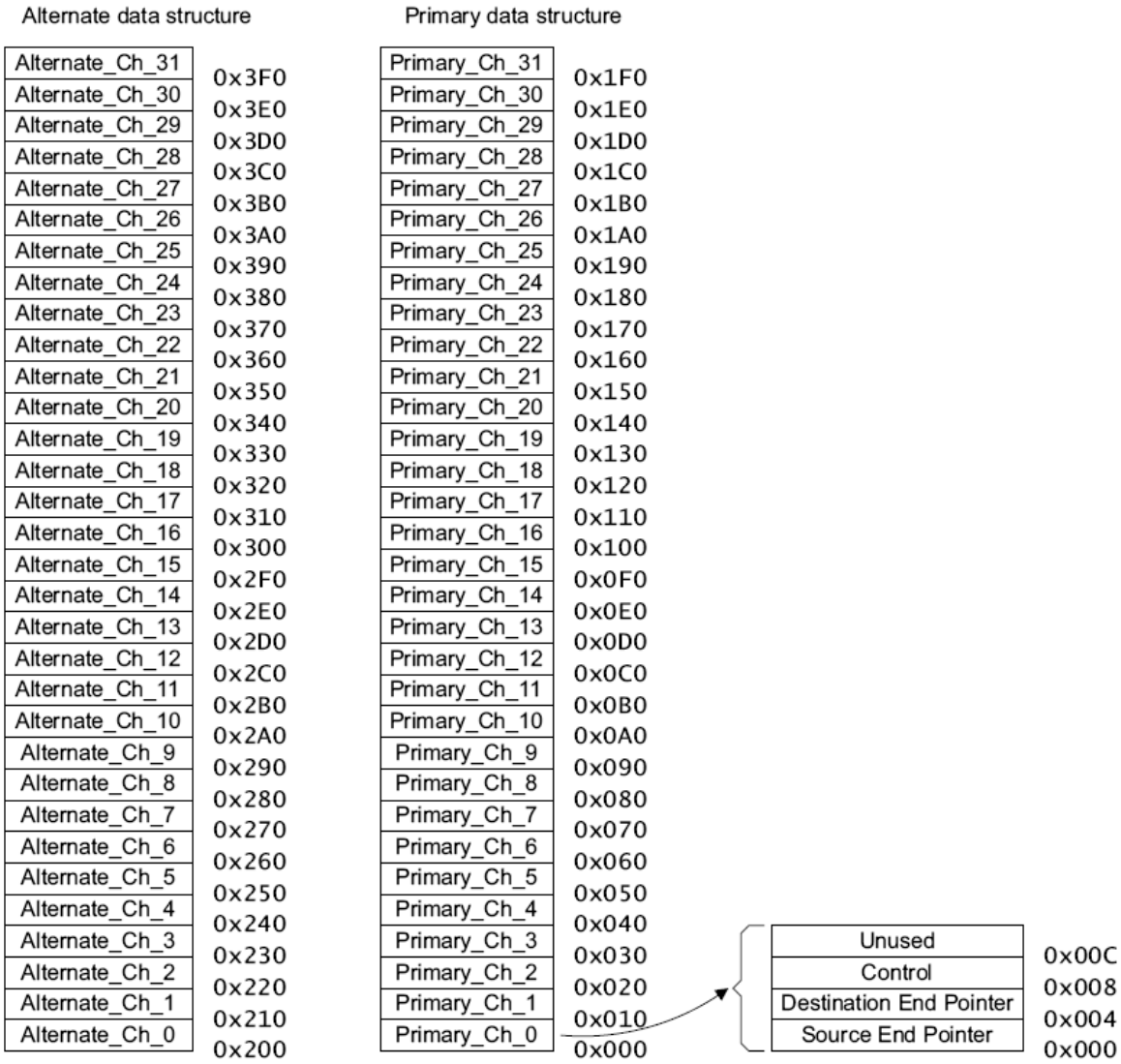



Рисунок 36 – Карта памяти для 32-х каналов, включая альтернативную структуру

Пример, показанный на рисунке, использует 1 Кбайт системной памяти. В этом примере контроллер использует младшие 10 разрядов адреса для доступа ко всем элементам структуры управляющих данных, и поэтому базовый адрес структуры должен быть 0xXXXXX000, далее 0xXXXXX400, далее 0xXXXXX800, далее 0xXXXXXC00.

Существует возможность установить базовый адрес для первичной структуры управляющих данных путем записи соответствующего значения в регистр cntr_base_ptr.

Необходимый размер области системной памяти зависит:

- от количества каналов, используемых в контроллере;
- от того, используется или нет альтернативная структура управляющих данных.

Таблица 51 – 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<br>0x4<br>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 – Базовый адрес альтернативной структуры управляющих данных вычислять не обязательно, так как регистр alt_cntr_base_ptr содержит эту информацию.

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.

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

Таблица 52 – 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.

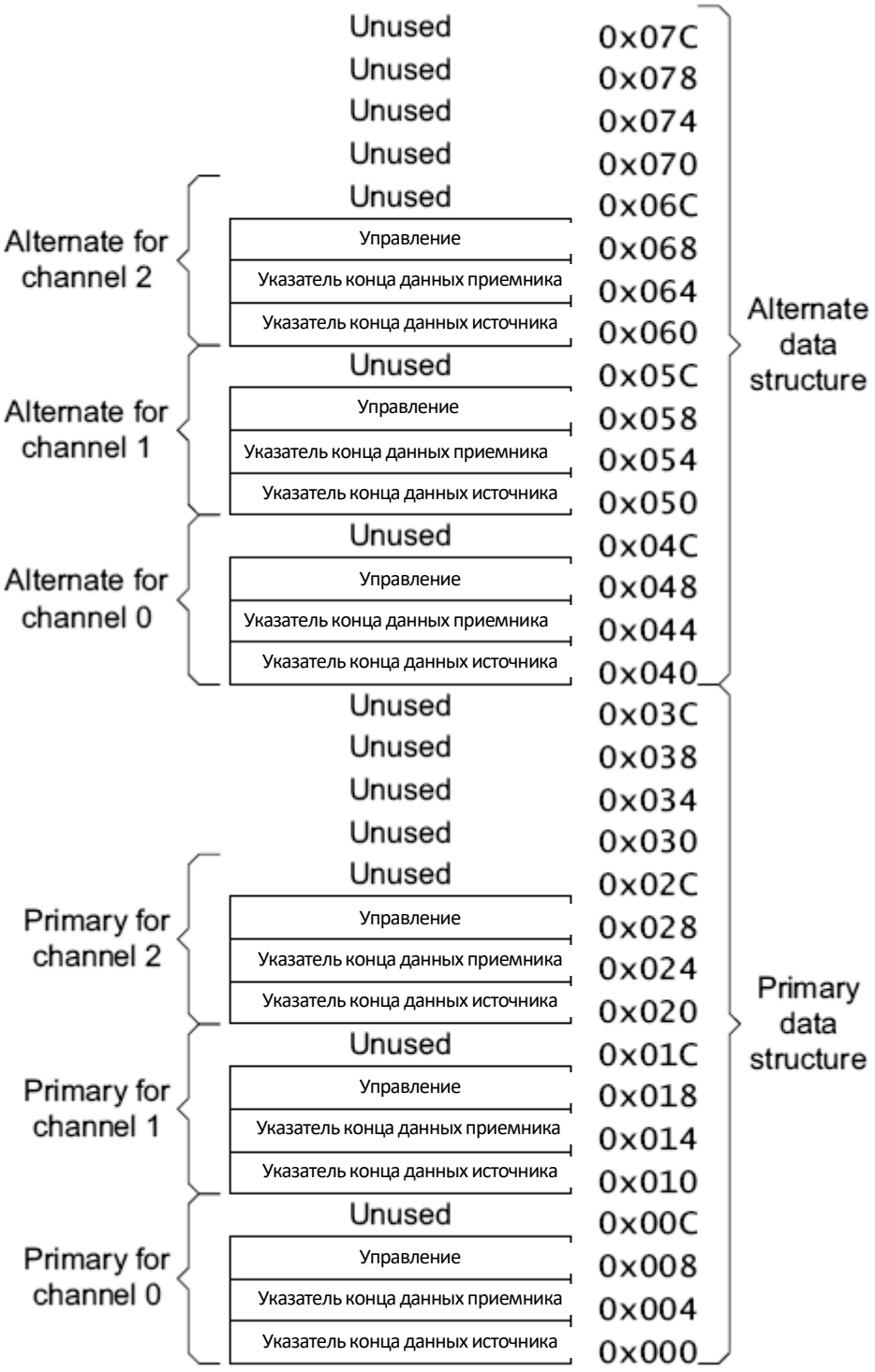



Рисунок 37 – Карта памяти для трех каналов DMA, включая альтернативную структуру

### 10.5.1 Указатель конца данных источника

Область памяти под названием src_data_end_ptr содержит указатель на последний адрес месторасположения данных источника.

Таблица 53 – Значения разрядов src_data_end_ptr

| Номер 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.

### 10.5.2 Указатель конца данных приемника

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

Таблица 54 – Значения разрядов dst_data_end_ptr

| Номер 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.

#### 10.5.2.1 *Вычисление адреса источника*

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 55 – 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_cntr=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_cntr=0 |

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

Таблица 56 – 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_cntr=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_cntr=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_cntr=0**), R_power=b11 | | | | |

* 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_cntr.

# 11 Контроллер обработки событий отказов, сбоев и ошибок (FT_CNTR)

В контроллере реализован механизм обработки событий отказов, основанный на дублировании исполняемой логики критических узлов микросхемы и помехозащищенном кодировании данных в системной памяти ROM, FLASH, RAM и регистрах ряда периферийных блоков.

Информация о событиях отказов поступает в контроллер событий отказов, где фиксируется в регистрах. По событиям отказов может быть настроено формирование прерываний и сброс микрокнтроллера.

В контроллере обработки событий отказов реализовано три уровня возникновения событий отказов.

## 11.1 Уровень диагностики 3

Третий уровень диагностики реализован в блоках:

- батарейный домен с часами реального времени BKP_CNTR;
- блок контроля тактовых частот CLK_CNTR;
- блок управления сбоями, ошибками и отказами FT_CNTR;
- блок сторожевого таймера IWDG_CNTR.

В данных блоках идет непрерывный контроль целостности данных через троирование либо ECC-кодирование. Таким образом, при возникновении одиночного сбоя, ошибки или отказа в данных блоках, они продолжат выполнение своей функции.

## 11.2 Уровень диагностики 2

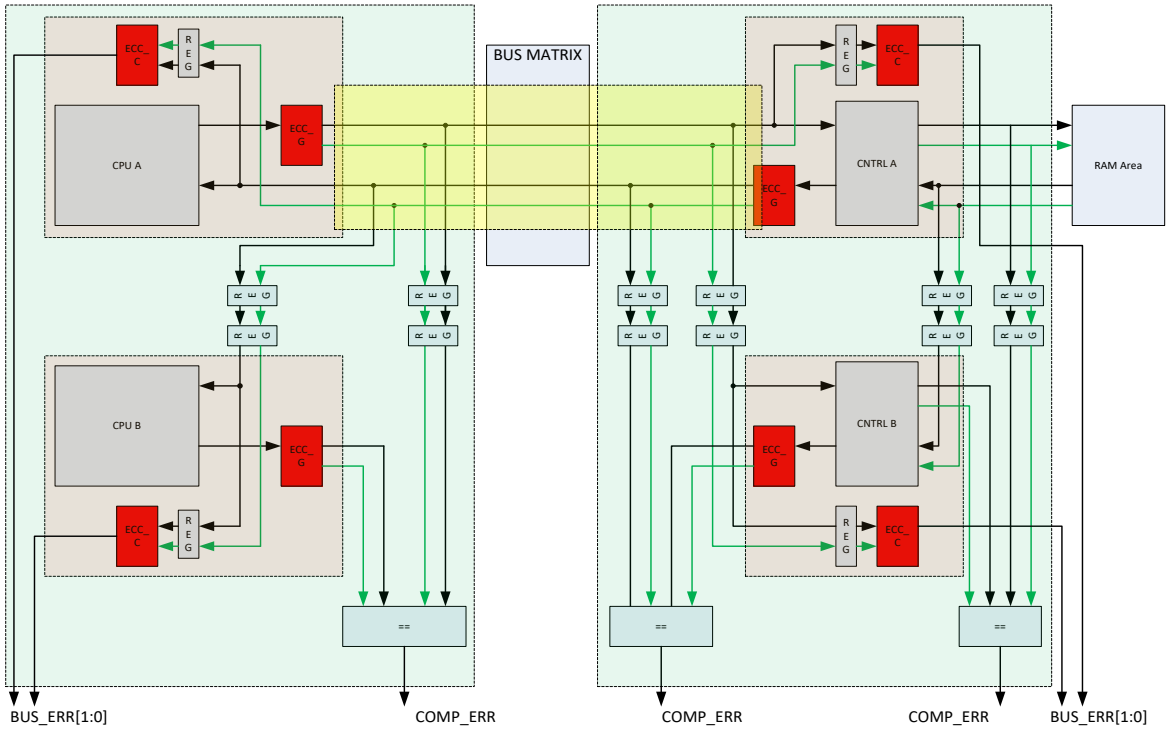



Рисунок 38 – Схема включения ядер в режиме LOCK STEP

Второй уровень диагностики обеспечивает обнаружение сбоев, ошибок и отказов в блоках:

- процессорное ядро (в режиме LOCKSTEP);
- контроллер DMA (в режиме LOCKSTEP);
- контроллер памяти ПЗУ;
- контроллер памяти FLASH;
- контроллер памяти SRAM_D;

- контроллер памяти SRAM_C;
- контроллер внешней системной шины EXT_BUS_CNTR.

В данных блоках реализована схема обнаружения сбоев за счет дублирования аппаратуры. Обнаружение сбоев осуществляется за счет сравнения поведения двух блоков, находящихся в режиме LOCKSTEP. В данном режиме (на примере процессорного ядра) процессорное ядро CPUA формирует запросы в память и периферию и получает ответы от памяти и периферии. При этом все входные сигналы CPUA с задержкой в два такта заводятся и на процессорное ядро CPUB. Выходные сигналы CPUA с задержкой на два такта и выходные данные CPUB сравниваются и, если в поведении двух ядер возникли отличия, то в одном из них произошел сбой. Разнесение на два такта для двух процессорных ядер служит для исключения влияния общей для обоих ядер помехи (по питанию, тактовой частоте и т.п.). За счет того, что процессорное ядро CPUB фактически отстает на два такта, при синхронной помехе для обоих ядер, в ядре CPUA сбой возникнет при исполнении n-ой инструкции, а ядре CPUB сбой возникнет при исполнении (n-2)-ой инструкции. Таким образом, схема сравнения выявит эти ошибки.

Аналогичным способом защищены контроллеры FLASH_CNTR, SRAM_C_CNTR, SRAM_D_CNTR и контроллер внешней шины EXT_BUS_CNTR.

Для процессорных ядер CPUA и CPUB режим LOCKSTEP может быть отключен, в этом случае ядра переходят в режим DUALCORE, становятся независимыми и могут исполнять независимые программы. В режиме DUALCORE реализована синхронизация процессорных ядер с помощью инструкций WFE и SEV. Для контроллеров памяти FLASH_CNTR, SRAM_C_CNTR, SRAM_D_CNTR и контроллера внешней шины EXT_BUS_CNTR данный режим не отключаем, и осуществляет непрерывный контроль работы этих блоков.

### 11.3 Уровень диагностики 1

Первый уровень диагностики реализован непосредственно для массивов памяти и системных шин. Диагностика и исправление сбоев и ошибок построены за счет избыточного кодирования ECC SEC-DED (исправление одиночных ошибок и обнаружение двойных). Подробное описание ECC-кодирования для блоков памяти приведено в разделе «Помехозащищенное кодирование». ECC-кодирование реализовано для блоков памяти:

- ROM;
- FLASH;
- SRAM_C;
- SRAM_D;
- BKP_REG с 0 по 59;
- память данных в CACHE;
- память тегов адресов в CACHE;
- память в блоке CAN;
- память в блоке MIL;
- память в блоке ETHERNET.

Ошибки в данных массивах памятей обнаруживаются при выполнении операции чтения. Для уменьшения времени обнаружения ошибки и минимизации вероятности ее перерастания в двойню ошибку может быть применен блок SCRUBBER. Подробнее в описании блока SCRUBBER.

Также с помощью ECC-кодирования защищены и внутренние системные шины. Причем данный уровень диагностики перекрывается с вторым уровнем. Например, контроллер памяти данных после чтения данных из памяти вычисляет для них ECC-сумму и возвращает процессорному ядру. При этом данные с вычисленной ECC-суммой идут по схеме диагностики LOCKSTEP и сравниваются с аналогичными данными и ECC полученными во втором контроллере. Таким образом, проверяется, что отправленные данные корректны и для них вычислена корректная ECC-сумма. При приеме данных проверка ECC-суммы также производится после раздачи данных двум процессорным

ядрам, образующим LOCKSTEP схему. Таким образом, гарантируется достоверность доставленных процессорным ядрам данных.

## 11.4 Уровень диагностики 0

К нулевому уровню диагностики относятся все периферийные блоки микроконтроллера. На данном уровне предусмотрена только диагностика, регламентированная стандартами периферийных блоков (вычисление контрольных сумм CRC, битов четности, проверки разрешённых временных интервалов передачи данных и т.п.).

## 11.5 Помехозащищенное кодирование

В микроконтроллере применяется помехозащищенное кодирование SEC-DED (исправление одиночных и обнаружение двойных ошибок) кодом Хемминга (72,64) для внутренних памятей и памяти на внешней системной шине и кодом Хемминга (7,4) для задания режима работы.

Кодом Хемминга (72,64) всегда защищаются внутренние памяти, включая кэш память. ECC-кодирование может использоваться также для защиты внешней памяти. С помощью ECC кодируются как непосредственно сами данные, так и адрес расположения данных. В результате, автоматически исправляются одиночные ошибки в шине данных, обнаруживаются двойные ошибки в поле данных, и обнаруживаются одиночные или двойные ошибки в поле адреса. Одиночные ошибки в поле адреса не исправляются и считаются неустранимыми ошибками. Кодирование производится при операциях записи в блоках ECC_T. На основании записываемых данных и адреса вычисляется поле HWECC[7:0] и передается совместно с полем данных (далее по тексту генерация ECC). При считывании на основании данных, адреса и поля HRECC[7:0] в блоке ECC_R проверяется полученная информация и при необходимости исправляется одиночная ошибка в поле данных, либо обнаруживается одиночная и двойные ошибки в поле адреса и двойные в поле данных (далее по тексту проверка ECC).

### 11.5.1 *ECC-кодирование для ПЗУ*

При обращении в область загрузочного ПЗУ памяти, проверка и, при необходимости, исправление по ECC происходит на границе контроллера памяти ПЗУ и непосредственно самой матрицей ПЗУ. После чего корректные данные с контрольной суммой передаются в процессорные ядра, где после разветвления их в LOCKSTEP схеме они еще раз проверяются по ECC, но уже без исправления, только с целью обеспечения достоверности.

ECC-биты загрузочной программы, вычислены на этапе создания программы и зашиты в ПЗУ микроконтроллера на этапе разработки.

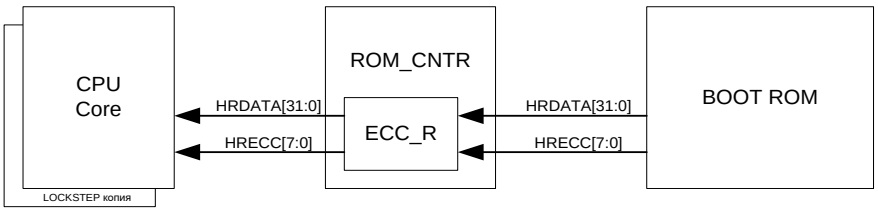



Рисунок 39 – Схема контроля ECC на пути от ПЗУ

### 11.5.2 *ECC-кодирование для FLASH*

При обращении в область FLASH проверка ECC происходит на границе контроллера FLASH и непосредственно самой матрицей FLASH. При этом из матрицы извлекается строка с размером в общей сложности 160 бит (128 бит данных и 32 бита ECC). Для всех слов строки проводится проверка и при необходимости исправление данных по ECC. Затем данные передается в кэш данных (DCACHE) или кэш инструкций (ICACHE) и сохраняются в них с ECC. При этом одновременно данные передаются и в процессорное

ядро. Возвращаемые процессорному ядру данные снабжены ECC-суммой, которая используется для обеспечения достоверности передаваемых данных. При этом, так как контроллер FLASH имеет сложную структуру, работающую под управлением автомата состояний, и для обеспечения диагностики сбоев в блоке контроллера FLASH он реализован в LOCKSTEP режиме. Таким образом, сравнение поведение двух копий контроллера позволяет обнаружить сбои, возникшие в самом контроллере FLASH.

Генерация ECC бит при программировании FLASH памяти осуществляется программными средствами разработки программ и вычисляется при программировании FLASH памяти.

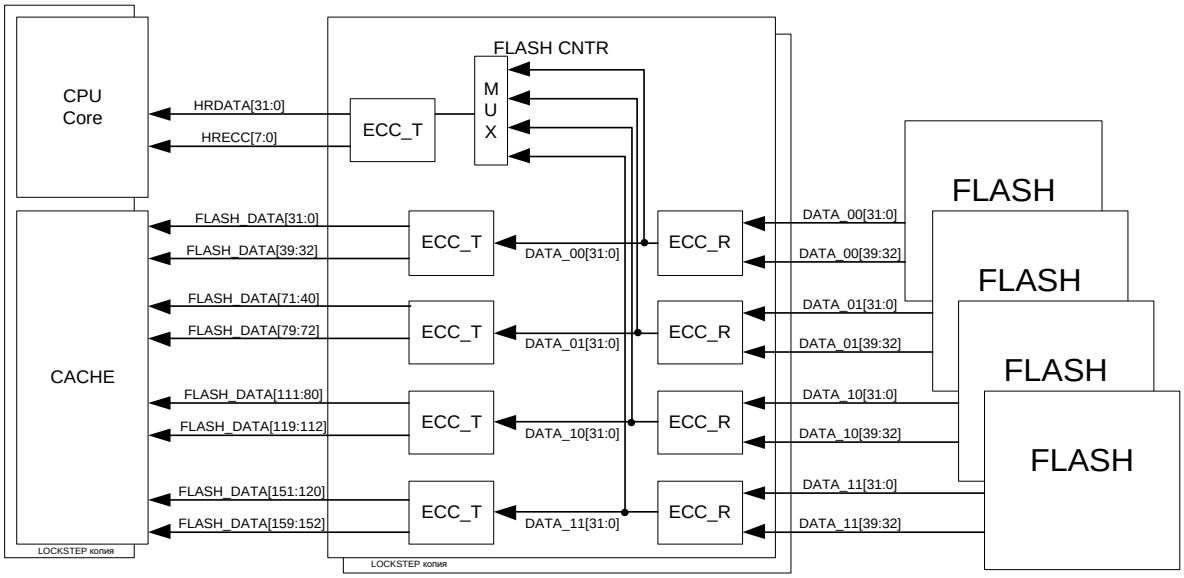



Рисунок 40 – Схема контроля ECC на пути от FLASH памяти

### 11.5.3 *ECC-кодирование для RAMC и RAMD*

При обращении в область RAMC или RAMD генерация и проверка ECC происходит на границе контроллера памяти и непосредственно самой матрицы.

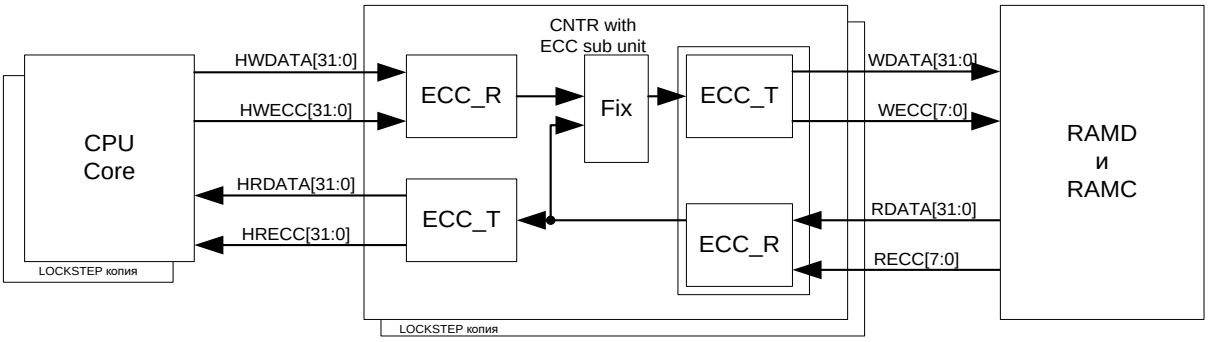



Рисунок 41 – Схема контроля ECC для RAMD и RAMC

В ряде случаев при записи в память сначала выполняется чтение. Это связано с необходимостью обработки байтов и полуслов при работе с памятью, имеющей 32-х битную организацию с защитой ECC. Так, например, при записи байта в RAMD необходимо считать из памяти ранее записанное слово с ECC, проверить его ECC, модифицировать в проверенном слове записываемый байт, сгенерировать новое ECC и записать его обратно в память. Также необходимо иметь в виду, что после включения питания микроконтроллера память содержит произвольные значения и требует проведение первоначальной процедуры инициализации.

### 11.5.4 *ECC-кодирование для внешней шины*

При обращении в область EXT_BUS существует два варианта организации ECC – параллельный (аналогичный организации ECC для RAMC и RAMD и последовательный (когда ECC-биты расположены в отдельной области памяти). Таким образом, при работе с внешней памятью в зависимости от режима ECC-биты считываются одновременно с данными (параллельная организация) либо в два этапа (последовательная организация, сначала считываются данные, затем ECC) и уже затем производится проверка ECC (Подробнее смотрите контроллер EXT_BUS).

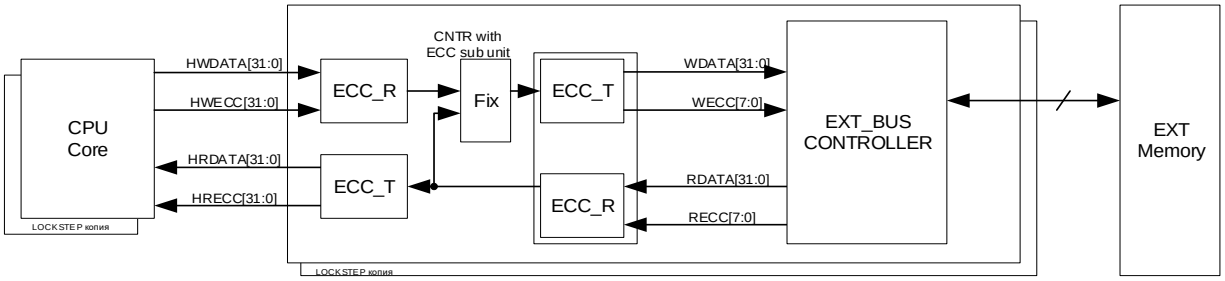



Рисунок 42 – *Схема контроля ECC для EXTBUS контроллера.*

Данные, считанные по внешней шине через шину инструкций – IBus или шину данных – DBus могут быть закэшированы в кэш. Для этого при их возвращении процессору к ним снова генерируется ECC для записи в кэш память. При работе кэш памяти когерентность закэшированных данных не обеспечивается. Также необходимо иметь в виду, что после включения питания внешняя память может содержать произвольные значения и требует дополнительной процедуры инициализации.

### 11.5.5 *ECC-кодирование для периферии*

Периферийные блоки, имеющие в своем составе большие массивы памяти, также защищены ECC, но при этом, в отличие от памяти RAMC или RAMD, запись в эти памяти возможна только 32-х разрядными словами. Таким образом, при записи в блоке ECC_T формируются ECC-биты, а при считывании в блоке ECC_R проверяются.

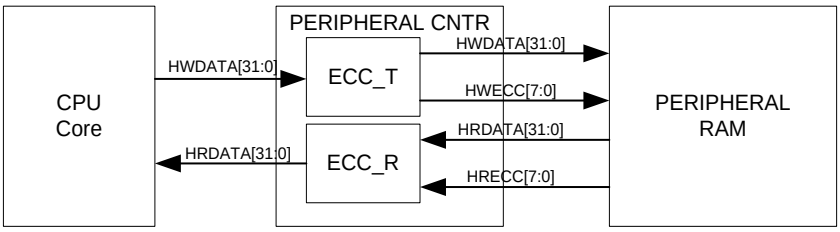



Рисунок 43 – Схема контроля ECC для периферии

Периферийные блоки также могут иметь собственные механизмы исправления и обнаружения ошибок.

# 12 Домен батарейного питания микроконтроллера

## 12.1 Контроллер батарейного домена (BKP_CNTR)

Блок батарейного домена предназначен для хранения основной конфигурации микроконтроллера, обеспечения функций часов реального времени и сохранения некоторого набора пользовательских данных при отключении основного источника питания. При снижении основного питания $V_{CC_DCDC}$ ниже уровня $U_{PORON}$ в блоке формирования питания происходит автоматическое переключение питания опорного питания блока батарейного домена $V_{CC_BKP}$ с питания $V_{CC_DCDC}$ на $V_{CC_BKP}$. Если в схеме подключено питание $V_{CC_BKP}$, то батарейный домен остается включенным и может выполнять свои функции. Встроенный регулятор формирует питание цифровой части батарейного домена $VLDO_{BKP}$ Напряжение $V_{CC_BKP}$ используется для питания генератора LSE, а напряжение $VLDO_{BKP}$ для регистров управления системой и регистров общего назначения, входящих в батарейный домен, сторожевого таймера, часов реального времени и генератора LSI. Таким образом, генераторы могут быть использованы для тактирования таймеров и регистров батарейного домена во время отключения основного питания.

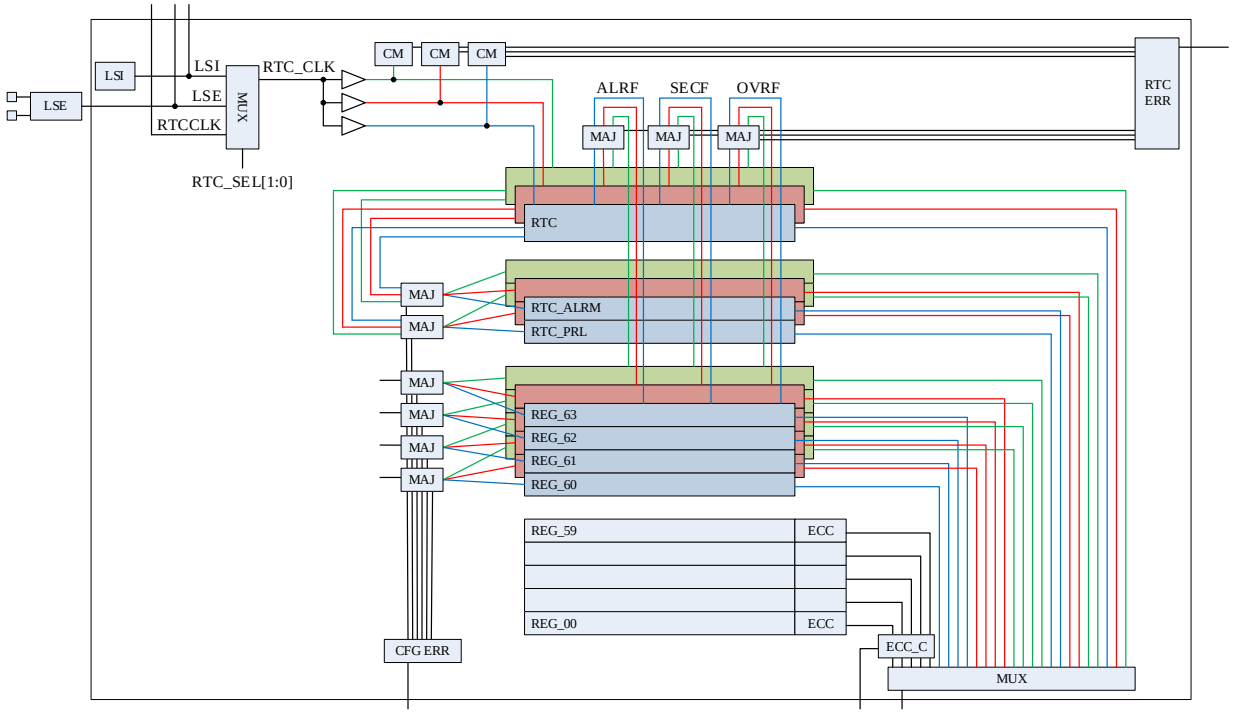



Рисунок 44 – Структурная блок-схема контроллера батарейного домена

## 12.2 Часы реального времени (RTC)

Часы реального времени позволяют организовать механизм отсчета времени в кристалле, в том числе при отключении основного источника питания. В качестве источника тактовой частоты часов реального времени RTC_CLK могут использоваться:

- выходной сигнал генератора LSI – встроенного кольцевого генератора, не требующий использования дополнительных площадок. Для питания используется $V_{CC_BKP}$, генератор может быть использован в момент отключения основного питания $V_{CC_DCDC}$;
- выходной сигнал генератора LSE – встроенного генератора частоты на основе входного высокоточного сигнала от внешнего источника. Для питания используется $V_{CC_BKP}$, генератор может быть использован в момент отключения основного питания $V_{CC_DCDC}$.

– сигнал тактовой частоты RTCCLK, формируемый в блоке управления синхросигналами CLK_CNTR. Сигнал RTCCLK может быть использован только при наличии питания $V_{CC_DCDC}$.

Выбор источника тактовой частоты часов реального времени RTC_CLK осуществляется битами RTCSEL[1:0] в регистре REG_64_TMRx.

Для отсчета секунд в часах реального времени применяется 20-битный предварительный делитель входной тактовой частоты RTC_CLK, на выходе которого формируется тактовый сигнал SEC_CLK. Регистр RTC_DIV выступает в качестве счетчика предварительного делителя, который тактируется на частоте RTC_CLK. Регистр RTC_PRL задает коэффициент деления предварительного делителя, при этом счетчик RTC_DIV инкрементируется в интервале от 0 до RTC_PRL. Коэффициент деления в регистре RTC_PRL должен быть задан таким образом, чтобы частота SEC_CLK составляла 1 Гц.

Для калибровки тактовой частоты RTC_CLK используются биты RTCCAL[7:0] (см. описание ошибки 3025 в документе Errata Notice). Значение RTCCAL[7:0] определяет, какое число тактов RTC_CLK из $2^{20}$ будет замаскировано с помощью дополнительного счётчика RTC_20. Данный способ калибровки заключается в том, что один период RTC_DIV, который приходится на начало интервала $2^{20}$, удлиняется на RTCCAL[7:0] тактов частоты RTC_CLK, или на RTCCAL[7:0]/RTC_CLK секунд. Таким образом, с помощью бит RTCCAL[7:0] производится замедление хода часов. Для ускорения хода часов необходимо задать меньшее, чем требуется, значение регистра RTC_PRL, а затем произвести замедление с помощью бит RTCCAL[7:0]. Изменение значения бит RTCCAL[7:0] может быть осуществлено в ходе работы часов реального времени.

Регистр RTC_CNT предназначен для отсчета времени в секундах и работает на выходной частоте предварительного делителя SEC_CLK. Регистр RTC_ALRM предназначен для задания времени, при совпадении с которым вырабатывается флаг прерывания и пробуждения процессора ALRF. Бит STANDBY, отключающий все внутренние регуляторы напряжения, автоматически сбрасывается при превышении RTC_CNT значения в регистре RTC_ALRM.

Процесс выполнения записи регистра RTC_CNT отображается с помощью флага WEC в регистре RTC_CS. На время выполнения записи флаг WEC устанавливается в «1», по окончании записи флаг WEC сбрасывается в «0». Пока флаг WEC установлен в «1», новая запись в регистр RTC_CNT (во все регистры RTC_CNT_TMR1-3) запрещена.

## 12.3 Контроллер сторожевого таймера (WDG_CNTR)

В микросхеме реализован сторожевой таймер, при использовании которого микроконтроллер может быть сброшен при несоблюдении временных интервалов обязательного периода сброса.

Сторожевой таймер WDG предназначен для аварийного перезапуска микроконтроллера, когда за период счета сторожевого таймера не будет выполнена хотя бы одна операция по его перезапуску. Сторожевой таймер выполнен в виде сбоеустойчивой логики, то есть при возникновении одиночного сбоя, таймер продолжит выполнять поставленную задачу. Для последующего устранения последствий сбоя требуется программная переинициализация таймера.

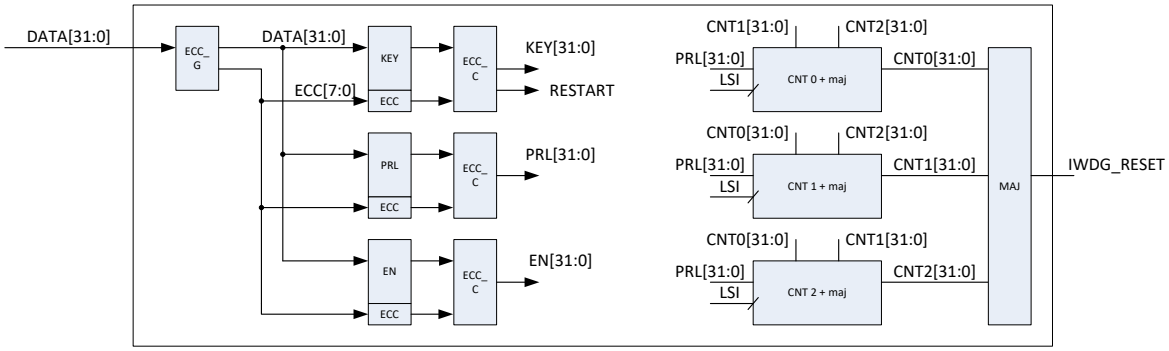



Рисунок 45 – Структурная схема блока сторожевых таймеров

***Для инициализации сторожевого таймера необходимо:***

1 в регистр KEY записать 0x05555;
2 считать регистр и убедиться, что записано верное значение;
3 в регистр PRL записать требуемое основание счета;
4 считать регистр и убедится, что записано верное значение;
5 записать в регистр KEY значение 0x0AAAA;
6 через время не менее 200 мкс значение PRL будет автоматически переписано в регистр CNT.
7 считать регистр CNT и убедиться, что в него записано верное значение из регистра PRL;
8 в регистр KEY записать 0x0CCCC;
9 считать регистр и убедиться, что записано верное значение;
10 в регистр EN записать значение 0x03333. После чего сторожевой таймер начнет работу;
11 при возникновении сбоев провести операции по переинициализации сторожевого таймера;
12 с периодом, меньшим заданного через регистр PRL, проводить перезапуск сторожевого таймера;
13 контролировать, что генератор LSI работает.

***Для перезапуска сторожевого таймера необходимо:***

1 при необходимости считать значение CNT и проверить, что оно находится в ожидаемом диапазоне;
2 записать в регистр KEY значение 0x0AAAA.

***Для переинициализации сторожевого таймера необходимо:***

1 в регистр KEY записать 0x0CCCC;
2 считать регистр и убедится, что записано верное значение;
3 в регистр EN записать значение, отличное от 0x03333. После чего будет остановлен счет;
4 выполнить операции по инициализации сторожевого таймера.

## 12.4 Способы обнаружения сбоев и исправления ошибок в батарейном домене

В батарейном домене применяются различные механизмы обнаружения и исправления сбоев. Пользовательская память использует биты ECC для автоматического исправления одиночных ошибок и обнаружения двойных. Основная конфигурация и часы реального времени выполнены на троированной логике с непрерывным контролем возникновения ошибок в схемах мажорирования. Мажорирование не исправляет сбоев

непосредственно в том канале, в котором он возник, а исправляет выход схемы, влияющий на поведение микросхемы в целом. Таким образом, при возникновении сбоя в одном из каналов, схема мажорирования исправит его влияние на систему в целом, и будет его обозначать флагом ошибки вплоть до его исправления программными средствами.

При записи в регистры батарейной памяти (REG_60_SYSx, REG_61_PWRx, REG_62_PWRx, REG_63_CLKx, REG_64_TMRx и BLDO_CNTRx) запись происходит одновременно в три регистра, при этом не имеет значения, по какому адресу из этих трех регистров производится запись.

Во время чтения данные всех трех регистров проходят через схему мажорирования, и возможные ошибки исправляются. При этом, если один из трех регистров отличается от других, будет выставлен флаг соответствующей ошибки.

Мажорирование в блоках сторожевого таймера и часов реального времени происходит по такому же принципу, за исключения того, что для записи значения регистра необходимо программно записать все три его дубликата, после чего очистить возникшие в контроллере ошибки.

# 13 Таймеры, блоки широтно-импульсной модуляции и обработки сигналов датчиков

## 13.1 Контроллер таймеров общего назначения (TIMER_CNTR)

В микросхеме реализовано четыре блока таймеров общего назначения, каждый из которых может быть использован для широкого спектра применений, включая:

- подсчет циклов частоты TIM_CLK или каких-либо внешних событий;
- формирование прерываний и запросов DMA по заданным событиям;
- захват входных сигналов, в том числе измерение длительности импульсов входных сигналов;
- генерацию различных форм выходных сигналов.

Основу таймеров составляет 32-битный перезагружаемый счетчик. Счет может быть прямой, обратный или двунаправленный. В качестве источника синхросигнала может выступать внутренняя тактовая частота TIM_CLK, внешние сигналы или другие таймеры.

В каждый блок таймера входит четыре канала, которые имеют в своем составе схему захвата и блок ШИМ с функциями формирования «мертвой зоны» и аппаратной блокировки.

Каждый из таймеров позволяет генерировать прерывания и запросы DMA.

### 13.1.1 *Основные характеристики*

Основные характеристики блока таймера:

- 32-битный счетчик: счет прямой, обратный или двунаправленный;
- 32-разрядный предварительный делитель частоты TIM_CLK;
- схема выбора источника тактирования основного счетчика от внешних сигналов или от других таймеров;
- четыре независимых канала, каждый канал может работать в одном из режимов:
  - режим захвата: позволяет захватить (сохранить) текущее значение счетчика при изменении некоторого входного сигнала;
  - режим ШИМ: позволяет осуществлять непрерывное сравнение заданных значений со значением счетчика для формирования выходных сигналов;
- формирование выходных сигналов в режиме ШИМ:
  - сброс в НИЗКИЙ уровень при совпадении;
  - установка в ВЫСОКИЙ уровень при совпадении;
  - переключение (инвертирование) при совпадении;
  - переключение при некотором условии;
- формирование прерываний и запросов DMA по событиям:
  - обновление счетчика;
  - захват;
  - сравнение;
  - внешние события по входам ETR и BRK.

#### 13.1.1.1 Структурная схема

Структурная схема блока «Таймер» представлена на рисунке 46.

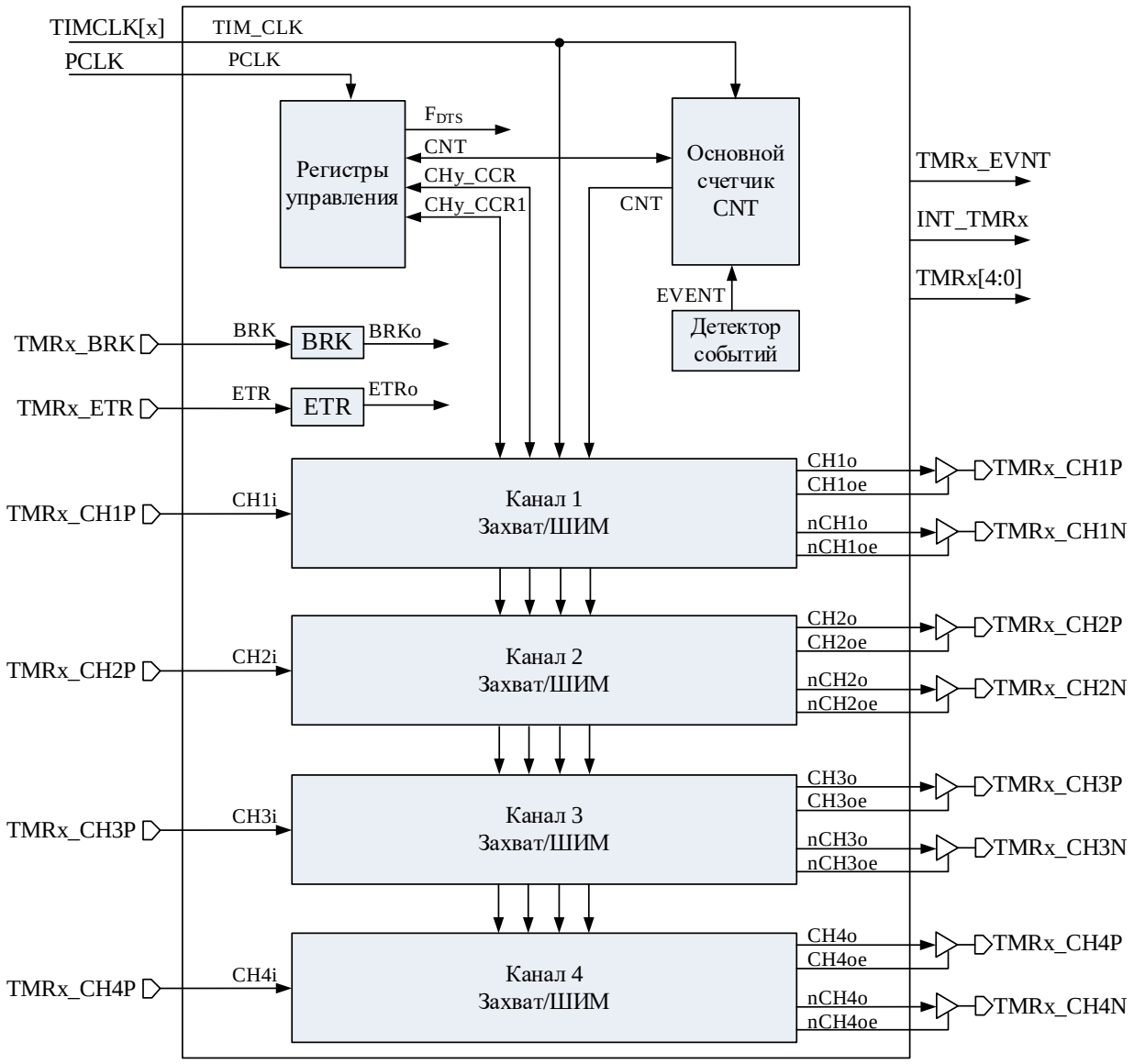



Рисунок 46 – Структурная схема блока «Таймер»

Таймер содержит основной 32-битный счетчик CNT, блок регистров управления и четыре канала схем захвата/ШИМ.

Таймер позволяет работать в режимах:

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

### 13.1.2 *Базовый блок таймера*

Таймер построен на базе 32-битного счетчика. Базовый блок таймера включает в себя:

- основной счетчик таймера (CNT);
- основание счета (максимальное значение) основного счетчика (ARR);
- делитель частоты TIM_CLK (PSG), используемый для тактирования основного счетчика;
- регистр управления основным счетчиком (CNTRL).

Сигналом для изменения основного счетчика CNT может служить как внутренняя частота TIM_CLK, так и события в других счетчиках, либо внешние входные сигналы (см. «Источники событий для счета»).

#### 13.1.2.1 Инициализация тактирования таймера

Перед началом работы с таймером в первую очередь должны быть включены тактовые сигналы. Параметры задаются в блоке «Контроллер тактовых частот (CLK_CNTR)».

Таймер общего назначения тактируется частотами PCLK и TIM_CLK. Частота PCLK используется для записи/чтения регистров блока по шине APB. Работа блоков таймера осуществляется на частоте TIM_CLK.

Для разрешения тактовой частоты PCLK необходимо установить бит тактирования блока в регистре PER1_CLK: бит 0 для таймера 0, бит 1 для таймера 1, бит 2 для таймера 2, бит 3 для таймера 3. Задание тактовой частоты TIM_CLK осуществляется в регистре TIMx_CLK: в поле DIV[15:0] устанавливается коэффициент деления тактовой частоты MAX_CLK для формирования частоты TIM_CLK, разрешение подачи частоты TIM_CLK на блок таймера управляется битом EN_CLK. Тактовая частота TIM_CLK должна быть меньше или равна тактовой частоте PCLK (CPU_CLK).

После подачи тактовых сигналов можно приступать к работе с таймером.

#### 13.1.2.2 Инициализация основного счетчика таймера

Чтобы запустить работу основного счетчика необходимо задать:

- начальное значение основного счетчика таймера в регистре CNT;
- значение основания счета для основного счетчика в регистре ARR;
- режим работы счетчика в регистре CNTRL:
  - выбрать источник события переключения счетчика EVNT_SEL[3:0];
  - режим счета основного счетчика CNT_MODE[1:0]:
  - значения 00 и 01 при тактировании внутренней частотой;
  - значения 10 при тактировании внешними сигналами;
  - направление счета основного счетчика DIR;
- при тактировании внутренней частотой установить значение предварительного делителя в регистре PSG, основной счетчик при этом будет считать на частоте TIM_CLKd = TIM_CLK / (PSG + 1);
- разрешить работу счетчика CNT_EN.

Значения регистров CNT, PSG и ARR можно изменять даже во время работы счетчика. Значения регистров CNT и PSG вступят в силу мгновенно после их записи. Значение регистра основания счета (ARR) может вступить в силу сразу после записи, если в регистре CNTRL бит ARRB_EN = 0.

При установленном бите ARRB_EN = 1 записанное значение ARR применяется при CNT == ARR. Необходимо учитывать, что если установлен прямой счет таймера, то новое значение ARR будет использоваться в следующем периоде счета. Если установлен обратный счет таймера, то новое значение ARR будет использовано через один период счета.

Поле CNT_MODE[1:0] в регистре CNTRL определяет режим работы основного счетчика:

- CNT_MODE[1:0] = 00 или 10 – направление счета определяется битом DIR:
  - DIR = 0 – счет прямой;
  - DIR = 1 – счет обратный;
- CNT_MODE[1:0] = 01 – счет двунаправленный с автоматическим изменением DIR.

#### 13.1.2.3 Режимы счета

***Счет прямой: CNT_MODE[1:0] = 00, DIR = 0***

```
TIMERx_CNTR->CNTRL  = 0x00000000;//Режим инициализации таймера
//Настраиваем работу основного счетчика
TIMERx_CNTR->CNT = 0x00000004;//Начальное значение счетчика
TIMERx_CNTR->PSG = 0x00000000;//Предделитель частоты TIM_CLK
TIMERx_CNTR->ARR = 0x00000013;//Основание счета
//Разрешение работы таймера
TIMERx_CNTR->CNTRL = 0x00000001;//Счет прямой по TIM_CLK
```

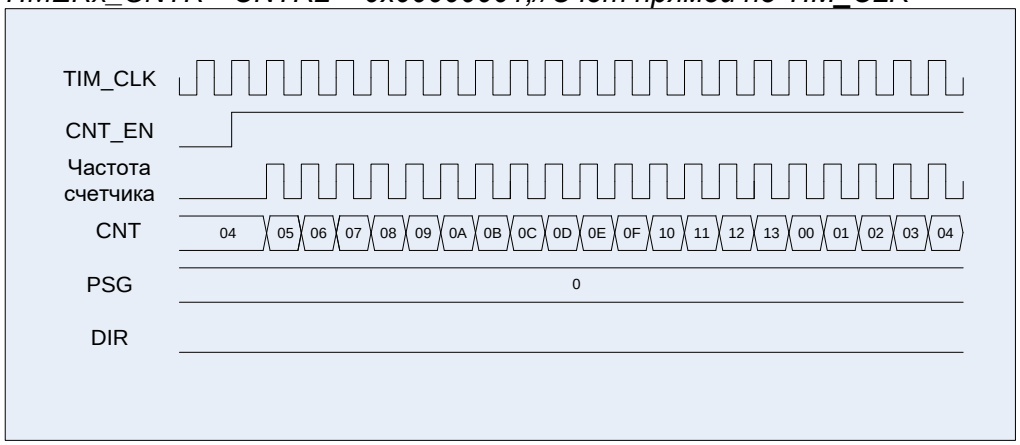



Рисунок 47 – Диаграммы работы таймера, счет прямой от 0 до 0x13, стартовое значение 0x04

***Счет обратный: CNT_MODE[1:0] = 00, DIR = 1***

```
TIMERx_CNTR->CNTRL = 0x00000000;//Режим инициализации таймера
//Настраиваем работу основного счетчика
TIMERx_CNTR->CNT = 0x00000004;//Начальное значение счетчика
TIMERx_CNTR->PSG = 0x00000000;//Предделитель частоты TIM_CLK
TIMERx_CNTR->ARR = 0x00000013;//Основание счета
//Разрешение работы таймера.
TIMERx_CNTR->CNTRL = 0x00000009;//Счет обратный по TIM_CLK
```

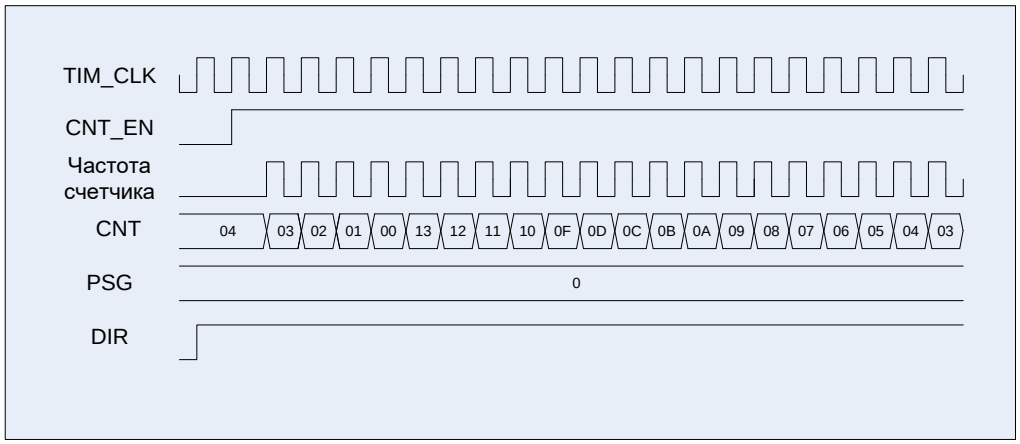



Рисунок 48 – Диаграммы работы таймера, счет обратный от 0x13 до 0, стартовое значение 0x04

***Счет двунаправленный: CNT_MODE = 01, DIR = 0***

```
TIMERx_CNTR->CNTRL  = 0x00000000;//Режим инициализации таймера
//Настраиваем работу основного счетчика
TIMERx_CNTR->CNT = 0x00000004;//Начальное значение счетчика
TIMERx_CNTR->PSG = 0x00000000;//Предделитель частоты TIM_CLK
TIMERx_CNTR->ARR = 0x00000013;//Основание счета
//Разрешение работы таймера.
TIMERx_CNTR->CNTRL = 0x00000041;//Счет двунаправленный по TIM_CLK
```

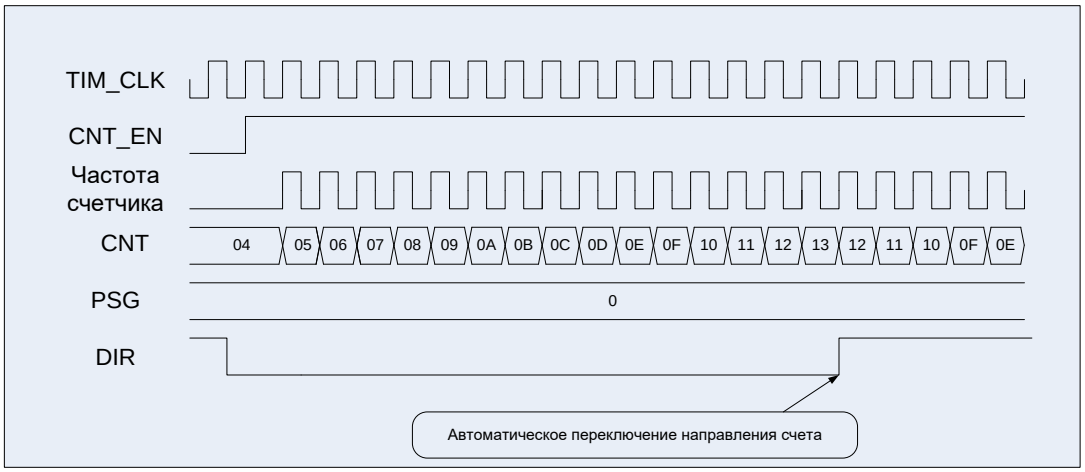



Рисунок 49 – Диаграммы работы таймера, счет двунаправленный, сначала прямой

***Счет двунаправленный: CNT_MODE = 01, DIR = 1***

```
TIMERx_CNTR->CNTRL  = 0x00000000;//Режим инициализации таймера
//Настраиваем работу основного счетчика
TIMERx_CNTR->CNT = 0x00000004;//Начальное значение счетчика
TIMERx_CNTR->PSG = 0x00000000;//Предделитель частоты TIM_CLK
TIMERx_CNTR->ARR = 0x00000013;//Основание счета
//Разрешение работы таймера.
TIMERx_CNTR->CNTRL = 0x00000049;//Счет двунаправленный по TIM_CLK
```

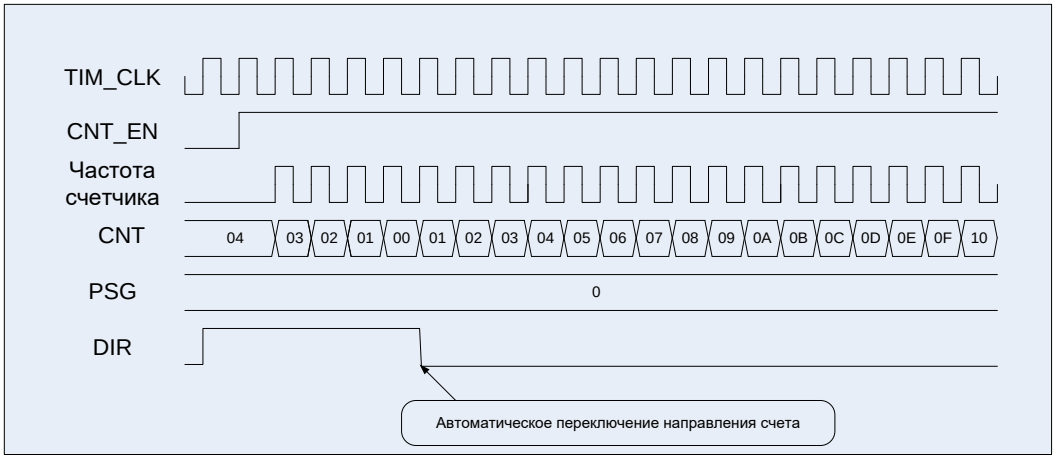



Рисунок 50 – Диаграммы работы таймера, счет двунаправленный, сначала обратный

#### 13.1.2.4 Тактовая частота $F_{DTS}$

В блоке таймера предусмотрено формирование дополнительной тактовой частоты $F_{DTS}$, которая может использоваться для работы генератора «мертвой зоны» и цифровых фильтров на входах ETR и CHyi.

Тактовая частота $F_{DTS}$ формируется из частоты TIM_CLK путём прореживания на заданный коэффициент (1, 2, 3 или 4). Настройка частоты $F_{DTS}$ осуществляется в регистре CNTRL, поле FDTS[1:0].

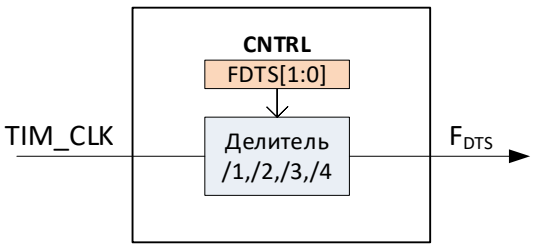



Рисунок 51 – Схема формирования тактовой частоты $F_{DTS}$

Диаграмма возможных частот FDTS приведена на рисунке 52.

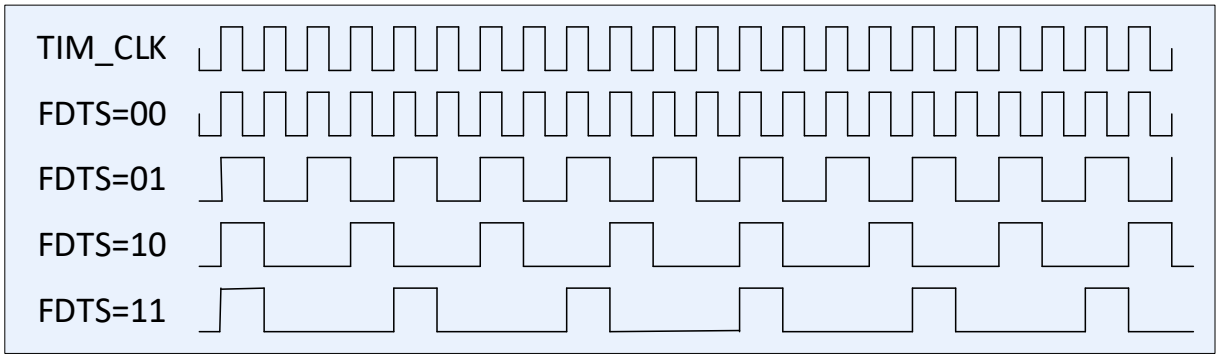



Рисунок 52 – Диаграмма тактовой частоты $F_{DTS}$ в зависимости от значения FDTS[1:0] в регистре CNTRL

### 13.1.3 *Источники событий для счета*

Тактирование основного счетчика таймера может осуществляться от следующих источников:

- внутренний тактовый сигнал (TIM_CLKd);
- событие в другом таймере (CNT==ARR);
- внешний тактовый сигнал, «Режим 1»: событие переднего фронта на входе канала CHyi;
- внешний тактовый сигнал, «Режим 2»: событие переднего или заднего фронта на входе ETR.

Выбор источника тактирования основного счетчика осуществляется в регистре CNTRL, поле EVNT_SEL[3:0]. При выборе любого источника, кроме внутреннего тактового сигнала (EVNT_SEL[3:0] = 0000), необходимо также установить CNT_MODE[1:0] = 10 в регистре CNTRL.

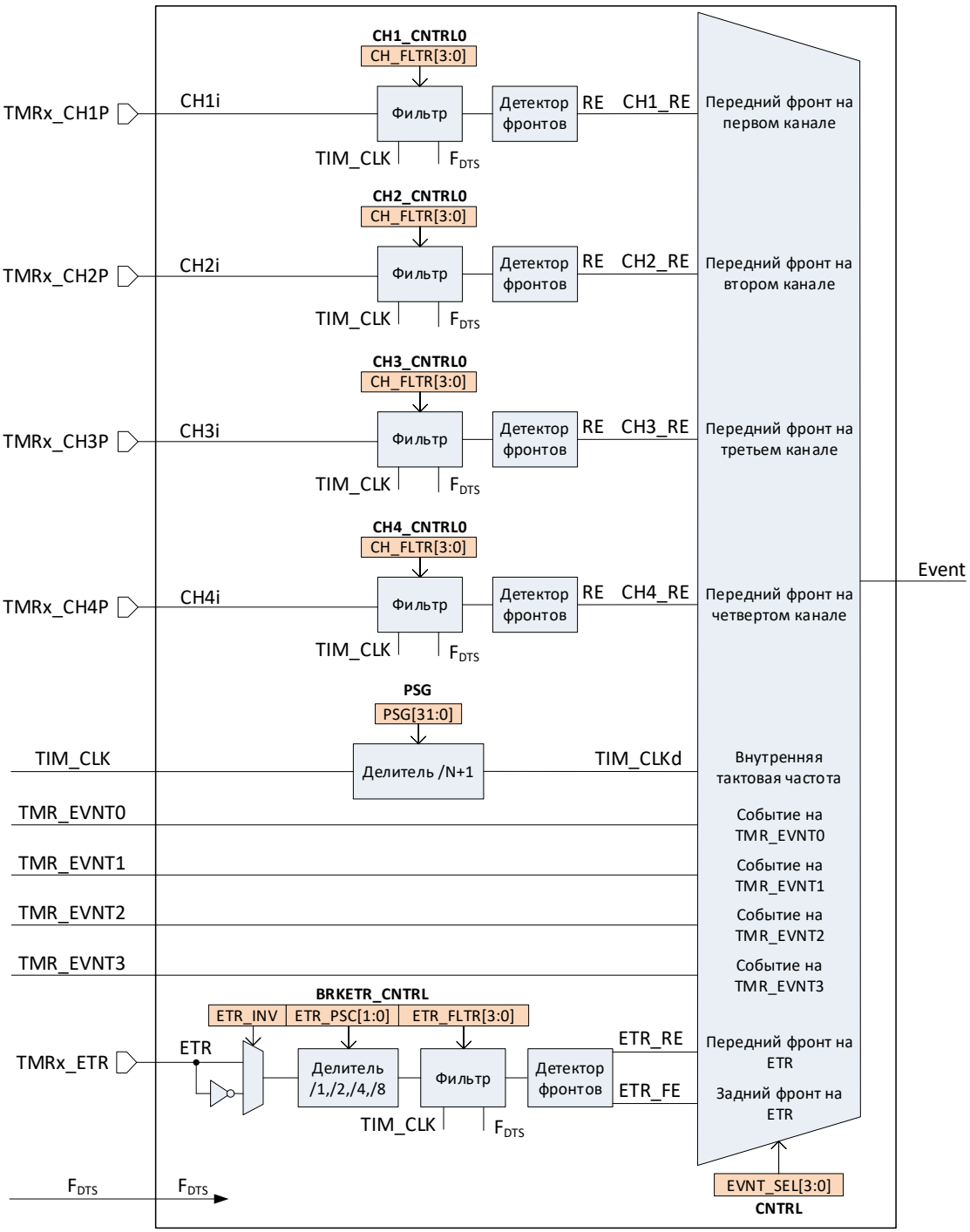



Рисунок 53 – Структурная схема формирования события для счета

#### 13.1.3.1 Внутренний тактовый сигнал (TIM_CLKd)

Данный режим выбирается, когда EVNT_SEL[3:0] = 0000 и CNT_MODE[1:0] = 0x в регистре CNTRL. Основной счетчик таймера тактируется от внутренней частоты TIM_CLKd, которая формируется путем деления частоты TIM_CLK в соответствии с коэффициентом деления, записанным в регистре PSG.

Если значение предварительного делителя основного счетчика (PSG) неравно нулю, то счетный регистр делителя будет инкрементироваться по каждому импульсу сигнала TIM_CLK до тех пор, пока не достигнет значения, находящегося в регистре делителя. Далее счетный регистр делителя сбрасывается в ноль, содержимое основного счетчика таймера изменяется на 1 и счет начинается заново. Таким образом выходная частота предварительного делителя составляет:

$$\mathrm{TIM_CLKd} = \frac{\mathrm{TIM_CLK}}{\mathrm{PSG} + 1} \quad (8)$$

Значение регистра PSG можно изменять даже во время работы счетчика, новое значение предделителя вступит в силу сразу после записи. На рисунках 54 и 55 приведены диаграммы работы счетчика при обновлении значения PSG.

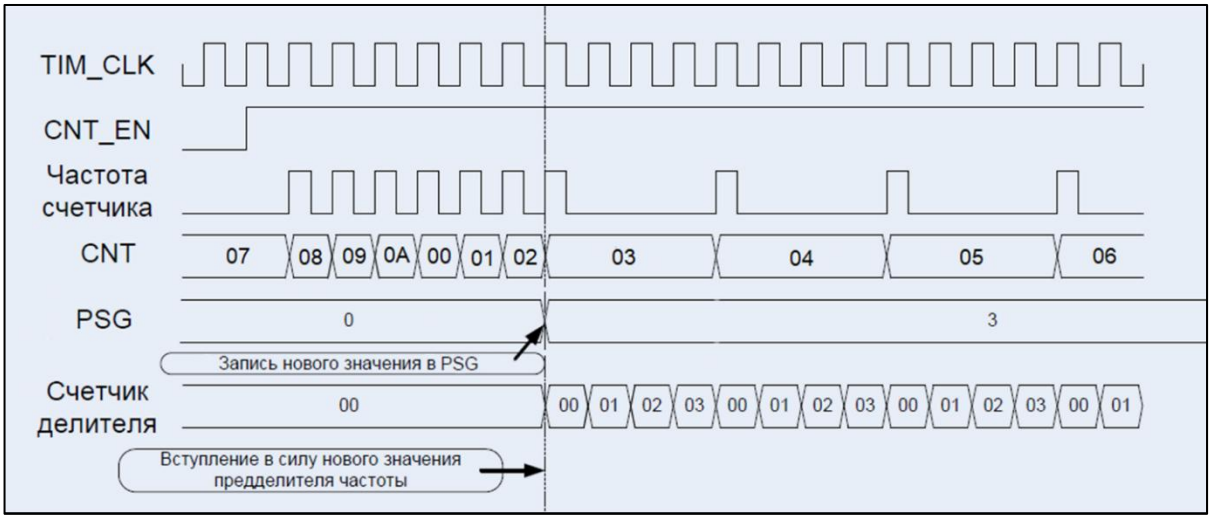



Рисунок 54 – Диаграмма работы счетчика: счет прямой
(CNT_MODE[1:0] = 00, EVNT_SEL[3:0] = 0000, DIR = 0)

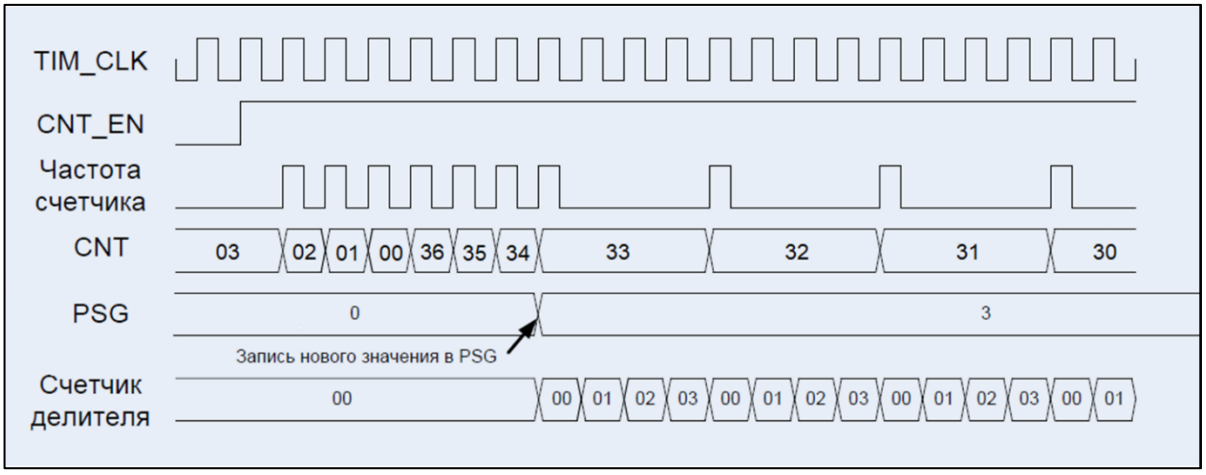



Рисунок 55 – Диаграмма работы счетчика: счет обратный
(CNT_MODE[1:0] = 00, EVNT_SEL[3:0] = 0000, DIR = 1)

#### 13.1.3.2 Событие в другом таймере (CNT==ARR)

Все таймеры полностью независимы друг от друга, но при этом у них предусмотрена возможность синхронизированной работы. Это позволяет создавать более сложные массивы таймеров, которые работают полностью автономно и не требуют написания какого-либо кода программы для выполнения сложных временных функций.

У каждого таймера имеется выход запуска TMRx_EVNT, который соединен с входами других таймеров. Подключение выходов запуска TMRx_EVNT к входам других таймеров приведено на рисунке 56.

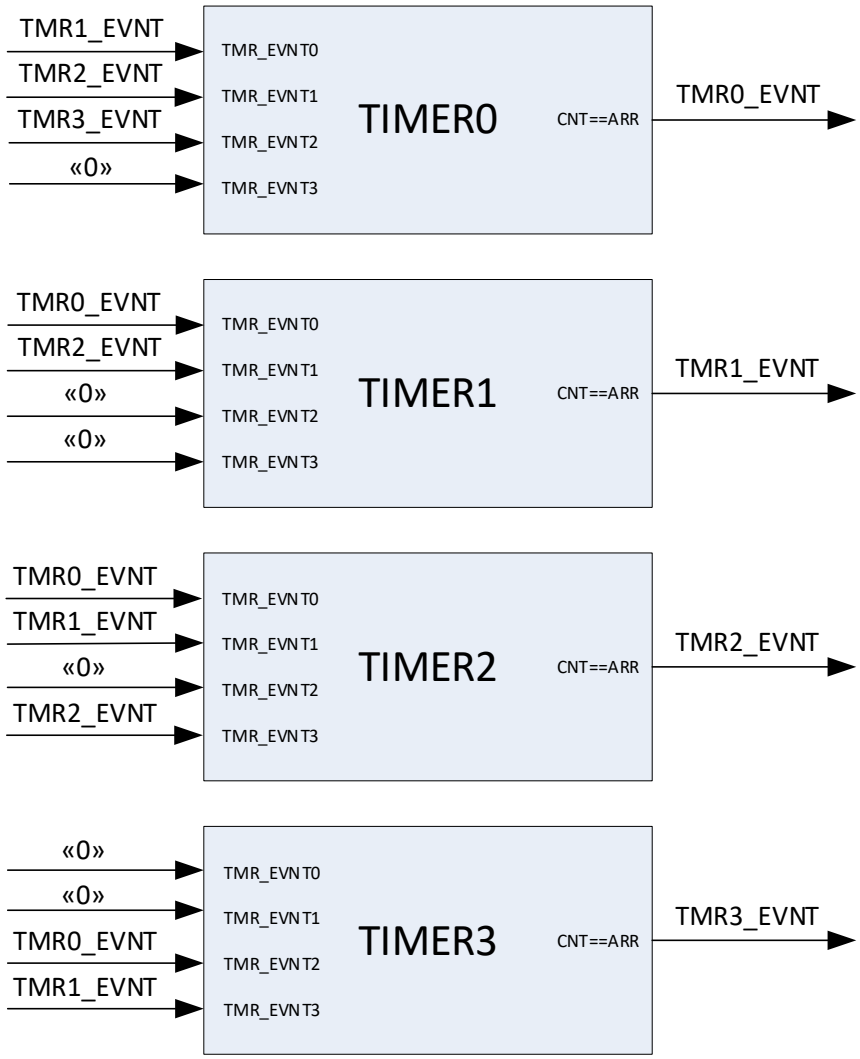



Рисунок 56 – Распределение выходов запуска по блокам таймеров

Тактирование от другого таймера выбирается, когда EVNT_SEL[3:0] = 0001 – 0011 или 1010, а также CNT_MODE[1:0] = 10 в регистре CNTRL. Основной счетчик таймера тактируется от другого таймера по сигналу TMRx_EVNT, который устанавливается при CNT == ARR. Пересинхронизация сигнала TMRx_EVNT (CNT == ARR) с одного таймера на другой происходит с задержкой один такт частоты TIM_CLK.

Синхронизация таймеров возможна в различных режимах. На рисунке 57 показан пример каскадного соединения таймеров, диаграммы работы данных таймеров приведены на рисунке 58.

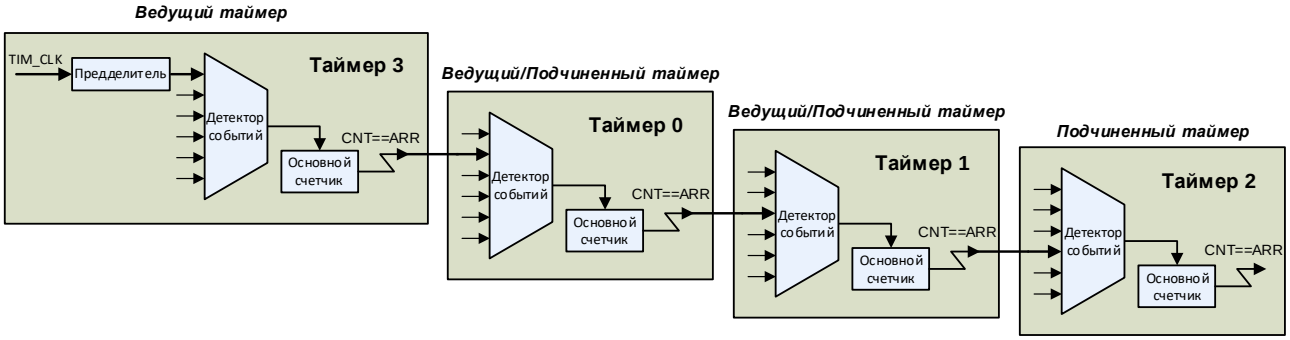



Рисунок 57 – Пример каскадного соединения таймеров

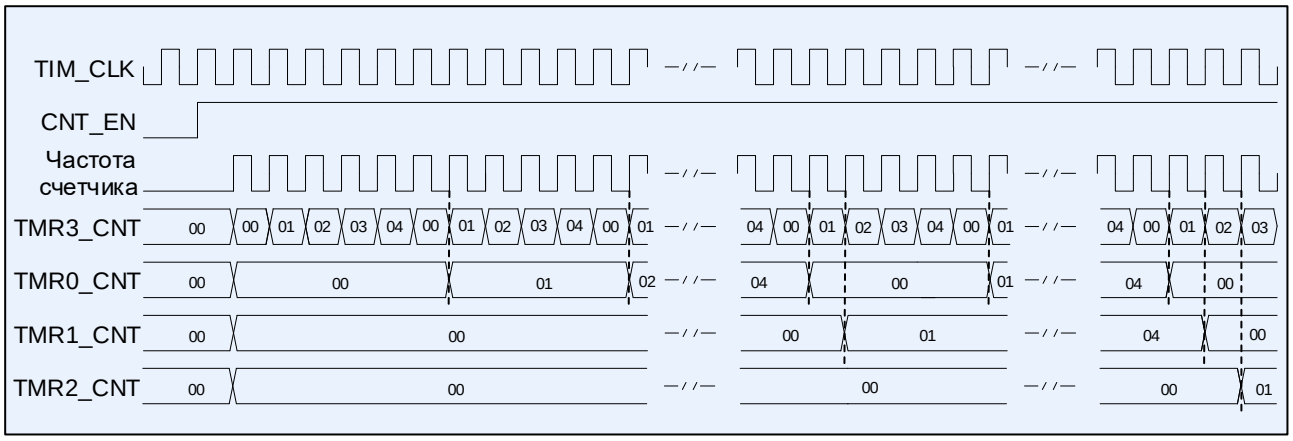



Рисунок 58 – Диаграммы работы четырех таймеров в каскаде
TIMER3: DIR = 0, EVNT_SEL[3:0] = 0000, CNT_MODE[1:0] = 00;
TIMER0: DIR = 0, EVNT_SEL[3:0] = 0011, CNT_MODE[1:0] = 10;
TIMER1: DIR = 0, EVNT_SEL[3:0] = 0001, CNT_MODE[1:0] = 10;
TIMER2: DIR = 0, EVNT_SEL[3:0] = 0010, CNT_MODE[1:0] = 10

#### 13.1.3.3 Внешний тактовый сигнал, «Режим 1»: событие переднего фронта на входе канала CHyi

Данный режим выбирается, когда EVNT_SEL[3:0] = 01xx и CNT_MODE[1:0] = 10. Основной счетчик таймера считает по переднему фронту внешнего сигнала, поступающего на вход канала CHyi. Биты CH_SEL[1:0] регистра CHy_CNTRL0 не оказывают влияния, так как они применяются для работы канала таймера только в режиме захвата.

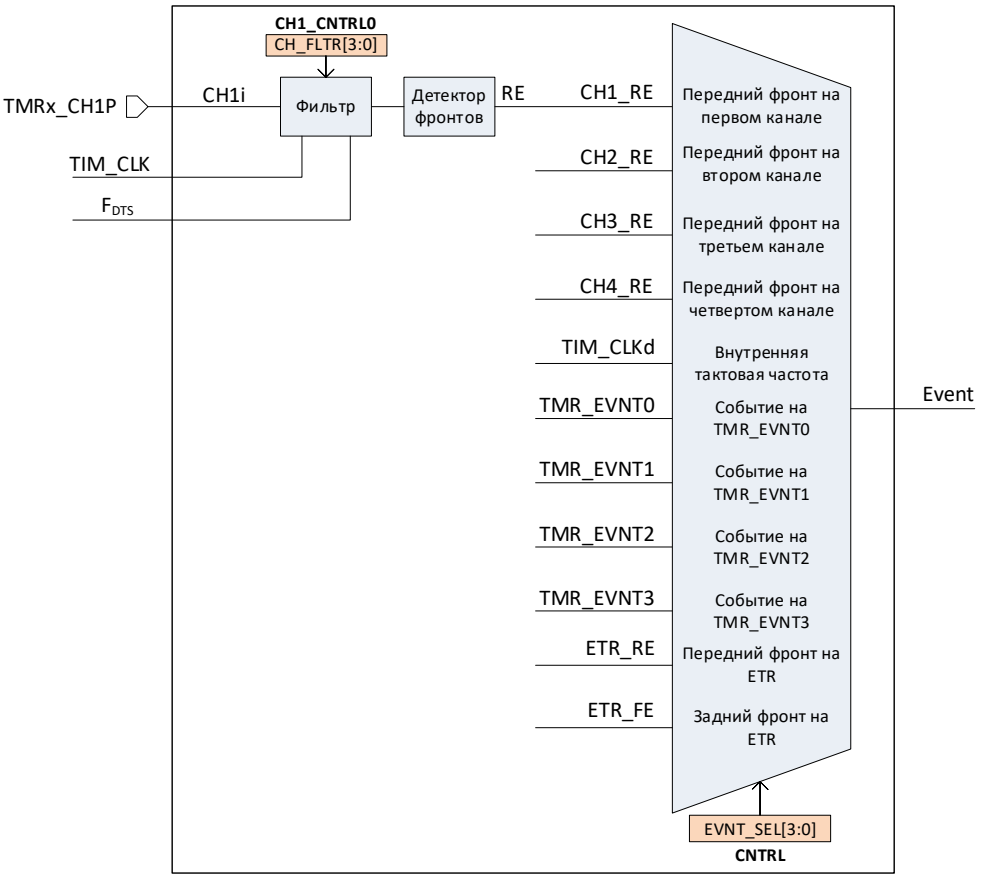



Рисунок 59 – Схема тактирования сигналом со входа первого канала

Со входа CHyi внешний тактовый сигнал поступает в блок цифрового фильтра. Данный блок позволяет отфильтровать входной сигнал с целью устранения импульсов, длительность которых меньше заданного порога (см. «Блок цифрового фильтра»). Настройки фильтра задаются в поле CH_FLTR[3:0] регистра CHy_CNTRL0.

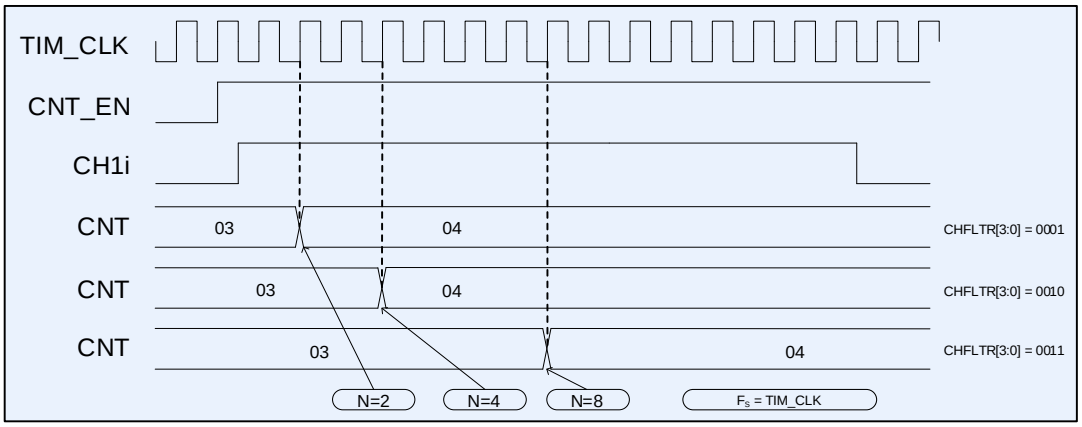



Рисунок 60 – Диаграмма внешнего тактирования с разными вариантами фильтра

#### 13.1.3.4 Внешний тактовый сигнал, «Режим 2»: событие переднего или заднего фронта на входе ETR

Данный режим выбирается, когда EVNT_SEL[3:0] = 100x и CNT_MODE[1:0] = 10 в регистре CNTRL. Основной счетчик таймера может тактироваться по переднему или по заднему фронту внешнего сигнала, поступающего на вход ETR, в зависимости от значения в поле EVNT_SEL[3:0].

Конфигурация тактового сигнала со входа ETR задается в регистре BRKETR_CNTRL. Бит ETR_INV позволяет установить инверсию входного сигнала. Поле ETR_PSC[1:0] задает коэффициент деления асинхронного предделителя внешней частоты (1, 2, 4 или 8). После предделителя тактовый сигнал поступает в блок цифрового фильтра, где он может быть дополнительно отфильтрован с целью устранения импульсов, длительность которых меньше заданного порога (см. «Блок цифрового фильтра»). Настройки фильтра задаются в поле ETR_FLTR[3:0].

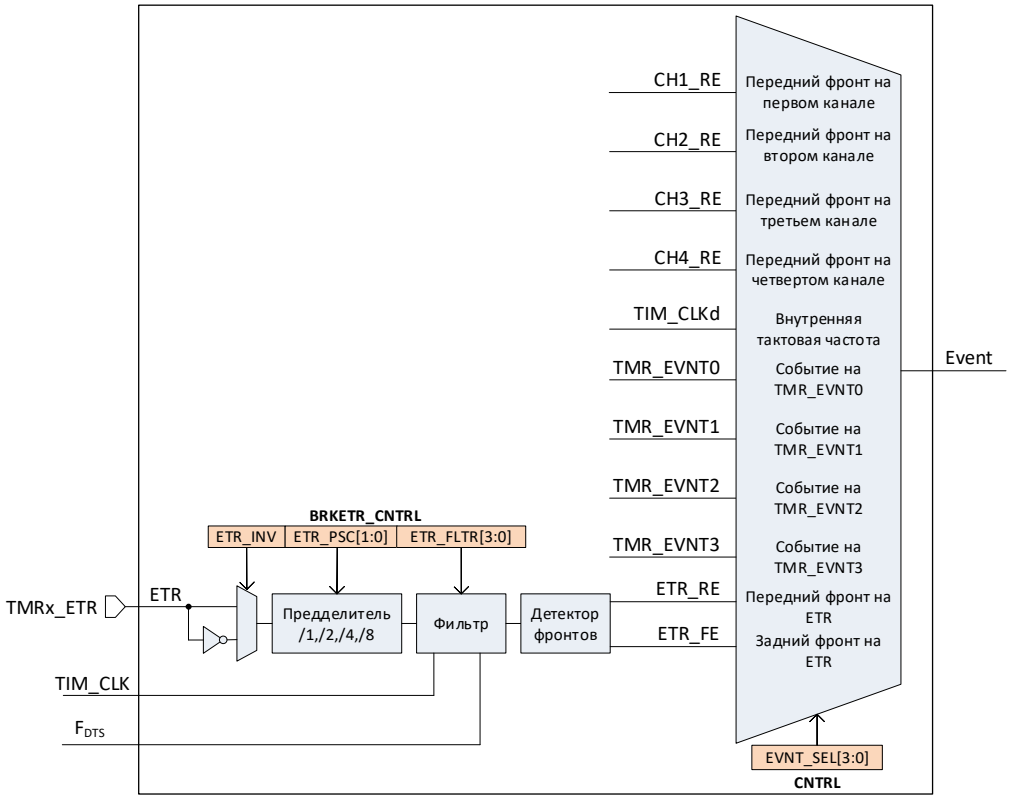



Рисунок 61 – Схема тактирования сигналом со входа ETR

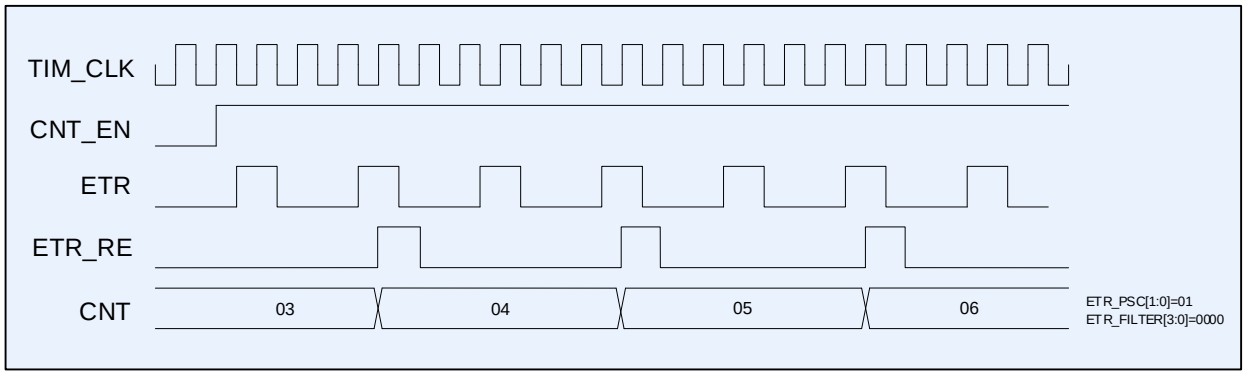



Рисунок 62 – Диаграмма тактирования со входа ETR, EVNT_SEL[3:0] = 1000

### 13.1.4 *Режим захвата*

Каждый канал таймера может быть независимо переведен в режим захвата. В режиме захвата по событию от внешнего входного сигнала происходит фиксация значения основного счетчика CNT в регистры CHy_CCR и CHy_CCR1. Регистрация событий осуществляется только по входам CHyi (выводы, настроенные в функцию TMRx_CHyP).

Структурная схема блока захвата представлена на рисунке 63.

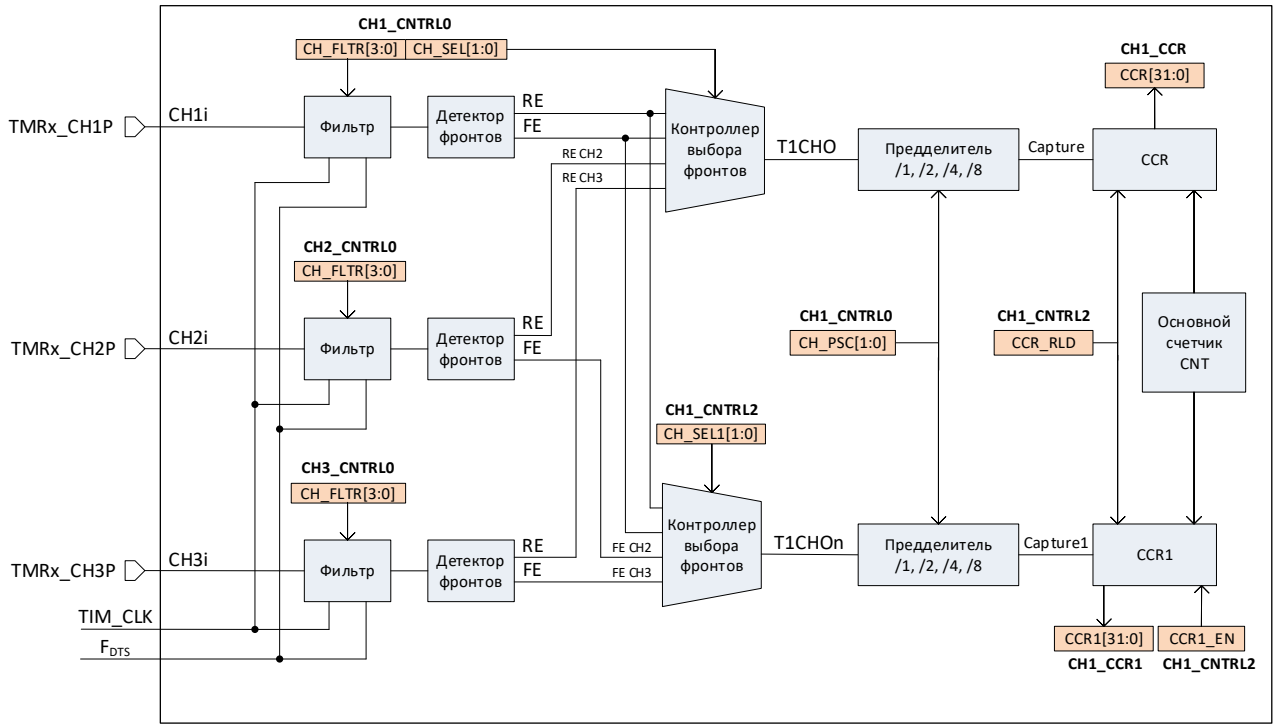



Рисунок 63 – Структурная схема блока захвата на примере канала 1

Для включения режима захвата для определенного канала необходимо записать «1» в бит CAP_NPWM регистра управления каналом CHy_CNTRL0. Для использования регистра CHy_CCR1 необходимо записать «1» в бит CCR1_EN регистра CHy_CNTRL2.

Внешний сигнал со входа CHyi сначала поступает в блок фильтра. Данный блок позволяет отфильтровать входной сигнал с целью устранения импульсов, длительность которых меньше заданного порога (см. «Блок цифрового фильтра»). Настройки фильтра задаются в поле CH_FLTR[3:0] регистра CHy_CNTRL0.

Сигнал с блока фильтра поступает в блок «Детектор фронтов». При обнаружении положительного фронта входного сигнала данный блок вырабатывает сигнал RE, а при обнаружении отрицательного фронта входного сигнала – сигнал FE.

В блоке «Контроллер выбора фронтов» производится выбор используемого для захвата сигнала между положительным фронтом канала, отрицательным фронтом канала и положительными и отрицательными фронтами сигналов от других каналов. Настройка блока «Контроллер выбора фронтов» для регистра CHy_CCR осуществляется в поле CH_SEL[1:0] регистра CHy_CNTRL0, а для регистра CHy_CCR1 – в поле CH_SEL1[1:0] регистра CHy_CNTRL2. Выбранный для захвата сигнал поступает в предварительный делитель, который в зависимости от значения в поле CH_PSC[1:0] регистра CHy_CNTRL0 позволяет фиксировать все события, либо каждое второе, каждое четвертое или каждое восьмое событие.

Предварительный делитель для регистра CHy_CCR формирует сигнал Capture, а предварительный делитель для регистра CHy_CCR1 формирует сигнал Capture1. По сигналам Capture и Capture1 выполняется запись текущего значения основного счетчика CNT в регистры CHy_CCR и CHy_CCR1.

На рисунке 64 показан пример захвата значения основного счетчика CNT в регистр CHy_CCR по положительному фронту на входе канала, а в регистр CHy_CCR1 – по отрицательному фронту на входе канала.

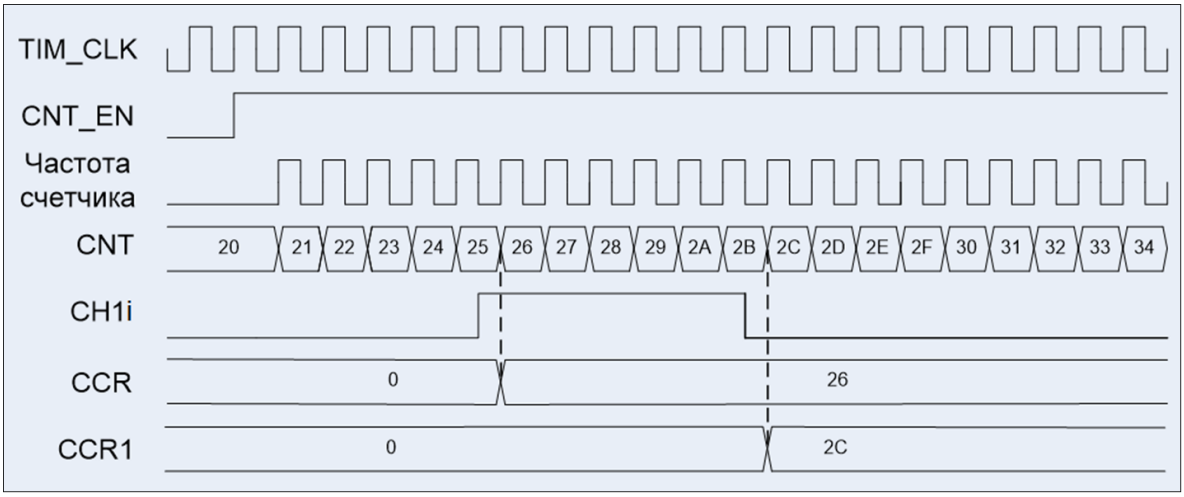



Рисунок 64 – Диаграмма захвата события со входа канала 1

По событию захвата на определенном канале в регистре IE можно разрешить выработку прерываний, а в регистрах DMA_RE0-DMA_RE4 можно разрешить формирование запросов DMA.

Между формированием события захвата и записью текущего значения основного счетчика CNT в регистры CHy_CCR и CHy_CCR1 может быть установлена задержка с помощью бита EVNT_DLY в регистре CHy_CNTRL2. Если бит EVNT_DLY равен «0», то сначала формируется событие захвата, а затем через один такт сигнала синхронизации TIM_CLK выполняется запись CNT в регистры CHy_CCR и CHy_CCR1. Если бит EVNT_DLY равен «1», то обновление информации в регистрах CHy_CCR и CHy_CCR1 происходит синхронно с событием захвата. При реализации чтения регистров CHy_CCR и CHy_CCR1 по событию захвата рекомендуется устанавливать бит EVNT_DLY в «1».

### 13.1.5 *Режим ШИМ*

Каждый канал таймера может быть независимо переведен в режим ШИМ для формирования выходных сигналов с возможностью задания «мертвой зоны». Структурная схема блока формирования ШИМ представлена на рисунке 65.

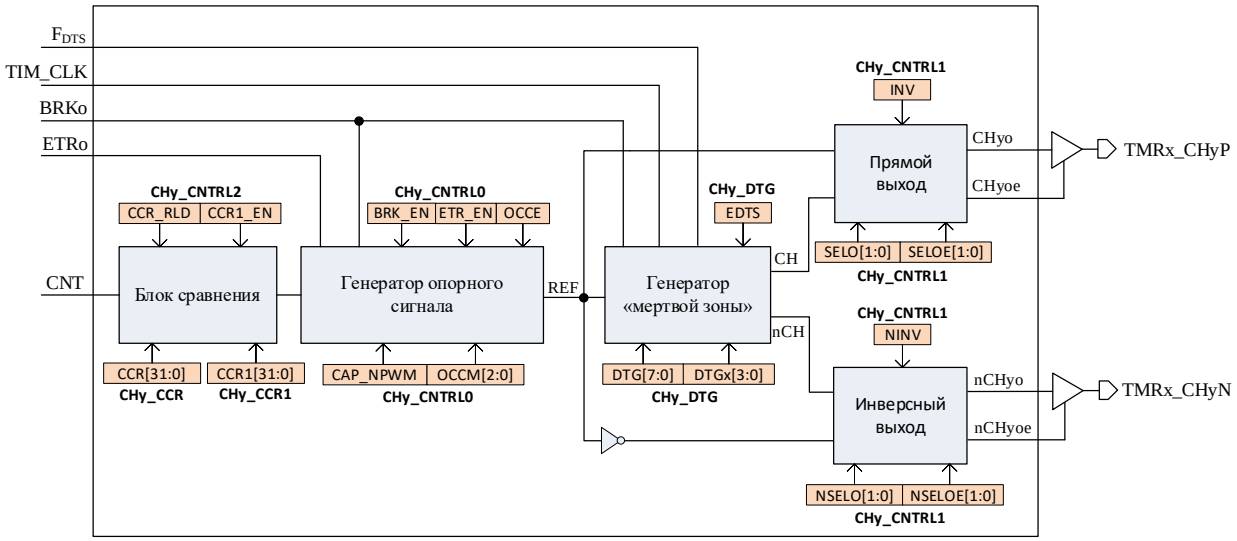



Рисунок 65 – Структурная схема блока формирования ШИМ

Для включения режима ШИМ для определенного канала необходимо в регистре управления каналом CHy_CNTRL записать «0» в бит CAP_NPWM.

#### 13.1.5.1 Генератор опорного сигнала REF

При работе в режиме ШИМ блок генератора опорного сигнала формирует сигнал REF. Данный сигнал формируется на основании сравнения значения в регистрах CHy_CCR, CHy_CCR1 и основного счетчика CNT. Формат выработки сигнала REF устанавливается в регистре управления каналом таймера CHy_CNTRL0, поле OCCM[2:0].

Если в регистре CHy_CNTRL2 бит CCR1_EN = 0, то для формирования сигнала REF используется только результат сравнения значения в регистре CHy_CCR и основного счетчика CNT.

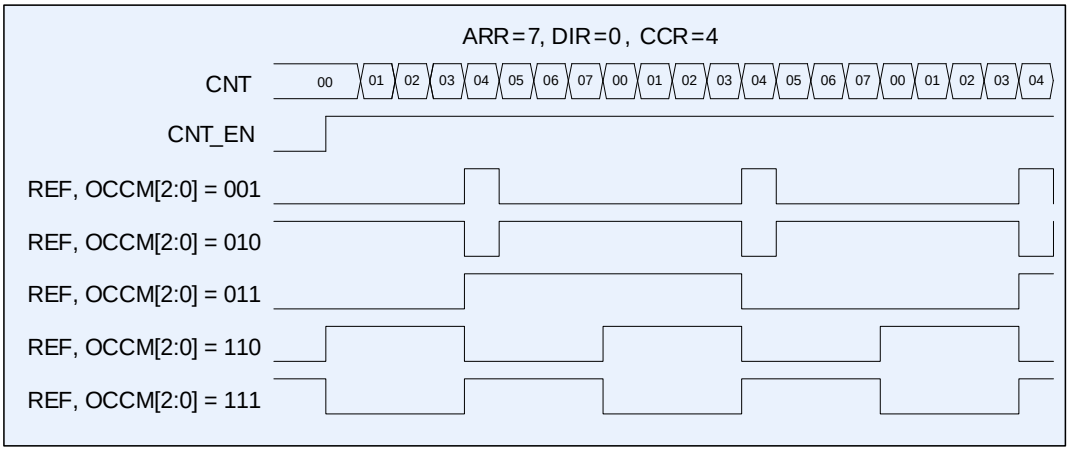



Рисунок 66 – Диаграмма работы в режиме ШИМ, CCR1_EN=0

Если в регистре CHy_CNTRL2 бит CCR1_EN = 1, то для формирования сигнала REF задействуются оба результата сравнения значения в регистрах CHy_CCR, CHy_CCR1 и основного счетчика CNT.

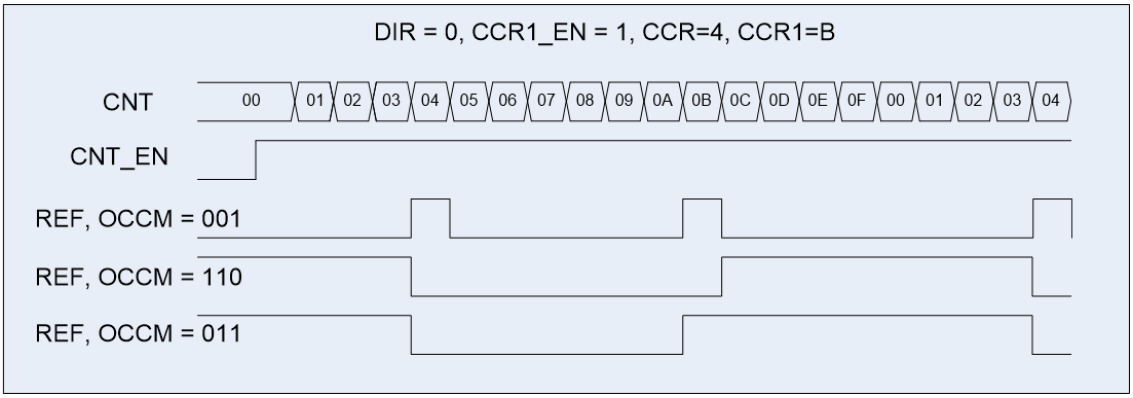



Рисунок 67 – Диаграмма работы в режиме ШИМ, CCR1_EN = 1

Запись новых значений в регистры CHy_CCR и CHy_CCR1 осуществляется немедленно, если в регистре CHy_CNTRL2 бит CCR_RLD установлен в «0», иначе регистры CHy_CCR и CHy_CCR1 получат новые значения только при CNT == 0. Процесс обновления значений в регистрах CHy_CCR и CHy_CCR1 обозначается в регистре CHy_CNTRL0 с помощью флагов WR_CMPL и WR_CMPL1, соответственно. На время выполнения записи флаг WR_CMPL/WR_CMPL1 устанавливается в «1», по окончании записи флаг WR_CMPL/WR_CMPL1 сбрасывается в «0».

Сигнал REF может быть принудительно установлен в «0» с использованием внешнего сигнала сброса, поступающего со входа ETR (высокий активный уровень) или со входа BRK (низкий активный уровень). Активный уровень на входах ETR и BRK может быть изменён с помощью инверсии входного сигнала, регистр BRKETR_CNTRL, биты ETR_INV и BRK_INV, соответственно.

Для разрешения сброса сигнала REF по входу ETR необходимо установить бит ETR_EN и OCCE в регистре CHy_CNTRL0. Активный уровень на входе ETR сбрасывает сигнал REF в «0». После снятия активного уровня на входе ETR сигнал REF остается в «0» до следующего события установки REF в «1», рисунок 68.

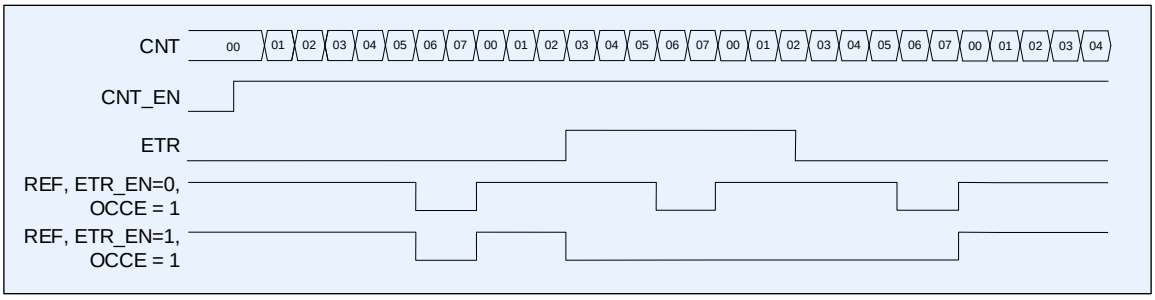



Рисунок 68 – Диаграмма сброса сигнала REF по выводу ETR

Для разрешения сброса сигнала REF по входу BRK необходимо установить бит BRK_EN в регистре CHy_CNTRL0. Активный уровень на входе BRK сбрасывает сигнал REF в «0» путем маскирования. После снятия активного уровня на входе BRK генерация сигнала REF сразу же восстанавливается, рисунок 69.

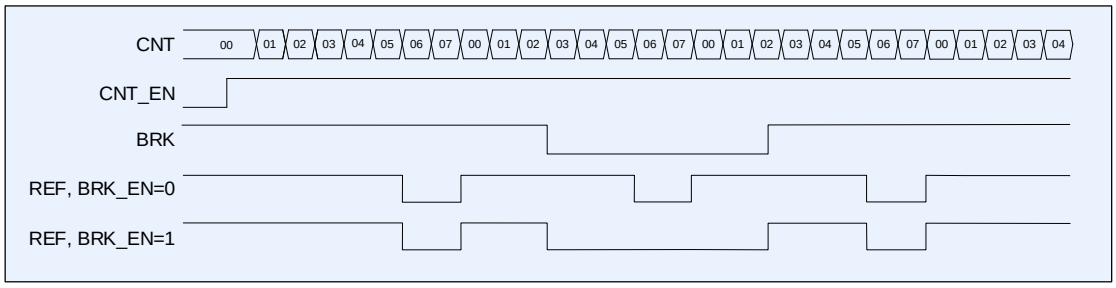



Рисунок 69 – Диаграмма сброса сигнала REF по выводу BRK

#### 13.1.5.2 Генератор «мертвой зоны»

Блок генератора «мертвой зоны» (dead-time generator, DTG) позволяет на основе сигнала REF формировать комплементарную пару сигналов с «мертвой зоной». Выходные сигналы с блока DTG передаются на выходные блоки следующим образом:

- сигнал на прямом выходе (CHyo, CHyoe) представляет собой инвертированный сигнал REF, в котором передний фронт задержан на величину DTGdel относительно заднего фронта опорного сигнала REF;
- сигнал на инверсном выходе (nCHyo, nCHyoe) представляет собой сигнал REF, в котором передний фронт задержан на величину DTGdel относительно переднего фронта опорного сигнала REF.

Значение «мертвой зоны» между сигналами на прямом и инверсном выходах рассчитывается в тактах частоты TIM_CLK или FDTS по формуле

$$\mathrm{DTGdel} = \mathrm{DTG} \cdot (\mathrm{DTGx} + 1), \quad (9)$$

где DTGx – предварительный делитель частоты;

DTG – основной делитель частоты.

Управление блоком DTG осуществляется через регистр CHy_DTG. Выбор источника тактирования для задания «мертвой зоны» задается битом EDTS. Значения делителей DTGx и DTG задаются в полях DTGx[3:0] и DTG[7:0], соответственно. Если задержка DTGdel больше ширины импульса высокого уровня, то соответствующий импульс не генерируется.

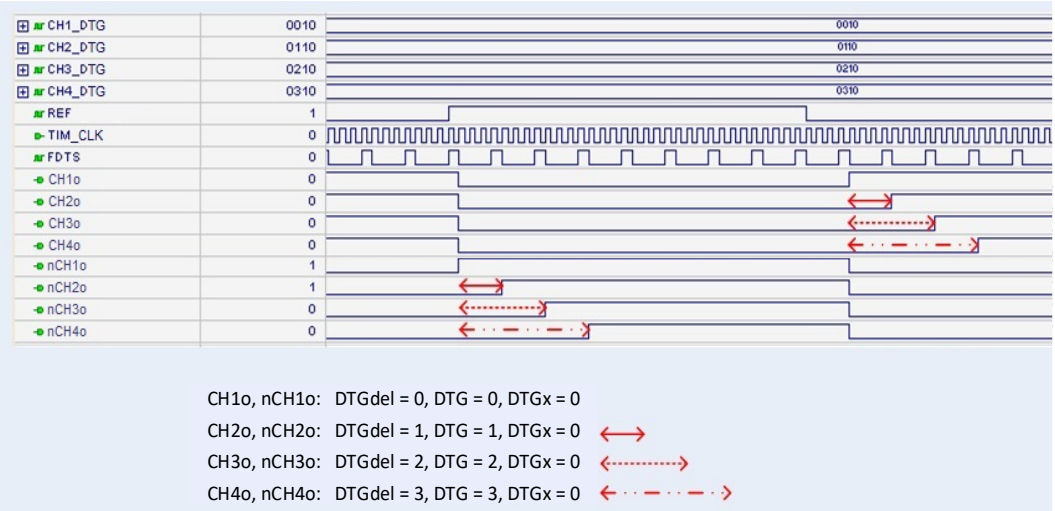



Рисунок 70 – Диаграмма работы блока DTG

Выходные сигналы блока DTG могут быть принудительно установлены в «0» с использованием внешнего сигнала сброса, поступающего со входа BRK (низкий активный уровень). Активный уровень на входе BRK может быть изменён с помощью инверсии входного сигнала, бит BRK_INV в регистре BRKETR_CNTRL. Для разрешения сброса выходных сигналов блока DTG по входу BRK необходимо установить бит BRK_EN в регистре CHy_CNTRL0. Активный уровень на входе BRK сбрасывает выходные сигналы блока DTG в «0» путем их маскирования.

#### 13.1.5.3 Выходные блоки

Каждый канал таймера имеет два выходных блока – прямой и инверсный. Каждый выходной блок формирует как сигнал выдачи (CHyo, nCHyo), так и сигнал разрешения выдачи (CHyoe, nCHyoe). В качестве сигналов для прямого (CHyo, CHyoe) и инверсного (nCHyo, nCHyoe) выходов в регистре CHy_CNTRL1 могут быть заданы либо постоянные уровни (0 или 1), либо сигналы, формируемые на основе сигнала REF. К таким сигналам относится сам сигнал REF, а также сигналы, формируемые блоком DTG.

Выбор источника сигнала выдачи для прямого (CHyo) и инверсного (nCHyo) выходов задается в полях SELO[1:0] и NSELO[1:0] регистра CHy_CNTRL1. Дополнительно каждый сигнал выдачи для прямого (CHyo) и инверсного (nCHyo) выходов может быть инвертирован путем установки битов INV и NINV в регистре CHy_CNTRL1. Выбор источника сигнала разрешения выдачи для прямого (CHyoe) и инверсного (nCHyoe) выходов задается в полях SELOE[1:0] и NSELOE[1:0] регистра CHy_CNTRL1. При этом, если сигнал разрешения выдачи равен «0», то соответствующий вывод работает в режиме входа, если сигнал разрешения выдачи равен «1» – то в режиме выхода.

### 13.1.6 *Блок цифрового фильтра*

В тракте входа ETR и входов каналов таймера CHyi предусмотрен блок цифрового фильтра, который позволяет исключить из входного сигнала импульсы высокого и низкого уровня, длительность которых меньше заданного порога.

Конфигурация фильтра для входа ETR выполняется в поле ETR_FLTR[3:0] регистра BRKETR_CNTRL, для входов каналов CHyi – в поле CH_FLTR[3:0] регистра CHy_CNTRL0. Значение в данных полях позволяет настроить два параметра фильтра:

- частота выборки $F_S$, на которой входной сигнал захватывается в сдвиговый регистр для накопления. В качестве частоты $F_S$ может использоваться частота TIM_CLK или $F_{DTS}$;
- количество выборок (длина фильтра) N, на протяжении которых входной сигнал должен оставаться стабильным, чтобы не подвергнуться фильтрации.

Если в течение заданного количества выборок N на частоте $F_S$ входной сигнал не изменяется, то значение входного сигнала передаётся на выход фильтра. Иначе внутренний счетчик накопления сбрасывается и захват сигнала начинается заново.

Таким образом, настраивая частоту $F_S$ и количество выборок N, задается минимальная длительность импульсов входного сигнала, которые не будут отфильтрованы. Диаграмма работы фильтра при использовании частоты TIM_CLK приведена на рисунке 71, частота $F_S$ = TIM_CLK, количество выборок N = 4.

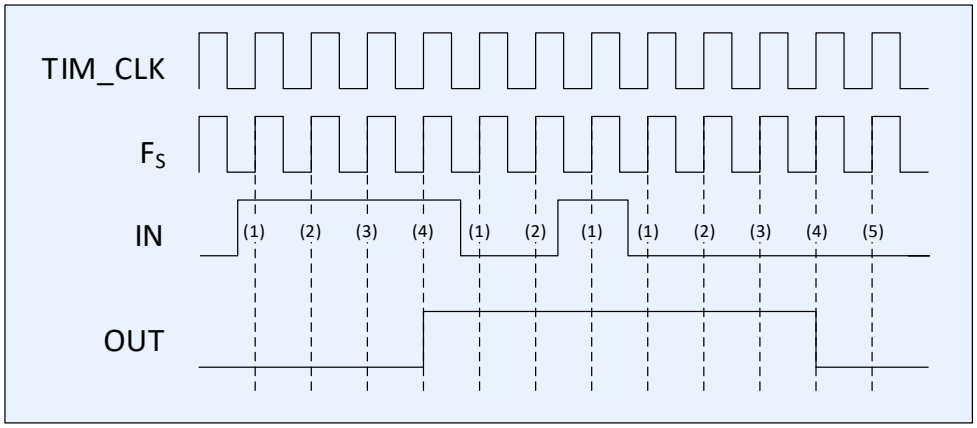



Рисунок 71 – Диаграмма работы фильтра, $F_S$ = TIM_CLK, N = 4

Для задания длительных интервалов накопления входного сигнала имеется возможность использовать частоту $F_{DTS}$, которая формируется из частоты TIM_CLK путём прореживания на заданный коэффициент (см. «Тактовая частота $F_{DTS}$»).

Диаграмма работы фильтра при использовании частоты $F_{DTS}$ приведена на рисунке 72, частота выборки $F_S = F_{DTS}/2$, количество выборок N = 6.

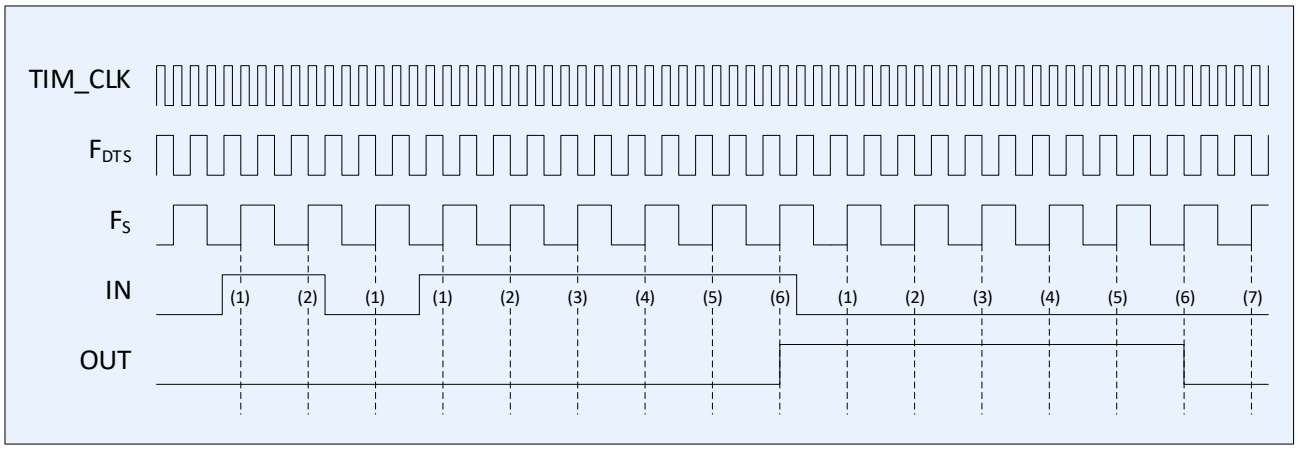



Рисунок 72 – Диаграмма работы фильтра, $F_S = F_{DTS}/2$, N = 6

Возможные варианты настройки блока фильтра приведены в таблице 57.

Таблица 57 – Возможные конфигурации фильтра

| CH_FLTR[3:0], ETR_FLTR[3:0] | Частота выборки Fs | Количество выборок N | Минимальная длительность импульсов, которые не будут отфильтрованы |
|---|---|---|---|
| 0000 | $F_{DTS}$ | 1 | - |
| 0001 | TIM_CLK | 2 | 2 x $T_{TIM_CLK}$ |
| 0010 | TIM_CLK | 4 | 4 x $T_{TIM_CLK}$ |
| 0011 | TIM_CLK | 8 | 8 x $T_{TIM_CLK}$ |
| 0100 | $F_{DTS}/2$ | 6 | 12 x $T_{FDTS}$ |
| 0101 | $F_{DTS}/2$ | 8 | 16 x $T_{FDTS}$ |
| 0110 | $F_{DTS}/4$ | 6 | 24 x $T_{FDTS}$ |
| 0111 | $F_{DTS}/4$ | 8 | 32 x $T_{FDTS}$ |
| 1000 | $F_{DTS}/8$ | 6 | 48 x $T_{FDTS}$ |
| 1001 | $F_{DTS}/8$ | 8 | 64 x $T_{FDTS}$ |
| 1010 | $F_{DTS}/16$ | 5 | 80 x $T_{FDTS}$ |
| 1011 | $F_{DTS}/16$ | 6 | 96 x $T_{FDTS}$ |
| 1100 | $F_{DTS}/16$ | 8 | 128 x $T_{FDTS}$ |
| 1101 | $F_{DTS}/32$ | 5 | 160 x $T_{FDTS}$ |
| 1110 | $F_{DTS}/32$ | 6 | 192 x $T_{FDTS}$ |
| 1111 | $F_{DTS}/32$ | 8 | 256 x $T_{FDTS}$ |

### 13.1.7 *Флаги состояний, прерывания и запросы DMA*

В процессе работы блок таймера отслеживает состояние внутренних блоков и формирует 17 событий:

- CNT_ZERO_EVNT – совпадение значения счетчика CNT с нулем;
- CNT_ARR_EVNT – совпадение значения счетчика CNT со значением в регистре ARR;
- ETR_RE_EVNT – фиксация переднего фронта на входе ETR;
- ETR_FE_EVNT – фиксация заднего фронта на входе ETR;
- BRK_EVNT – фиксация высокого уровня на входе BRK;
- CCR_CAP_EVNT[3:0] – запись значения счетчика CNT в регистр CHy_CCR по захвату настроенного фронта на входе канала CHyi, события формируются индивидуально для каждого канала;
- CCR_REF_EVNT[3:0] – фиксация переднего фронта на выходе генератора опорного сигнала REF, события формируются индивидуально для каждого канала;
- CCR_CAP1_EVNT[3:0] – запись значения счетчика CNT в регистр CHy_CCR1 по захвату настроенного фронта на входе канала CHyi, события формируются индивидуально для каждого канала.

#### 13.1.7.1 Флаги состояний

При возникновении события устанавливается соответствующий флаг в регистре STATUS. Сброс флагов в регистре STATUS осуществляется записью «0», запись «1» не оказывает влияния. Если запись «0» выполняется одновременно с новым событием, то приоритет у нового события.

#### 13.1.7.2 Прерывания

Блок таймера на основе флагов в регистре STATUS формирует один общий сигнал запроса прерывания INT_TMRx. Выбор флагов, формирующих запрос прерывания, осуществляется через регистр разрешения прерываний IE. При формировании запроса прерывания маскированные состояния флагов из регистра STATUS объединяются по схеме ИЛИ.

#### 13.1.7.3 Запросы DMA

На основе отслеживаемых событий блок таймера формирует сигналы запросов DMA TMRx[4:0]. Выбор событий, формирующих запрос DMA, осуществляется через регистры DMA_RE0-DMA_RE4. Данные регистры имеют одинаковые поля, при этом

каждый из регистров отвечает за конфигурацию индивидуального запроса от таймера к соответствующему каналу DMA.

Запросы TMRx[4:0] не привязаны к определенным каналам таймера, поэтому в полях CCR_CAP_EVNT_RE[3:0], CCR_REF_EVNT_RE[3:0] и CCR_CAP1_EVNT_RE[3:0] регистров DMA_RE0-DMA_RE4 может быть выбрано событие от любого канала таймера. Например, событие настроенного фронта на входе CH3i третьего канала таймера (CCR_CAP_EVNT_RE[3:0]=0x4) может вызвать запрос TMRx[1] при конфигурации регистра DMA_RE1 = 0x80.

### 13.1.8 *Примеры*

В данном разделе приведены примеры инициализации таймера 0 в различных режимах работы. Для других таймеров инициализация выполняется аналогично.

#### 13.1.8.1 Обычный счетчик

```
CLK_CNTR->PER1_CLK |= 0x00000001;          //Разрешение частоты PCLK для таймера 0
CLK_CNTR->TIM0_CLK = 0x00010000; //Настройка и подача частоты TIM_CLK
                                          //для таймера 0 (TIM_CLK = MAX_CLK)
TIMER0_CNTR->CNTRL = 0x00000000;  //Режим инициализации таймера
//Настраиваем работу основного счетчика
TIMER0_CNTR->CNT = 0x00000000;    //Начальное значение счетчика
TIMER0_CNTR->PSG = 0x00000000;    //Предделитель частоты TIM_CLK
TIMER0_CNTR->ARR = 0x0000000F;    //Основание счета
TIMER0_CNTR->IE = 0x00000002;     //Разрешение генерировать прерывание при CNT = ARR
//Разрешение работы таймера
TIMER0_CNTR->CNTRL = 0x00000001;  //Счет прямой по TIM_CLK
```

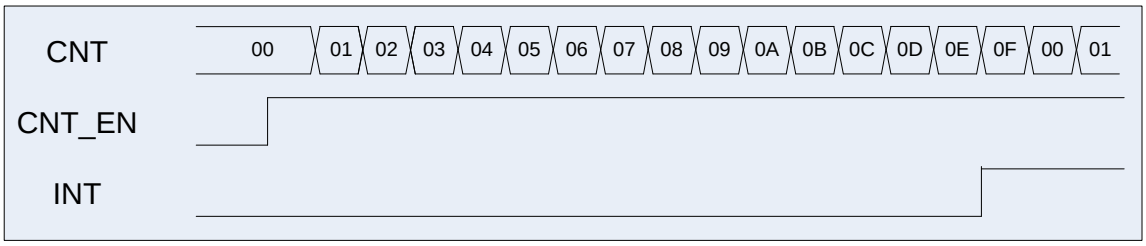



Рисунок 73 – Режим обычного счетчика

#### 13.1.8.2 Режим захвата

```
CLK_CNTR->PER1_CLK |= 0x00000001;          //Разрешение частоты PCLK для таймера 0
CLK_CNTR->TIM0_CLK = 0x00010000; //Настройка и подача частоты TIM_CLK
                                          //для таймера 0 (TIM_CLK = MAX_CLK)
TIMER0_CNTR->CNTRL = 0x00000000;  //Режим инициализации таймера
//Настраиваем работу основного счетчика
TIMER0_CNTR->CNT = 0x00000000;    //Начальное значение счетчика
TIMER0_CNTR->PSG = 0x00000000;    //Предделитель частоты TIM_CLK
TIMER0_CNTR->ARR = 0x000000FF;    //Основание счета
TIMER0_CNTR->IE = 0x000001E0;     //Разрешение генерировать прерывание по событию
                                          //настроенного фронта на входах CH1i-CH4i
TIMER0_CNTR->CH1_CNTRL0 = 0x00008000; //Захват по положительному фронту сигнала
```

```
                                                    //на входе CH1i, фильтрация
отключена
        TIMER0_CNTR->CH2_CNTRL0 = 0x00008010; //Захват по отрицательному
фронту сигнала
                                                    //на входе CH2i, фильтрация
отключена
        TIMER0_CNTR->CH3_CNTRL0 = 0x00008001; //Захват по положительному
фронту сигнала
                                                    //на входе CH3i, фильтрация
выполняется
                                                    //по 2 выборкам на частоте
TIM_CLK
        TIMER0_CNTR->CH4_CNTRL0 = 0x00008011; //Захват по отрицательному
фронту сигнала
                                                    //на входе CH4i, фильтрация
выполняется
                                                    //по 2 выборкам на частоте
TIM_CLK
        //Режим работы выхода канала – канал работает на вход
        TIMER0_CNTR->CH1_CNTRL1= 0x00000000;
        TIMER0_CNTR->CH2_CNTRL1= 0x00000000;
        TIMER0_CNTR->CH3_CNTRL1= 0x00000000;
        TIMER0_CNTR->CH4_CNTRL1= 0x00000000;
        //Разрешение работы таймера
        TIMER0_CNTR->CNTRL = 0x00000001;   //Счет прямой по TIM_CLK
```

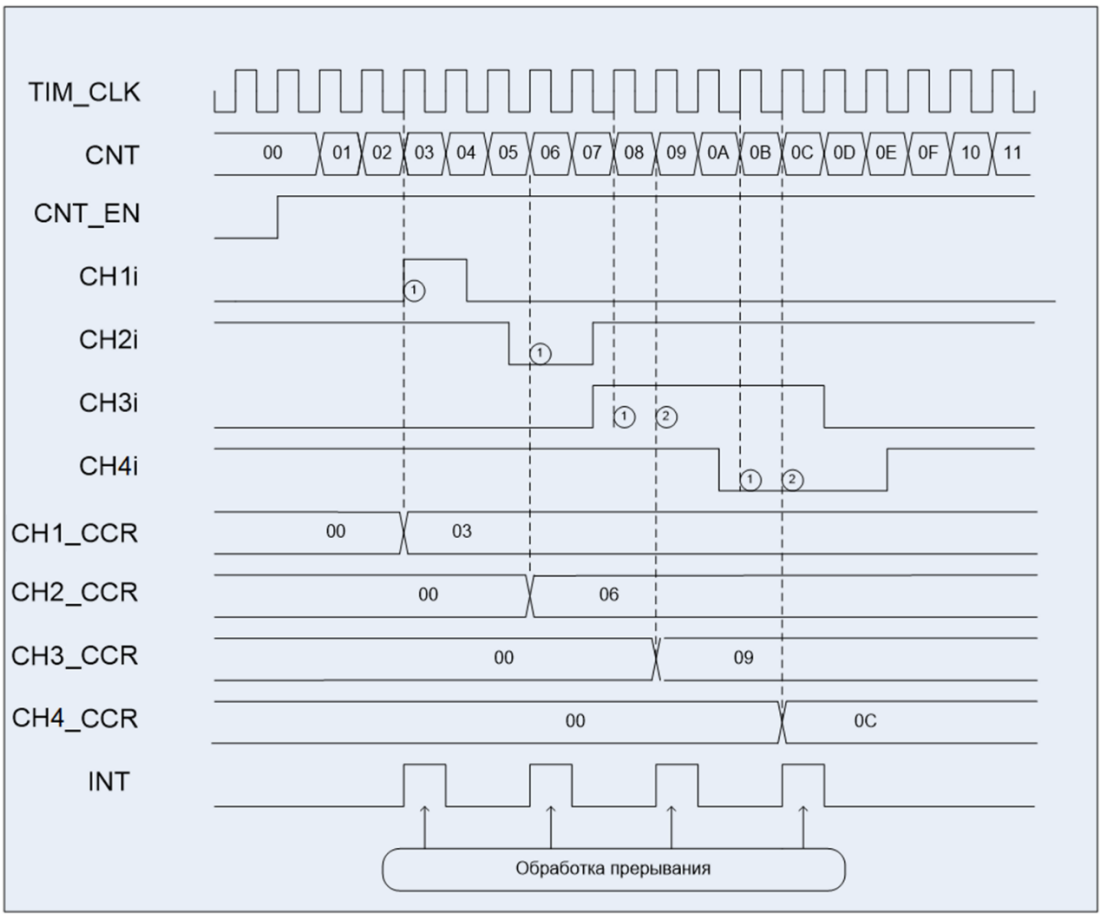



Рисунок 74 – Диаграммы примера работы в режиме захвата

#### 13.1.8.3 Режим ШИМ

```
CLK_CNTR->PER1_CLK |= 0x00000001;          //Разрешение частоты PCLK для таймера 0
CLK_CNTR->TIM0_CLK = 0x00010000; //Настройка и подача частоты TIM_CLK
                                                    //для таймера 0 (TIM_CLK = MAX_CLK)
TIMER0_CNTR->CNTRL = 0x00000000;  //Режим инициализации таймера
//Настраиваем работу основного счетчика
TIMER0_CNTR->CNT = 0x00000000;    //Начальное значение счетчика
TIMER0_CNTR->PSG = 0x00000000;    //Предделитель частоты TIM_CLK
TIMER0_CNTR->ARR = 0x00000010;    //Основание счета
TIMER0_CNTR->IE  = 0x00001E00;     //Разрешение генерировать прерывание по событию
                                                    //переднего фронта на выходе REF для всех каналов
//Режим работы каналов – ШИМ
TIMER0_CNTR->CH1_CNTRL0 = 0x00000200;  //REF = 1, если CNT == CCR
TIMER0_CNTR->CH2_CNTRL0 = 0x00000200;  //REF = 1, если CNT == CCR
TIMER0_CNTR->CH3_CNTRL0 = 0x00000400;  //REF = 0, если CNT == CCR
TIMER0_CNTR->CH4_CNTRL0 = 0x00000600;  //Переключение REF, если CNT == CCR
//Режим работы выхода канала – канал работает на выход,
//на выходы канала выдается сигнал REF
TIMER0_CNTR->CH1_CNTRL1= 0x00000909;
TIMER0_CNTR->CH2_CNTRL1= 0x00000909;
TIMER0_CNTR->CH3_CNTRL1= 0x00000909;
TIMER0_CNTR->CH4_CNTRL1= 0x00000909;

//Установка значений CCR, с которыми сравнивается CNT при работе в режиме ШИМ
TIMER0_CNTR->CH1_CCR = 0x00000003;
TIMER0_CNTR->CH2_CCR = 0x00000006;
TIMER0_CNTR->CH3_CCR = 0x00000009;
TIMER0_CNTR->CH4_CCR = 0x0000000F;
//Разрешение работы таймера
TIMER0_CNTR->CNTRL = 0x00000001; //Счет прямой по TIM_CLK
```

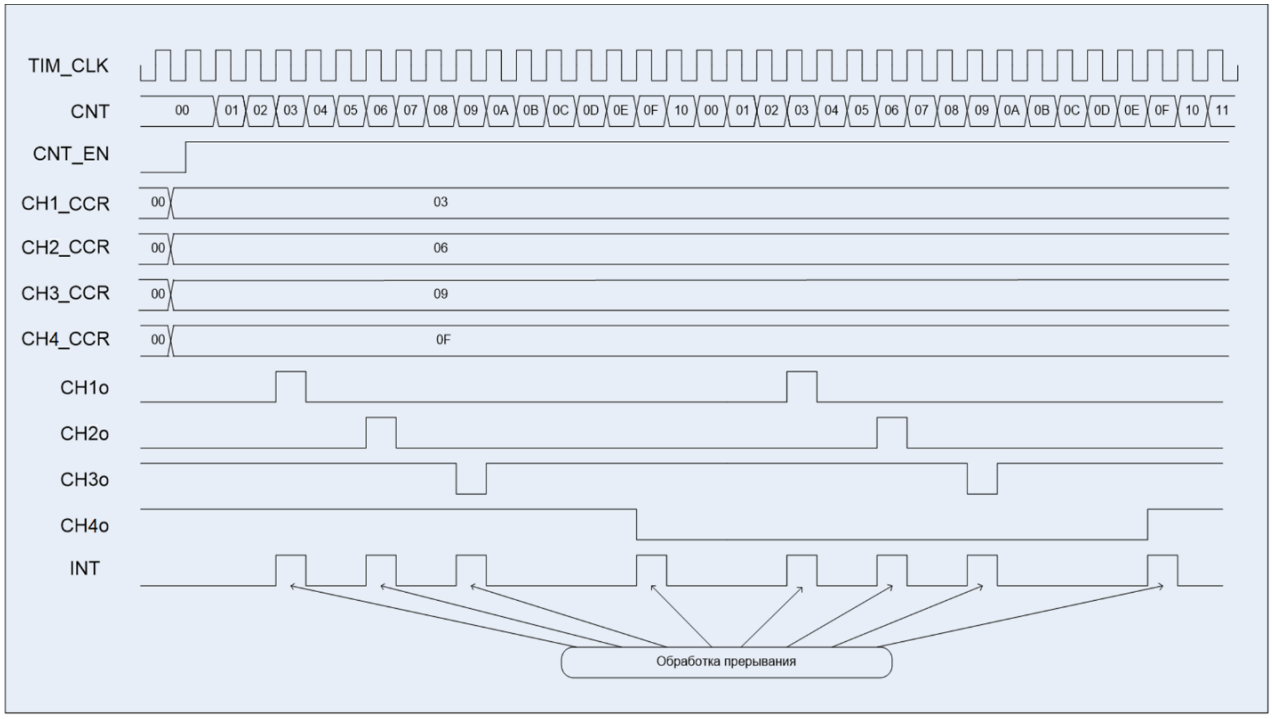



Рисунок 75 – Диаграммы примера работы в режиме ШИМ

## 13.2 Контролер ШИМ (EPWM_CNTR)

### 13.2.1 *Описание работы контроллера ШИМ*

#### 13.2.1.1 Общее описание и структурная схема блока

В основе модуля лежит 32-разрядный счетчик с гибкими возможностями управления счетом (вверх-вниз, вверх, вниз, изменение значения и направления счета по внешним событиям синхронизации), расположенный в блоке таймера основного счета.

В блоке сравнения происходит сравнение текущего значения счетчика с установленными значениями основания счета, нулем и единицей, по результатам которого в блоке анализа событий происходит формирование выходных импульсов модуля. Для выходных сигналов могут быть настроены параметры мертвой зоны и высокочастотной модуляции.

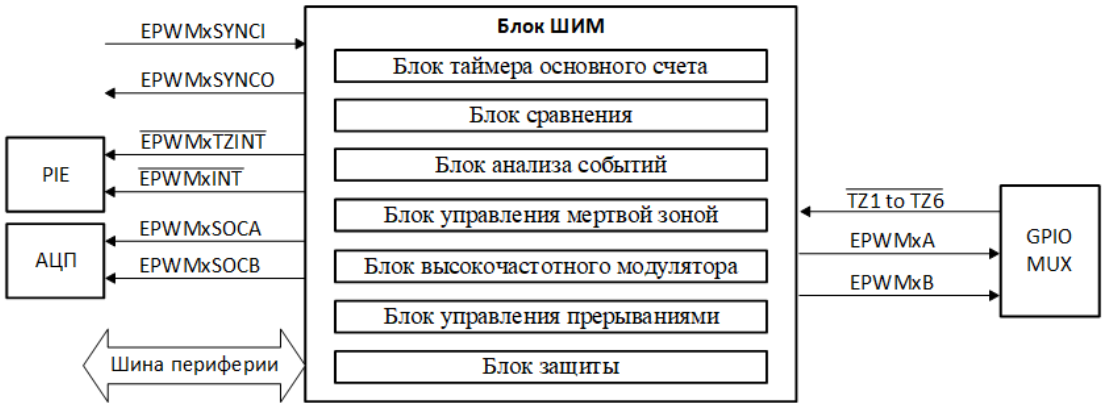



Рисунок 76 – Блок-схема модулятора

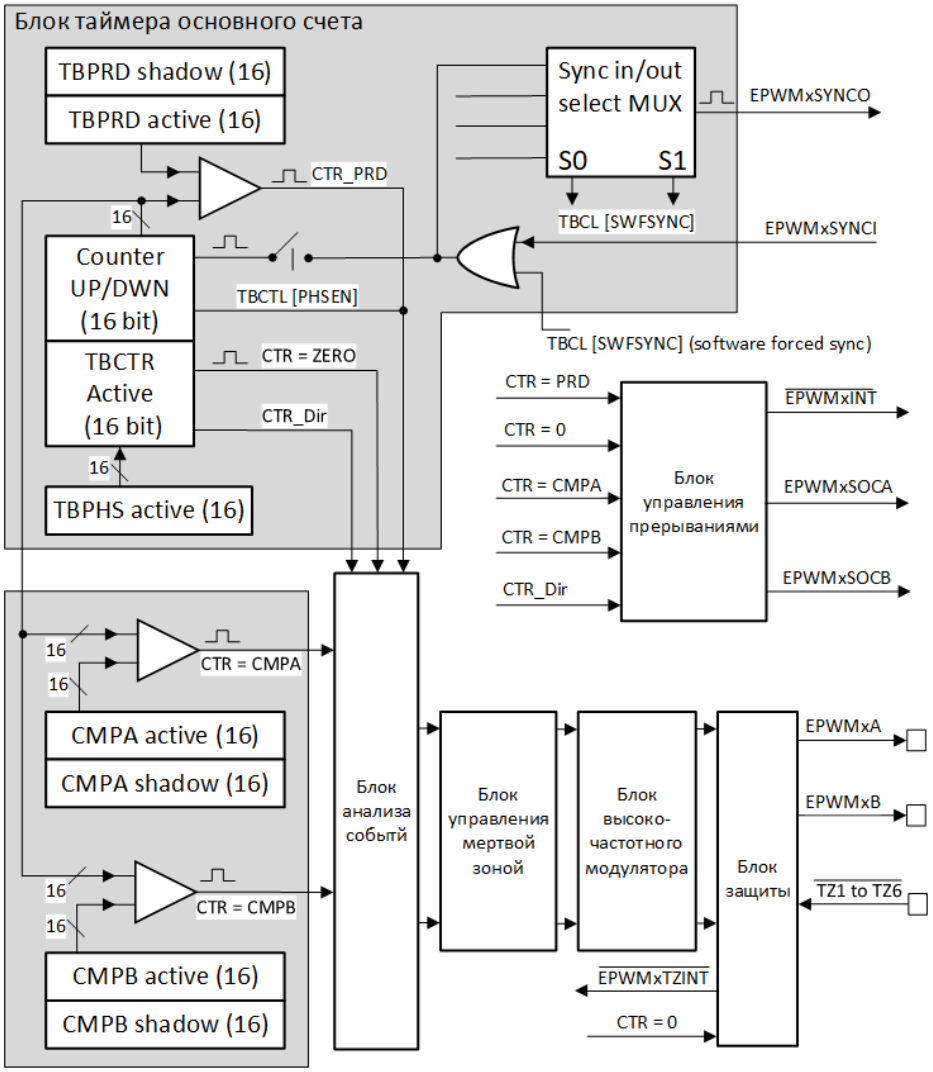



Рисунок 77 – Структурная схема ШИМ

#### 13.2.1.2 Блок таймера основного счета

Блок таймера основного счета содержит 32-разрядный счетчик, имеющий гибкие возможности настройки режима счета. На основе событий счетчика вырабатываются импульсы ШИМ EPWMxA и EPWMxB, событие синхронизации остальных блоков ШИМ EPWMxSYNCO, события синхронизации АЦП EPWMxSOCA и EPWMxSOCB и прерывания для АЦП.

Основными событиями блока являются:

- EPWMxSYNCi – внешнее входное событие синхронизации;
- EPWMxSYNCo – выходное событие синхронизации;
- CTR=PRD – Счетчик таймера TBCTR равен значению периода TBPRD;
- CTR=0 – Счетчик таймера TBCTR равен 0;
- CTR=CMPB и CTR=CMPA – Счетчик таймера TBCTR равен значению регистров сравнения CMPB или CMPA;
- смена направления счета;
- достижения счетчиком максимального значения (0xFFFFFFFF).

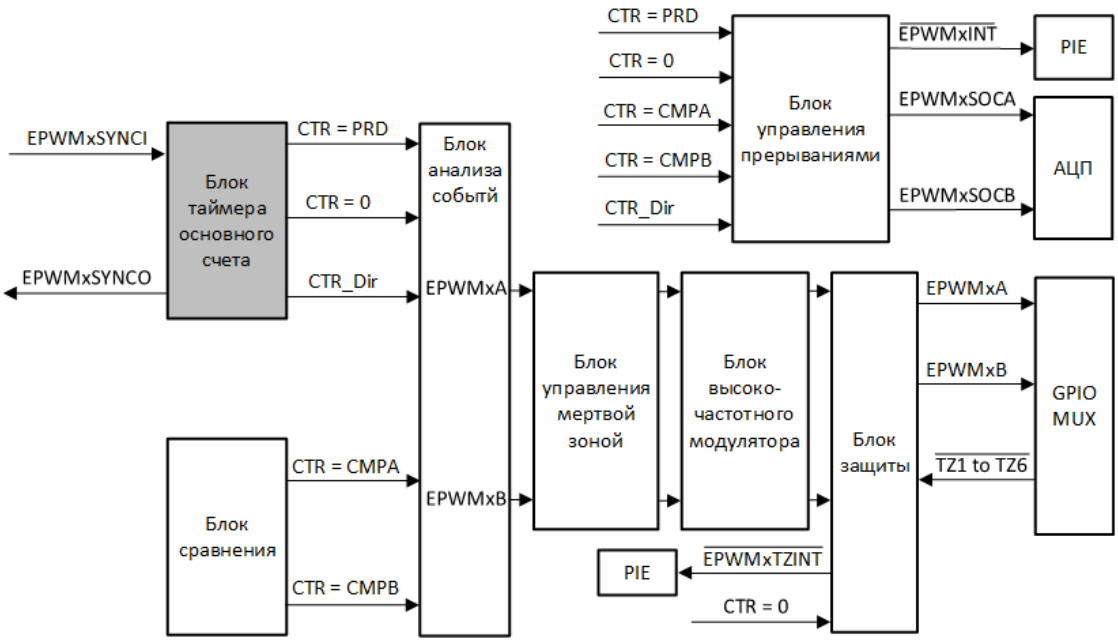



Рисунок 78 – Структурная схема блока таймера основного счета

Блок регистра счетчика таймера содержит пять регистров, доступных по шине APB. Основной 32-разрядный регистр счетчика таймера TBCTR доступен интерфейсу как для записи, так и для чтения.

В регистре TBPRD содержится значение периода счета таймера, значение которого таймер никогда не превышает.

В регистре TBPHS содержится значение, загружаемое в таймер по событию синхронизации, что позволяет синхронизировать по фазе значение таймера с блоками, стоящими ранее в цепочке синхронизации.

Общая настройка блока, управление режимом счета таймера, источники событий синхронизации блока, а также реакция блока на входные события синхронизации определяется управляющим регистром блока TBCTL.

Режим счета таймера определяется полем CTRMODE регистра TBCTL. Доступны четыре основных режима счета таймера: счет вверх, счет вниз, счет вверх-вниз и режим остановки счета.

В режиме счета вверх таймер TBCTR инкрементируется до достижения им значения периода ШИМ, содержащегося в регистре TBPRD, после чего принимает нулевое значение.

В режиме счета вниз таймер TBCTR декрементируется до достижения им значения нуля, после чего принимает значение периода ШИМ, содержащегося в регистре TBPRD, и начинает следующий цикл декремента.

В режиме счета вверх-вниз направление счета таймера TBCTR зависит от значения бита направления счета CTRDIR. В том случае, если бит CTRDIR равен 0, таймер декрементируется до достижения нулевого значения, после чего бит CTRDIR изменит свое состояние на 1, и таймер начнет инкрементировать до достижения им значения периода TBPRD. По достижению значения периода бит направления счета изменит свое состояние на 0, таймер начнет снова декрементироваться.

Алгоритм работы таймера, в том случае, если значение бита направления CTRDIR в момент перевода таймера в режим счета вверх-вниз было 1, будет аналогичным, за исключением того, что в начале счета таймер начнет считать вверх до достижения значения периода.

Значение бита CTRDIR можно узнать чтением бита CTRDIR регистра TBSTS. Прямое изменение бита направления счета со стороны процессора не доступно. Принудительно задать значение бита можно, задав режим счета таймера: вверх – бит будет установлен в 1, вниз – бит будет установлен в 0. Значение бита сохраняется при переходе к режиму счета вверх-вниз.

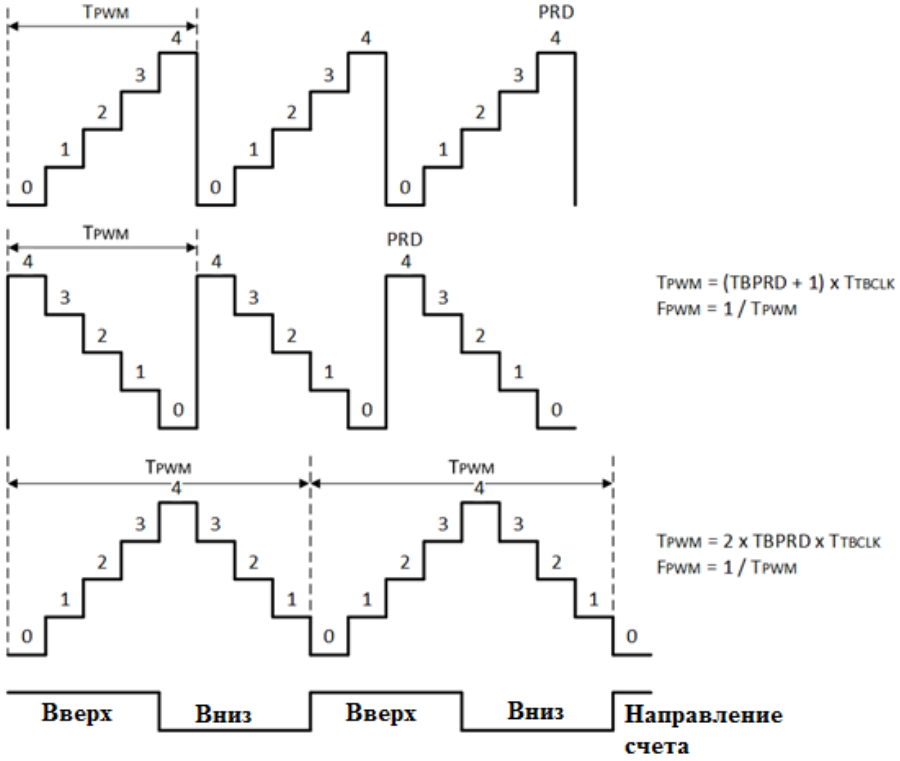



Рисунок 79 – Работа счетчика таймера в различных режимах:
а) Работа таймера в режиме счета вверх;
б) Работа таймера в режиме счета вниз;
в) Работа таймера в режиме счета вверх-вниз

Значение регистра периода счета CTRPRD доступно по шине APB в режиме прямого доступа или в режиме дублирования, в зависимости от бита PRDLD регистра CTRCTL. В том случае, если выбран режим дублирования, запись по шине APB происходит в регистр дублирования. Загрузка значения дублирующего регистра в основной регистр происходит при достижении таймером нулевого значения. В том случае, если запись нового значения в регистр совпадает с нулевым значением таймера, загрузка записываемого значения дублирующего регистра в основной произойдет по следующему событию достижения таймера нуля. По текущему событию произойдет загрузка текущего значения дублирующего регистра. В том случае, если выбран режим прямого доступа, запись по шине APB также приводит к записи в буфер, но значение из буфера выгружается

уже на следующем такте рабочей частоты синхросигнала. Новое значение будет оказывать воздействие на работу таймера уже в текущем периоде счета.

В регистре TBPHS содержится значение, загружаемое в основной регистр счетчика таймера по событию внешней синхронизации, для синхронизации фазы таймера с таймерами, стоящими ранее в цепочки синхронизации. При возникновении входного события синхронизации в том случае, если разрешено битом PHS_EN регистра управления TBCTL в регистр счетчика таймера загружается значение, содержащееся в регистре TBPHS.

Также по событию внешней синхронизации возможно изменение направления счета таймера на значение бита PHSDIR регистра TBCTR. Изменение направления счета таймера по событиям синхронизации возможно только в режиме счета вверх-вниз. Запретить изменение направления счета можно при помощи бита PHSDIR_EN.

В регистре TBSTS содержится информация о возникновении событий синхронизации, достижении счетчиком максимального возможного значения 0xFFFFFFFF и направлении счета основного счетчика таймера.

### *Управление событиями синхронизации*

Для синхронизации таймера с другими таймерами системы, а также, для более гибкого управления счетом таймера внутри периода, в блоке таймера реализованы входное, выходное и программное события синхронизации.

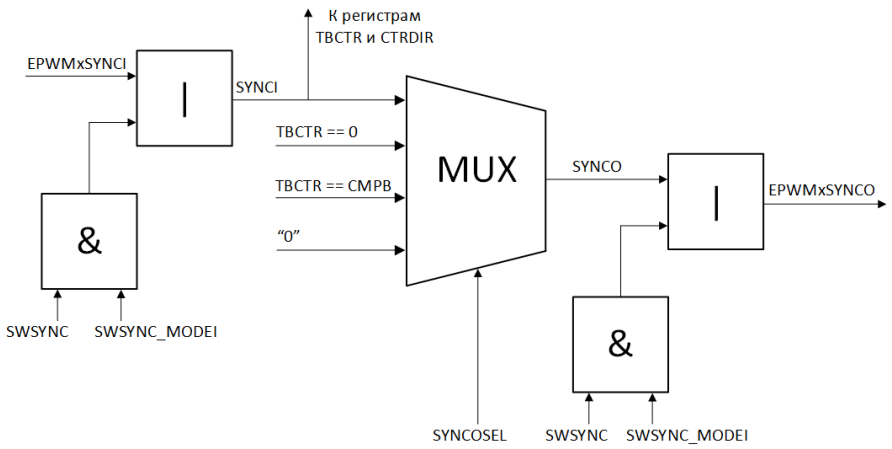



Рисунок 80 – Управление событиями синхронизации блока счетчика таймера

Входное событие синхронизации SYNCI в случае, если это разрешено битами PHSEN и PHSDIREN регистра TBCTL, управляет значением основного счетчика таймера и направлением счета. В качестве входного события синхронизации может быть либо событие на входе EPWMxSYNCI с блока таймера, стоящего ранее в цепочке синхронизации, либо программное событие синхронизации SWSYNC, если это разрешено SWSYNC_MODEI регистра TBCTL.

Выходным событием синхронизации SYNCO может быть в зависимости от поля SYNCOSEL входное событие синхронизации, событие равенства таймера основного счета нулю, событие достижения таймером значение сравнения CMPB (является входом из блока сравнения) либо выключено совсем. Также возможна программная установка выходного сигнала события синхронизации EPWMxSYNCO с помощью бита SWSYNC, если это разрешено битом SWSYNC_MODEI регистра TBCTL.

Входное событие синхронизации для блока PWM0 может быть выбрано с одного из выводов общего назначения в соответствии с таблицей назначения портов.

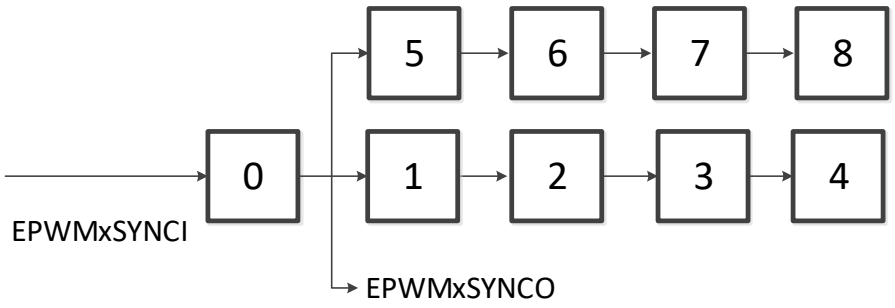



Рисунок 81 – Синхронизация блоков ШИМ

***Управление частотой счета таймера***

Перед началом работы с блоком необходимо разрешить частоту интерфейса APB PCLK ссоответствующего блока в регистре PER_CLK и задать опорную частоту счетчика ШИМ PWMCLK в регистре PWM_CLK блока управления тактовыми частотами микроконтроллера.

Управление частотой счета таймера производится в полях CLK_DIV и HSPCLKDIV регистра TBCTL путем задания соответствующих коэффициентов деления частоты опорной частоты PWMCLK.

#### 13.2.1.3 Блок сравнения

Основным назначением блока сравнения является сравнение значения основного таймера со значениями регистров сравнения CMPA, CMPB, в результате которого, вырабатываются сигналы CTReqCMPA и CTReqCMPB, которые, наряду с сигналами CTReqPRD, CTReqZERO и CTRDIR, передаются в блок анализа в качестве событий для управления выходами EPWMxA и EPWMxB.

Для задания значений сравнения доступны регистры CMPA и CMPB. Запись в регистры может производится как напрямую, так и через буфер глубиной восемь дискретных значений. Режим записи в регистры определяется битами SHDWAMODE и SHDWBMODE управляющего регистра CMPCTL. В том случае, если один из этих битов имеет единичное значение, запись осуществляется непосредственно в регистр и оказывает действие на работу блока по следующему за записью фронту, внутри текущего периода счета таймера. При этом возможно возникновение ситуации, в которой событие равенства по одному из значений CMPA, CMPB возникнет неоднократно внутри одного периода счета таймера, например, при счете вверх, если имеющееся в регистре значение меньше текущего значения таймера, а записываемое больше. Также возможна ситуация, в которой событие равенства внутри периода не возникнет вообще.

В том случае, если один из битов SHDWAMODE и SHDWBMODE имеет единичное значение, записываемое значение заносится в соответствующий буфер FIFO. В этом случае условие записи значения из буфера в регистр определяются полями LOADAMODE и LOADBMODE регистра CMPCTL. Доступны режимы записи по достижению таймером периода счета, нуля, либо по обоим этим условиям. Буфера FIFO для регистров CMPA и CMPB имеют глубину восемь значений, то есть возможна работа блока в режиме записи восьми значений раз в восемь периодов ШИМ. Для определения состояния буферов в регистре CMPCTL доступны биты опустошения и переполнения для каждого буфера SHDWAFULL, SHDWAEMPTY, SHDWBFULL и SHDWBEMPTY.

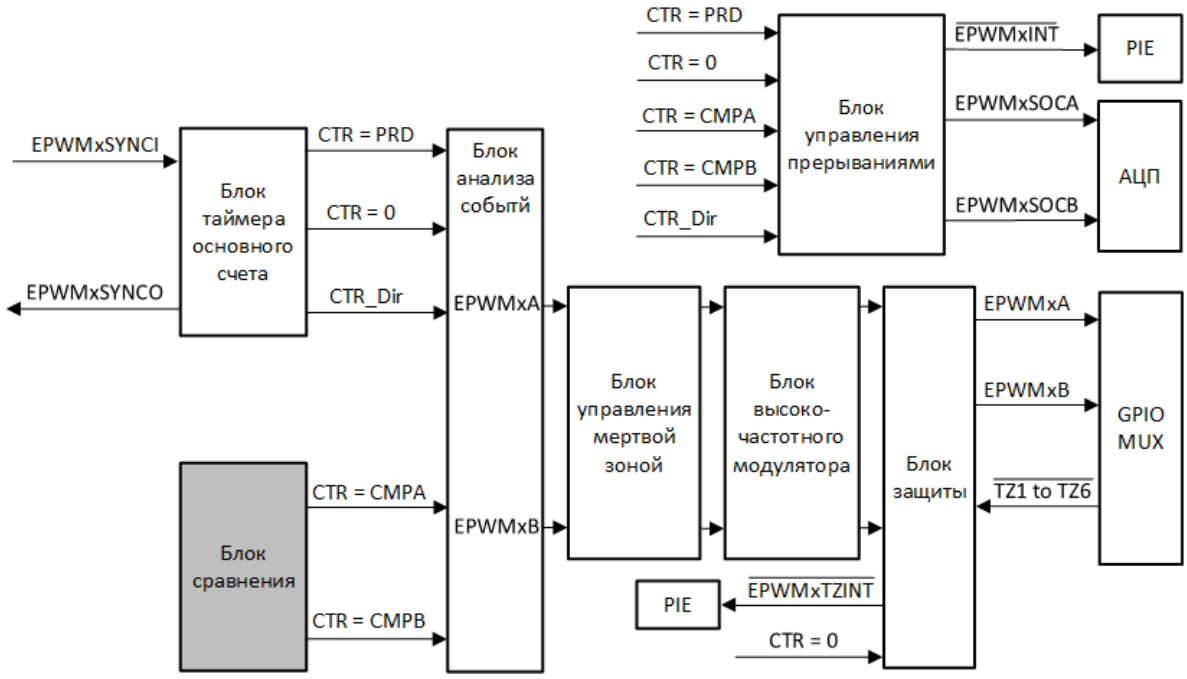



Рисунок 82 – Структурная схема блока сравнения

### 13.2.1.4 Блок анализа событий

Назначением блока анализа событий является формирование выходных сигналов ШИМ EPWMA и EPWMB в зависимости от значения управляющих регистров и событий на входе блока.

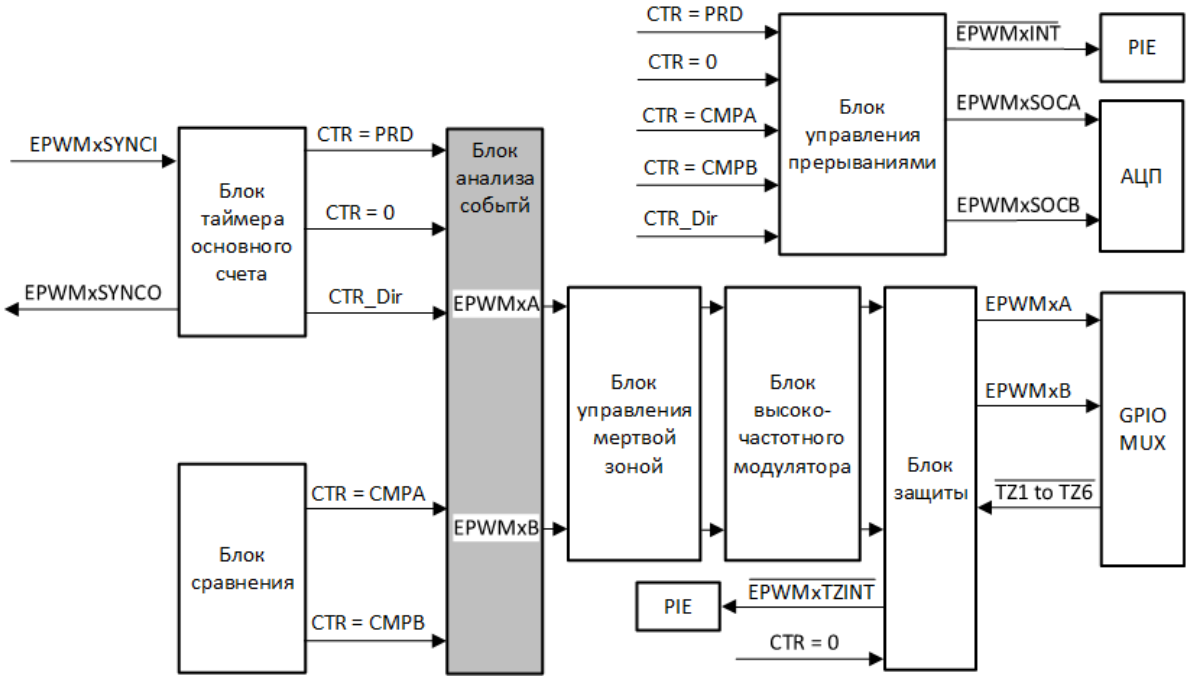



Рисунок 83 – Структурная схема блока анализа событий

Входными событиями для блока являются следующие события:

- CTReqPRD – достижение счетчиком периода счета;
- CTReqZERO – достижение счетчиком нуля;
- CTReqCMPA достижения счетчиком значения CMPA;
- CTReqCMPB достижения счетчиком значения CMPB.

Внутри блока входные события перераспределяются на восемь внутренних событий EVNT1_A, EVNT2_A, EVNT3_A, EVNT4_A и EVNT1_B, EVNT2_B, EVNT3_B, EVNT4_B, по четыре для каждого вывода. Распределение событий осуществляется назначением полей EVNT1_A_SEL, EVNT2_A_SEL, EVNT3_A_SEL, EVNT4_A_SEL, EVNT1_B_SEL, EVNT2_B_SEL, EVNT3_B_SEL, EVNT4_B_SEL регистров AQCTL и AQCTLB, для каждого из выводов EPWMxA и EPWMxB, соответственно.

В регистрах AQCTLA и AQCTLB определяется поведение выводов при возникновении одного из четырех событий и направления счета (CTRDIR), во время которого произошло событие. Таким образом, для каждого из выводов EPWMA и EPWMB возможно задание одного из четырех возможных действий, в одном из восьми случаев (четыре возможных события, два возможных направления счета во время возникновения каждого события).

В число возможных действий для каждого вывода входят:

- оставить прежнее значение вывода;
- задать на выводе «0»;
- задать на выводе «1»;
- задать значение, как функцию текущих значений выводов.

При конфигурации действия как функции текущих допустимы следующие варианты:

- текущее значение вывода (для EPWMA это EPWMA, для EPWMB это EPWMB);
- инверсию текущего значения вывода (для EPWMA это ~EPWMA);
- текущее значение другого вывода (для EPWMA это EPWMB, для EPWMB это EPWMA);
- инверсию текущего значения другого вывода (для EPWMA это ~EPWMB, для EPWMB это ~EPWMA).

Задать функцию для вывода можно при помощи полей I_MODEA и I_MODEB регистра AQSFRC.

Кроме автоматически назначаемых по событиям значений на выводы EPWMA и EPWMB возможно их ручное управление при помощи регистров AQSFRC и AQCSFRC.

Значения на выводы можно задавать в двух режимах: постоянное назначение константного значения вывода или назначения вывода до момента возникновения одного из событий.

Постоянное назначение константных значений осуществляется полями CSFA и CSFB регистра AQCSFRC для выводов EPWMA и EPWMB, соответственно. При задании значений «00» или «11» эти поля не оказывают влияние на выходы, заданием «01» или «10» можно установить выводы в константное состояние нуля или единицы. Запись значения полей CSFA и CSFB возможна как напрямую, так и через дублирующий регистр.

В случае записи напрямую значение полей устанавливают выводы в заданное состояние на следующем такте после завершения записи.

В случае записи через дублирующий регистр запись со стороны процессора ведется в дублирующий регистр, из которого значения переносятся в основной регистр по одному из следующих событий:

- таймер достиг нулевого значения;
- таймер достиг периода счета;
- таймер достиг нулевого значения или периода счета.

Режим записи регистра AQCSFRC определяется полем RLDCSF регистра AQSFRC.

Временное задание значений выводов EPWMA и EPWMB осуществляется полями ACTSFA, OTSFA и ACTSFB, OTSFB регистра AQSFRC, соответственно. Полями ACTSFA и ACTSFB определяется возможное действие:

- оставить прежнее значение вывода;
- задать на выводе «0»;

– задать на выводе «1»;
– задать значение, как функцию текущих значений выводов.

При этом вывод принимает заданное значение только при установке бит OTSFA для EPWMA или OTSFB для EPWMB. Биты OTSFA и OTSFB являются самоочищаемыми, то есть после задания на выводе заданного значения биты обнуляются, и по следующему событию выводы примут значение в соответствии с регистрами AQCTLA и AQCTLB.

Для каждого выхода ШИМ (EWPMA и EPWMB) по каждому из 4-х событий (EVENT1, EVENT2, EVENT3, EVENT4) можно выбрать один из четырех источников (CTReqCMPB, CTReqCMPA, CTReqZERO, CTReqPRD).

Приоритет событий: EVENT 1, 2, 3, 4.

То есть выбрав, например:

– для события 1 источник CTReqCMPA и действие – установка по инкременту (EVNT1U_A_MODE);
– для события 2 источник CTReqCMPA и действие – сброс по декременту (EVNT2D_A_MODE)

получим, что установка выхода произойдет, а при декременте и достижении CTReqCMPA не будет произведет сброс по декременту, а будет выполнено действие согласно значению поля EVNT1D_ A_MODE.

#### 13.2.1.5 Блок управления мертвой зоной

Основными назначениями блока управления мертвой зоной являются:

– создание мертвой зоны между сигналами EPWMxA и EPWMxB;
– добавление программных задержек на фронты сигналов;
– добавление программных задержек на срезы сигналов;
– возможность полного отключения и пропускания сигнала без изменений.

Блок управления мертвой зоной позволяет независимо управлять фронтами и срезами сигналов EPWMxA и EPWMxB в соответствии со схемой на рисунке 84. Для преобразования может быть выбран один из входов, оба входа, или оба входа могут быть пропущены на выход без изменения.

Задержка фронтов осуществляется на основе 10-разрядных счетчиков для среза и фронта входящих сигналов: DBRED для фронта сигналов и DBFED для среза сигналов.

Вычисление задержки осуществляется по следующему принципу:

– FED = DBFED x $T_{clk}$;
– RED = DBRED x $T_{clk}$.

Где $T_{clk}$ – период рабочей частоты модуля ШИМ определяется полем HSPCTLDIV регистра TBCTL.

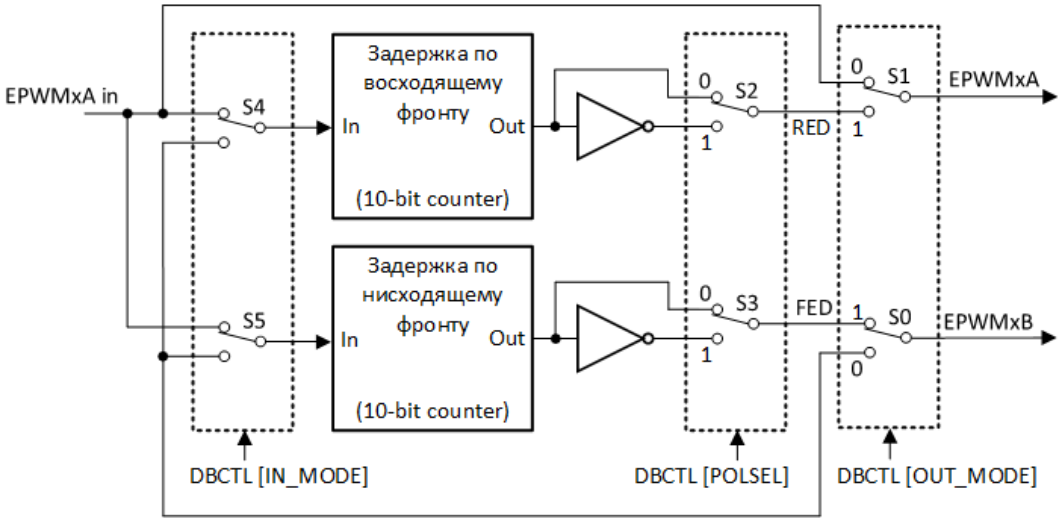



Рисунок 84 – Схема преобразования сигналов модуля управления мертвой зоной

По результатам работы блока управления мертвой зоной могут быть получены как задержки фронтов одиночных сигналов, так и комплементарный выход на основе задержек фронтов и срезов одного из входных сигналов.

#### 13.2.1.6 Блок высокочастотного модулятора

Блок высокочастотного модулятора предназначен для управления силовыми ключами с использованием импульсного трансформатора.

Модулятор позволяет реализовать следующие функции:

- программируемая частота модуляции;
- программируемая ширина первого импульса;
- программируемая сважность выходного сигнала.

Управление блоком осуществляется полностью при помощи регистра PCCTL.

С помощью бита CHPEN возможно выключение блока высокочастотной модуляции и пропускание на выход неизмененного сигнала.

Несущий сигнал получается делением системной частоты на 8 и на значение, соответствующее полю CHPFREQ регистра PCCTL. Скважность несущего сигнала может быть задана при помощи установки требуемого значения в поле CHPDUTY регистра PCCTL.

Первый импульс сигнала может иметь большую длину, чем последующие, для гарантии быстрого включения системы. Управление длинной первого импульса осуществляется полем OSHTWTH регистра PCCTL.

В том случае, если блок включен (бит CHPEN регистра PCCTL в состоянии 1) на выход блока вместо сплошных единичных сигналов выдается модулированные по частоте PSCLK сигналы с расширенным первым импульсом.

За расширение импульса отвечает блок формирования расширенного импульса. Для генерации расширенного импульса в качестве значения поля OSHTWTH необходимо указать, какое количество тактов деленной на 8 системной частоты необходимо подать в качестве первого импульса.

Для формирования модулирующей частоты PSCLK необходимо в поле CHPREQ указать значения делителя. При этом, делитель частоты PSCLK расположен последовательно к предделителю системной частоты на 8. Таким образом результирующая частота будет равна

$$PSCLK = SYSCLK / (8 \bullet CHPREQ).$$

При помощи поля CHPDUTY можно также указать длительность «1» внутри каждого периода модулирующей частоты.

Модулирующий сигнал получается объединением по «ИЛИ» первого импульса и модулирующей частоты PSCLK.

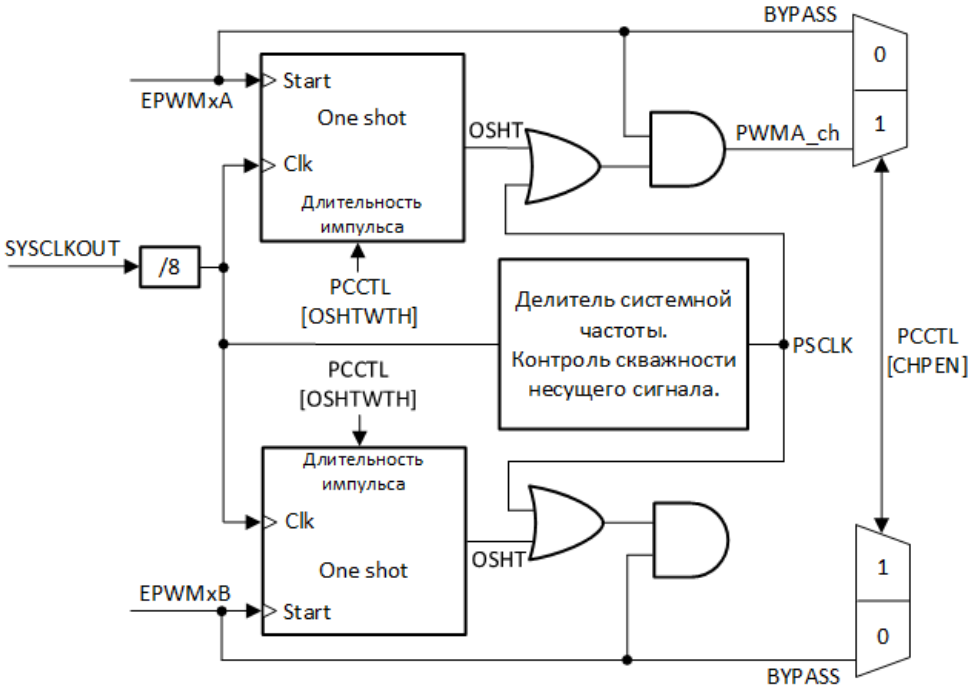



Рисунок 85 – Структурная схема блока высокочастотной модуляции

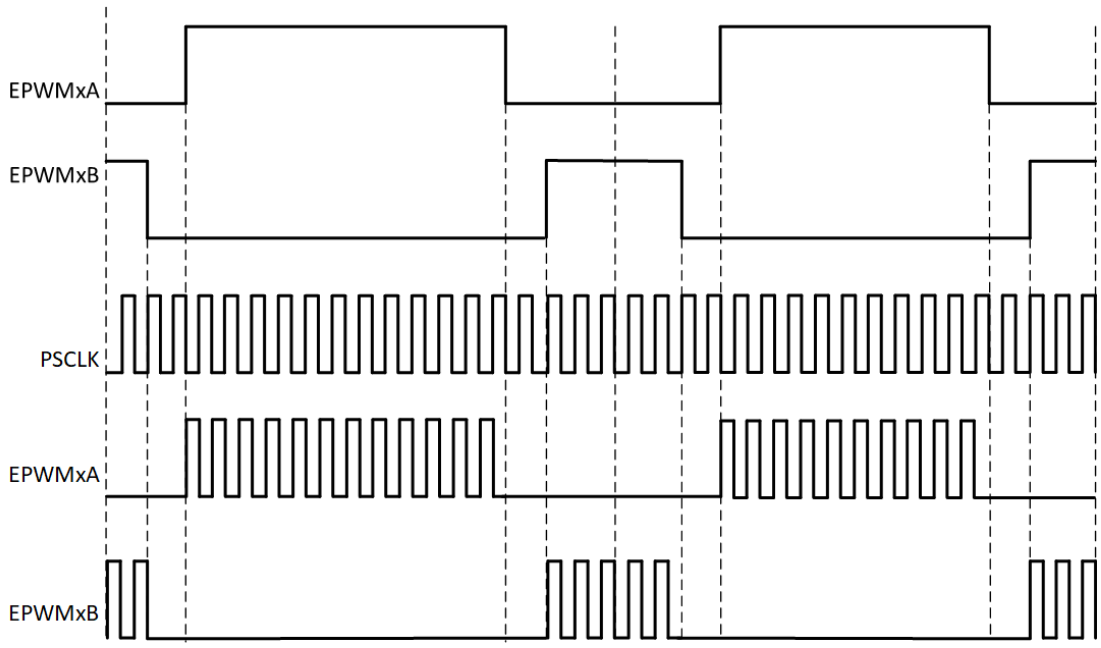



Рисунок 86 – Диаграмма формирования выходного сигнала в блоке

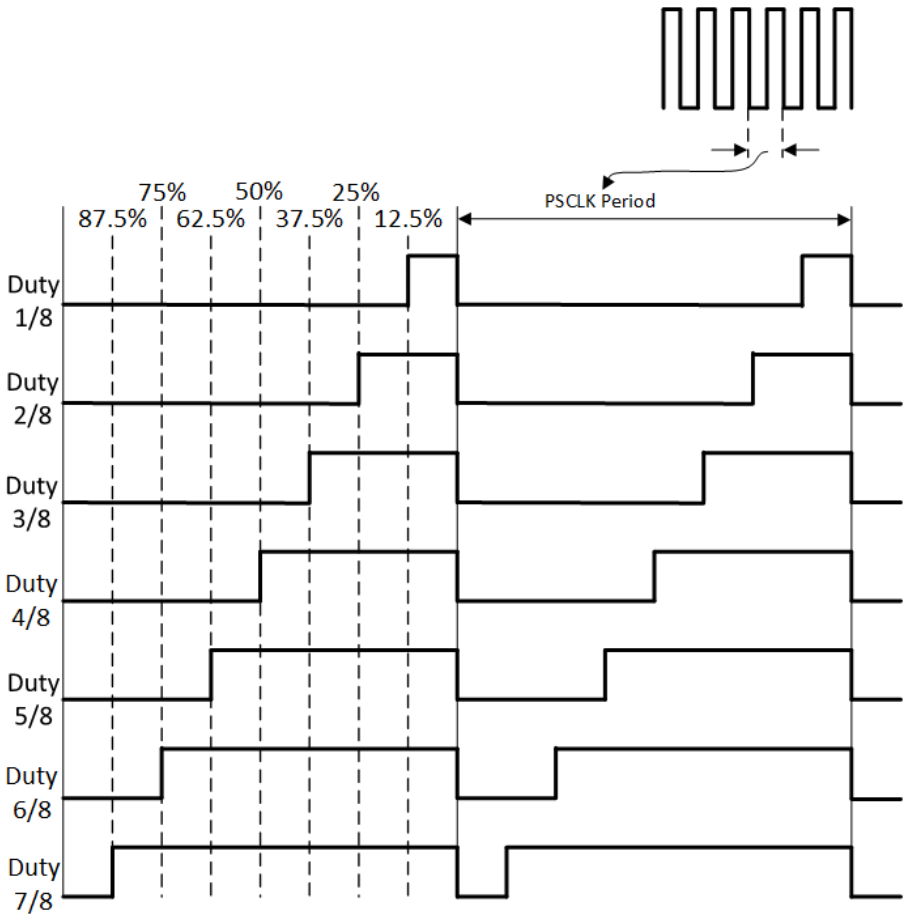



Рисунок 87 – Скважность модулирующей частоты высокочастотного модулятора

Для формирования выходного модулированного сигнала модулирующий сигнал объединятся по «И» с подсинхронизированным к частоте SYSCLK/8 сигналами EPWMxA_syn8 и EPWMxB_syn8. Заметим, что задержка подсинхронизированных к частоте SYSCLK/8 сигналов EPWMxA_syn8 и EPWMxB_syn8 может составлять до 8 тактов относительно сформированных в предшествующем в цепочке формирования выходных сигналов модуле EPWMxA и EPWMxB.

#### 13.2.1.7 Блок защиты

Для быстрого отключения в случае аварии модуль ШИМ имеет шесть входов сигнала ошибок. По каждому событию на входах выходы EPWMxA и EPWMxB ШИМ могут быть настроены на выполнение одного из следующих действий:

- переключение в «1»;
- переключение в «0»;
- переключение в высокоимпедансное состояние;
- продолжать работать.

Модулем поддерживается однократная блокировка выходов в случае короткого замыкания или перегрузки по току и циклическая блокировка в режиме ограничения тока. Также поддерживается комбинационная блокировка, гарантирующая отключение выходной цепи при отсутствии синхросигнала. Все режимы блокировки поддерживаются для любого из входов аварии. Также возможна программная установка сигнала аварии. Для управления блоком реализовано 6 регистров. Когда на одном из входов ошибки TZ1-TZ6 появляется низкий уровень, модуль реагирует на это событие, если данный сигнал ошибки выбран для модуля в регистре TZSEL.

Входные сигналы ошибок могут быть синхронизированы с системной частотой, а могут быть полностью асинхронны. Синхронная работа сигналов ошибок позволяет

отфильтровать короткие импульсы на них, в то же время асинхронный вход сигналов гарантирует их активность даже при выключенном синхросигнале.

Каждое событие TZ может быть сконфигурировано для управления одиночным или циклическим синхронным и асинхронным режимом заданием соответствующих полей CDCn и OSHTn регистра TZSEL.

В циклическом режиме (Cycle-by-Cycle (CBC)) модуль незамедлительно реагирует на сигнал ошибки и переводит выходы EPWMxA и EPWMxB в состояние, определенное регистром TZCTL (состояние на время циклической аварии должно быть настроено в двух полях TZx_CW и TZx_CR). Также выставляется флаг CBC регистра TZFLG и генерируется прерывание EPWMx_TZINT, если это разрешено битом TZEINT регистра PIE. При этом, если сигнал ошибки снимается, выходы переходят в обычное состояние по событию (TBCTR = 0x0000).

В режиме одиночного срабатывания (One-Shot (OSHT)), модуль незамедлительно реагирует на сигнал ошибки и переводит выходы EPWMxA и EPWMxB в состояние, определенное регистром TZCTL. При этом в регистре TZFLG выставляется флаг одиночной ошибки OST и вырабатывается прерывание EPWMx_TZINT, если это разрешено битом TZEINT регистра PIE. Состояние одиночной ошибки должно быть очищенно программно записью бита OST регистра TZCLR.

В режиме асинхронной работы вывод будет переведен в соответствующее состояние на все время активности сигнала аварии, при этом, вывод продолжит работать в штатном режиме в момент отключения сигнала аварии.

Действие, выполняемое модулем по событию ошибки на каждом из входов, выбирается индивидуально полями TZA и TZB регистра TZCTL.

При этом, если различные действия будут выбраны для асинхронного и циклического или однократного событий, возможна ситуация, при которой вывод сначала переключится из рабочего режима в состояние, соответствующие настройке асинхронного режима, после чего – в состояние, соответствующее одному из синхронных режимах отключения.

Входы имеют следующие назначения:

- nTZ1_INT назначен на CMP0_PE,
- nTZ2_INT назначен на CMP1_PE,
- nTZ3_INT назначен на CMP2_PE,
- nTZ4_INT назначен на CMP3_PE,
- nTZ5_INT назначен на 1'b1,
- nTZ6_INT назначен на 1'b1.

Все входы nTZ6_EXT назначены на внешние выводы в соответствии с таблицей выводов. Активный уровень сигнала аварии – логический ноль.

#### 13.2.1.8 Блок ШИМ высокого разрешения

Основным назначением блока ШИМ высоко разрешения (HRPWM) является повышение разрешения цифрового ШИМ (ePWM), когда его разрешения не хватает для применения.

На блок HRPWM поступает сигнал PWMx, скважность и период которого задаются в блоке ePWM. Скважность PWMx может изменяться с шагом в один период тактовой частоты контроллера ePWM. На рисунке 88 показан случай, когда длительность логической «1» на EPWMx равна трем периодам тактовой частоты. Функция блока состоит в том, чтобы выработать сигнал, изменение длительности которого можно регулировать с меньшим шагом. Регулировка осуществляется в зависимости от значения, которое записано в поле HRPWMx_SHIFT регистра HRPWM_CNTR.

В результате работы будет получен выходной сигнал, который может иметь 32 значения длительности логической «1». Длительность логической «1» определяется по формуле:

$$T(HRPWMxh) = T(EPWMxh) + HRPWMx_SHIFT \cdot \frac{Tclk}{32} - \frac{Tclk}{2}$$

Где $T(EPWMx\text{h})$ – длительность логической «1» сигнала цифрового ШИМ, $Tclk$ – период сигнала тактовой частоты работы контроллера цифрового ШИМ ePWM.

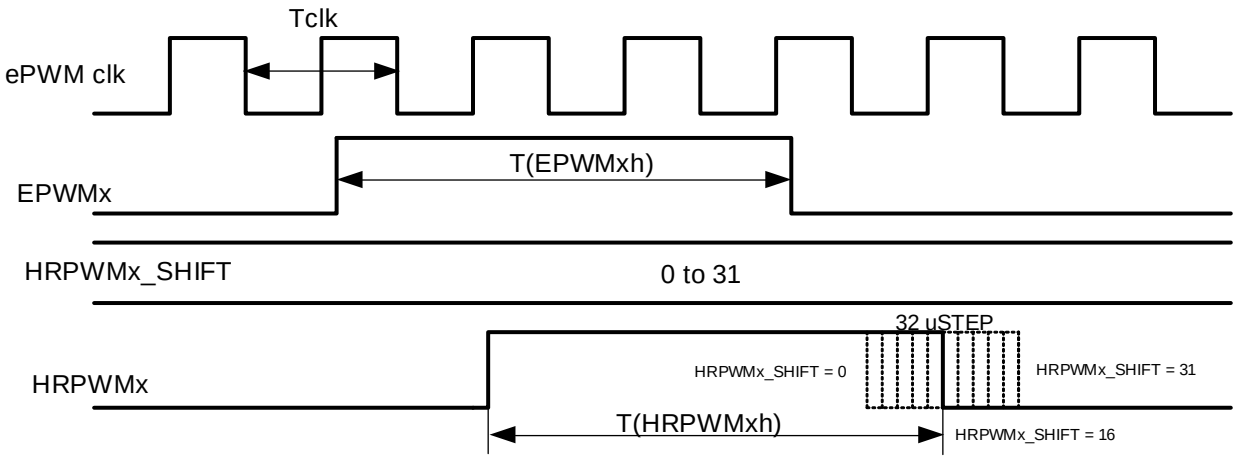



Рисунок 88 – Временная диаграмма формирования сигнала HRPWMx

Основным рабочим режимом блока является режим при HRPWMx_MX = 1. При HRPWM1_MX = 2 выводится инверсный сигнал. При HRPWMx_MX = 0 на выход HRPWMx выводится сигнал цифрового ШИМ (EPWMx).

Для корректной работы блока HRPWM необходимо настраивать частоту работы блока ePWM в диапазоне от 50 МГц до 160 МГц (ePWM clk), а также частота EPWMx должна быть в диапазоне от 20 кГц до 2 МГц. Длительность «0» или «1» сигнала EPWMx должна быть не менее двух тактов сигнала ePWM clk. В противном случае работа блока не гарантируется.

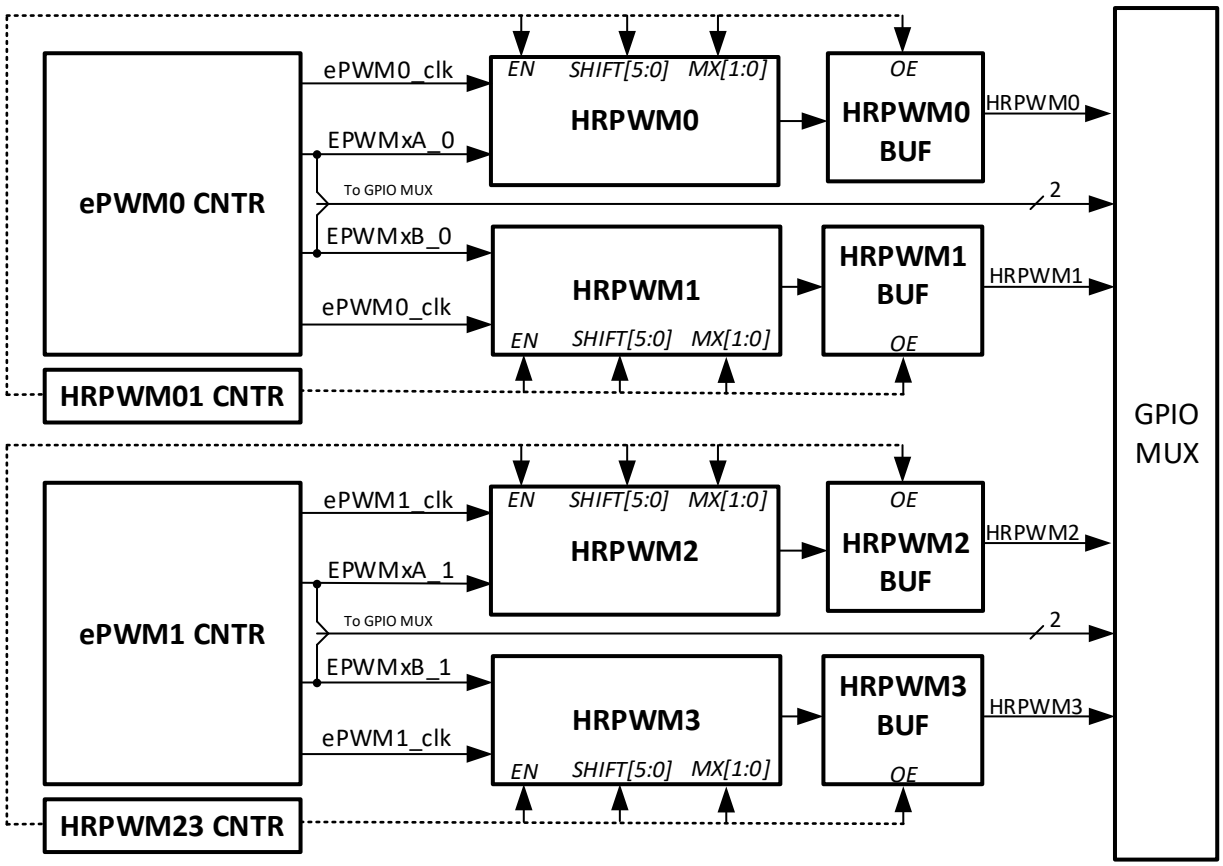



Рисунок 89 – Схема подключения блоков HRPWM к контроллерам ePWM.

#### 13.2.1.9 Прерывания и запросы синхронизации АЦП

Контроллер позволяет реализовать прерывания от буферов FIFO регистров периода и сравнения, прерывания от блока защиты и прерывание по событиям таймера.

- ***SHDWAFULL_INT*** – прерывание устанавливается по событию переполнения буфера регистра сравнения А; Событие прерывания также, как и флаг прерывания, в регистре ETFLG стоит до момента устранения источника события, то есть считывания хотя бы одного значения из буфера регистра сравнения А;
  Программная установка флага возможна установкой соответствующего бита ETFRC. При программной установке и отсутствии переполнения запрос прерывания и соответствующий флаг регистра ETFLG будет стоять один такт PCLK.
  Программный сброс регистра возможен при помощи регистра ETCLR, но действовать он будет также один такт PCLK, после чего значение флага регистра примет значение, соответствующее реальной ситуации в буфере. Синхронизирован с синхросигналом PCLK;
- ***SHDWAEMPTY_INT*** – прерывание устанавливается по событию опустошения буфера регистра сравнения А.
  Функционирует аналогично SHDWAFULL_INT;
- ***SHDWBFULL_INT*** – прерывание устанавливается по событию переполнения буфера регистра сравнения Б;
- ***SHDWBEMPTY_INT*** – прерывание устанавливается по событию опустошения буфера регистра сравнения Б;
- ***SHDWPRDFULL_INT*** – прерывание устанавливается по событию переполнения буфера регистра периода;
- ***SHDWPRDEMPTY_INT*** – прерывание устанавливается по событию опустошения буфера регистра периода;
  События DMA запросов по событиям опустошения буферов FIFO регистров периода и сравнения ***(SHDWAEMPTY_DMA, SHDWBEMPTY_DMA, SHDWPRDEMPTY_DMA)*** функционируют аналогично прерываниям, то есть флаг запроса выставляется до момента появления в FIFO значений регистров. Флаги запросов DMA могут быть разрешены или запрещены при помощи бит SHDWAEMPTY_DMA_EN, SHDWBEMPTY_DMA_EN и SHDWPRDEMPTY_DMA_EN регистра ETSEL. События DMA синхронизируются по синхросигналу PCLK.
- ***nEPWMxTZINT*** – прерывания по событиям аварий. Подробное функционирование описано в разделе «Блок защиты»;
- ***nEPWMxINT*** – прерывание по событиям таймера основного счета. Выбор события, по которому прерывания выставляются, осуществляется полем INTSEL регистра ETSEL. Кроме того, при помощи поля INTPRD регистра ETPS можно управлять на каком событии по счет (от первого до третьего) будет выставлено прерывание. Текущее значения счетчика событий может быть прочитано в поле INTCNT того же регистра.

Контроллер позволяет выдавать запросы начала цепочки преобразований АЦП ***EPWMxSOCA, EPWMxSOCB,*** по событиям таймера. Логика формирования запросов аналогична логике формирования прерывания ***nEPWMxINT*** и управляется полями SOCBSEL, SOCASEL регистра ETSEL. Выдачу запросов преобразований можно разрешить или запретить битами SOCAEN и SOCBEN регистра ETSEL. Запросы на преобразования, синхронные с синхросигналом TCLK. Запрос выставляется до момента устранения источника запроса.

## 13.3 Контроллер квадратурных преобразователей (QEP_CNTR)

Контроллеры квадратурных преобразователей предназначены для преобразования цифрового сигнала с датчика положения вала для определения текущего положения вала, а также его скорости и направления вращения.

В микросхеме реализовано два контроллера квадратурных преобразователей. В состав каждого квадратурного преобразователя входят:

- блок квадратурного декодера QDU, декодирующий входные сигналы с датчика в сигналы направления и счета контроллера;
- блок счетчика положения PCCU;
- блок захвата фронтов QCAP для низкоскоростных измерений;
- блок таймера временных отсчетов UTIME;
- блок сторожевого таймера QWDOG.

Структурная схема контроллера приведена на рисунке 90.

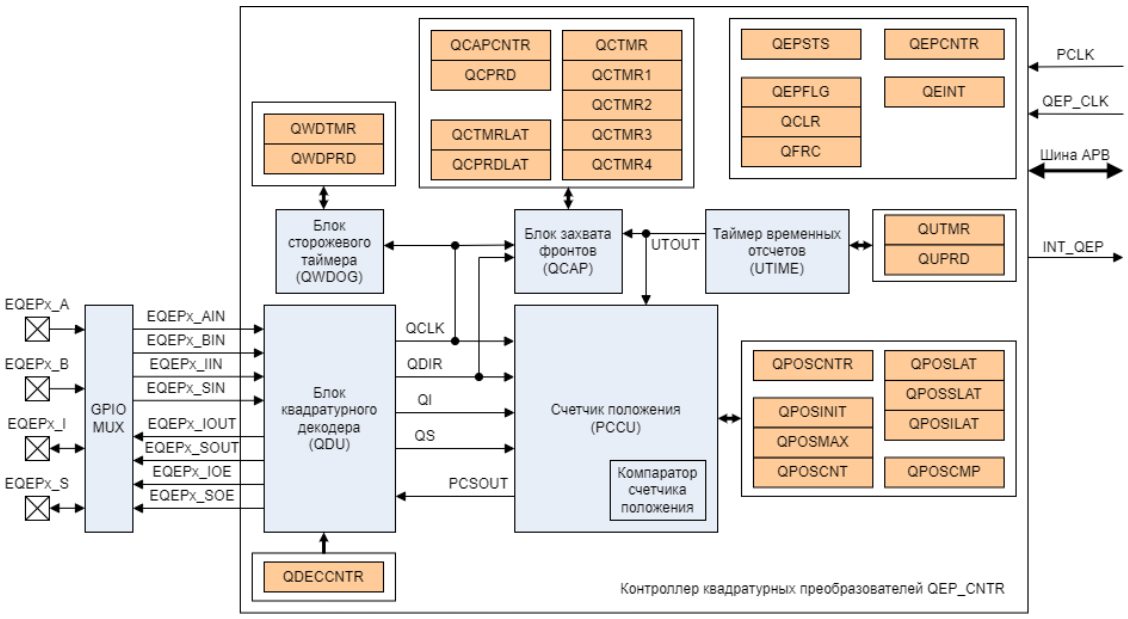



Рисунок 90 – Структурная схема контроллера квадратурных преобразователей

Контроллер квадратурных преобразователей тактируется частотами PCLK и QEP_CLK. Частота PCLK используется при записи в регистры блока по шине APB. Работа блоков контроллера осуществляется на частоте QEP_CLK. Возможен программный сброс внутреннего состояния контроллера с помощью бита QPEN регистра QEPCNTR.

Контроллер использует два вывода микросхемы в квадратурном режиме и в режиме счета/направления в качестве входов. Также имеются специальный индексный вывод и вывод стробирования, которые могут работать на вход и выход.

Список выводов:

- EQEPx_A и EQEPx_B. Эти входы могут использоваться в режиме квадратурного счета или режиме счета/направления блока квадратурного декодера QDU. В режиме квадратурного счета входы представляют собой отстоящие друг от друга на 90° по фазе периодические сигналы, по которым можно определить скорость и направление вращения вала. В режиме счета/направления тактовый сигнал xCLK и сигнал направления xDIR подаются непосредственно с внешнего датчика с выводов EQEPx_A и EQEPx_B соответственно;
- EQEPx_I – вход/выход сигнала полного оборота вала (сигнала индексации), относительно которого определяется положение вала. По появлению этого сигнала на входе возможна инициализация или сброс счетчика положения.

– EQEPx_S – вход/выход сигнала стробирования – сигнала общего назначения, по появлению которого на входе может осуществляться инициализация или сохранение счетчика положения.

Контроллер квадратурных преобразователей формирует сигналы направления вывода сигналов полного оборота вала EQEPx_I и стробирования EQEPx_S для контроллеров портов ввода-вывода при выполнении выводами функций контроллера QEP – сигналы EQEPx_IOE и EQEPx_SOE соответственно.

### 13.3.1 *Блок квадратурного декодера QDU*

Блок предназначен для преобразования входных сигналов с датчика сигнала счета/направления. Блок формирует внутренние входные для остальных блоков контроллера сигналы счета QCLK, направления QDIR, сигнал полного оборота вала QI и стробирования QS, а также внешний выходной сигнал синхронизации на одном из выводов EQEPx_I или EQEPx_S. Активен всегда после включения контроллера квадратурного преобразователя.

Схема квадратурного декодера приведена на рисунке 91.

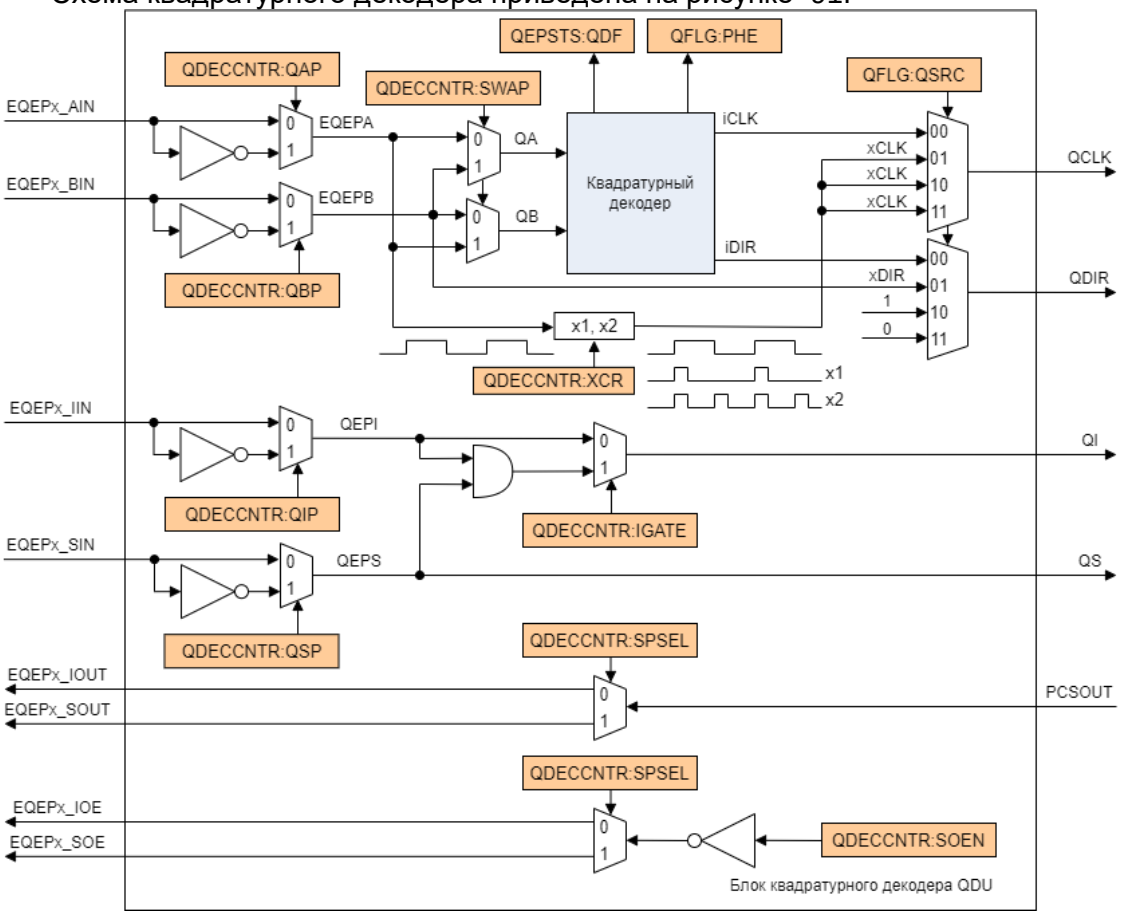



Рисунок 91 – Структурная схема блока квадратурного декодера

Управление квадратурным декодером осуществляется через регистр QDECCNTR.

Каждый из входных сигналов с выводов EQEPx_AIN, EQEPx_BIN, EQEPx_IIN, EQEPx_SIN может быть инвертирован установкой соответствующего бита в регистре QDECCNTR.

Также с помощью бита SWAP возможен обмен местами сигналов EQEPA и EQEPB, что изменяет направление счета.

Возможно объединение сигналов QEPI и QEPS на сигнал QI с помощью бита IGATE.

Формирование сигнала внешней синхронизации на выводы EQEPx_I или EQEPx_S контролируется битами SOEN и SPSEL.

Блок формирует сигналы направления вращения в текущий момент (бит QDF регистра QEPSTS) и флаг ошибки фазы сигналов EQEPA, EQEPB (бит PHE регистра QFLG).

Блок поддерживает следующие режимы работы (выбор режима задается полем QSRC):

- квадратурный счет (QSRC = 2'b00);
- режим счета/направления (QSRC = 2'b01);
- счет вверх (QSRC = 2'b10);
- счет вниз (QSRC = 2'b11).

Во всех режимах, кроме режима квадратурного счета, предусмотрен выбор скорости счета по передним и задним фронтам сигналов EQEPA и EQEPB или только по передним фронтам (управляется битом XCR).

#### 13.3.1.1 Режим квадратурного счета

Квадратурный преобразователь формирует тактовый сигнал iCLK и сигнал направления счета iDIR (вверх/вниз) для блока счетчика положения PCCU.

Для генерации импульсов счета iCLK для счетчика положения используются как задние, так и передние фронты входных сигналов. Таким образом, частота счетных импульсов, генерируемых блоком декодера для блока счетчика положения, в два раза превышает частоту каждого из входов и соответствует частоте каждого переключения сигналов {QA, QB} согласно автомату определения направления.

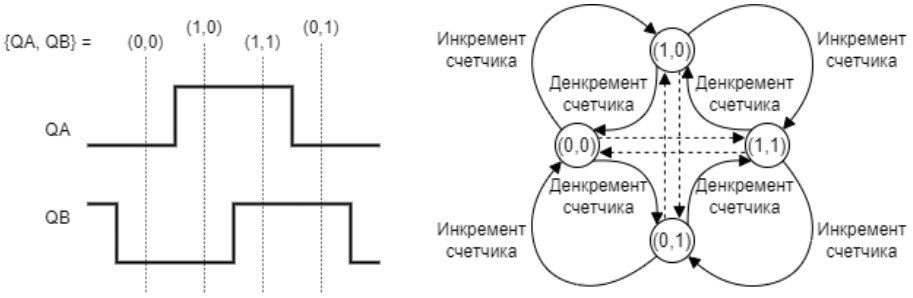



Рисунок 92 – Автомат определения направления вращения

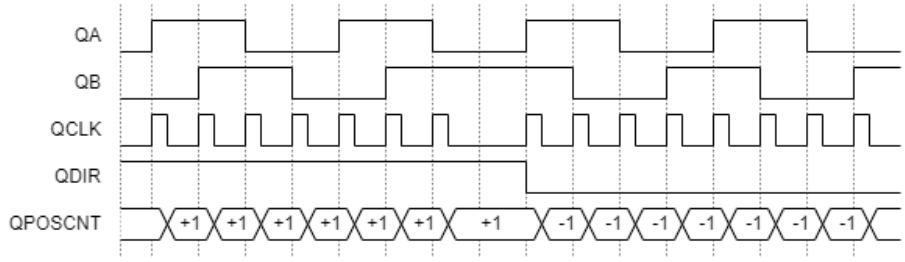



Рисунок 93 – Временная диаграмма работы квадратурного декодера в режиме квадратурного счета

В том случае, если входные сигналы равны по фазе (переходы между состояниями, соответствующие равенству сигналов по фазе, указаны пунктирными линиями для автомата состояний на рисунке 92), то есть осуществляется одновременное переключение сигналов {QA, QB}, блоком квадратурного декодера вырабатывается флаг ошибки фазы PHE.

#### 13.3.1.2 Режим счета/направления

Сигнал EQEPA используется для определения скорости вращения, а сигнал EQEPB определяет направление. Счетчик положения PCCU по каждому переднему фронту сигнала EQEPA инкрементируется в том случае, если EQEPB = 1, или декрементируется, если EQEPB = 0.

#### 13.3.1.3 Режимы счета вверх и вниз

Режимы используются для вычисления частоты следования импульсов на выводе EQEPx_A. Сигнал на выводе EQEPx_B не используется. Счетчик положения PCCU в таком режиме работает всегда на увеличение или уменьшение в зависимости от режима счета.

### 13.3.2 *Блок счетчика положения PCCU*

Блок осуществляет счет текущего положения внешнего датчика в 32-разрядный регистр QPOSCNT на тактовой частоте QCLK от квадратурного декодера QDU (направление счета задается сигналом QDIR) в границах от 0 до QPOSMAX.

Управление блоком счетчика положения осуществляется через регистры QEPCNTR и QPOSCNTR.

Возможна установка различных независимых режимов работы блока: сброса, инициализации и сохранения значения счетчика положения.

В блоке счетчика положения предусмотрен дополнительный модуль компаратора текущего положения, с помощью которого может быть сформирован сигнал выходной синхронизации PSCOUT.

При появлении сигнала QI, счетчик положения ожидает от квадратурного декодера любое изменение на квадратурных входах и запоминает значение этого события – фронт, активный вход (QA или QB), а также направление вращения. Это событие называется маркером индексации. По маркеру индексации в регистре QEPSTS устанавливается флаг формирования первого маркера индексации FIMF и направление вращения в момент захвата первого маркера индексации FIDF. Флаг FIMF может быть сброшен программно. Если направление вращения изменится, то сохраненное в маркере значение фронта (передний или задний фронт) меняется на обратное. Например, первый маркер индексации сформирован при счете вверх при QA = 0 и переключении сигнала QB из 1 в 0, тогда все последующие маркеры индексации формируются при QA = 0 и переключении QB из 1 в 0 при счете вверх, а при счете вниз при QA = 0 и переключении QB из 0 в 1.

При работе квадратурного декодера в режимах, отличных от квадратурного счета, маркер индексации формируется счетчиком положения всегда по первому переключению сигналов {QA, QB} после формирования сигнала QI без определения события.

Маркер индексации используется для сброса и сохранения значения счетчика положения.

#### 13.3.2.1 Режимы сброса счетчика положения

Режим сброса счетчика положения определяется полем PCRM регистра QEPCNTR.

Режим **сброса счетчика по достижении максимального значения** (рисунок 94) устанавливается значением поля PCRM = 2’b01. Максимальное положение задается регистром QPOSMAX. Счетчик устанавливается в 0 при достижении счетчиком значения QPOSMAX (QPOSCNT больше или равен QPOSMAX) или в значение QPOSMAX при достижении счетчиком 0. По каждому событию выставляется соответствующий флаг прерывания, фиксирующийся в регистре QFLG – прерывание о переполнении при счете вверх PCO (условия: QPOSCNT ≥ QPOSMAX, счет вверх) и прерывание о переполнении при счете вниз PCU (условия: QPOSCNT = 0, счет вниз). Все остальные режимы сброса счетчика положения включают в себя функционал этого режима.

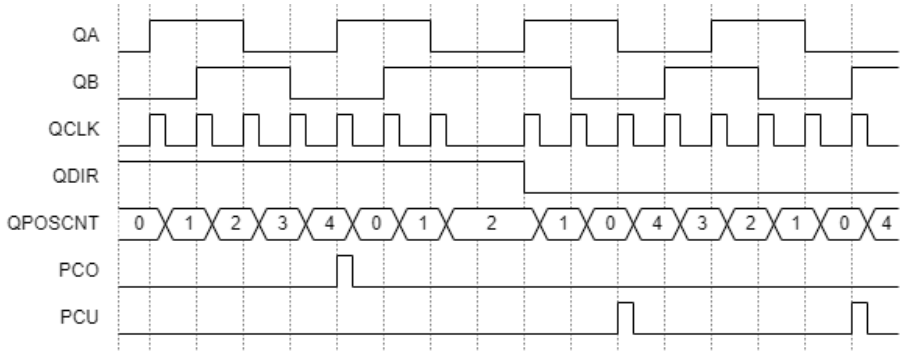



Рисунок 94 – Временная диаграмма сброса счетчика по достижении максимального значения

Режим **сброса счетчика по маркеру индексации** (рисунок 95) устанавливается значением поля PCRM = 2'b00 и активен по умолчанию. Если маркер индексации формируется во время инкремента счетчика положения (вращения в «положительном» направлении), счетчик положения сбрасывается в 0. Если маркер индексации формируется в момент реверсивного движения, счетчик положения сбрасывается в значение, устанавливаемое QPOSMAX.

Если значение счетчика до сброса счетчика по маркеру индексации не соответствует значению сброса, формируются флаг статуса ошибки счетчика положения PCEF (обновляемый по каждому новому событию полного оборота) и флаг прерывания PCE. Помимо этого, по событию полного оборота осуществляется сохранение значения счетчика положения до сброса QPOSCNT в регистр QPOSILAT.

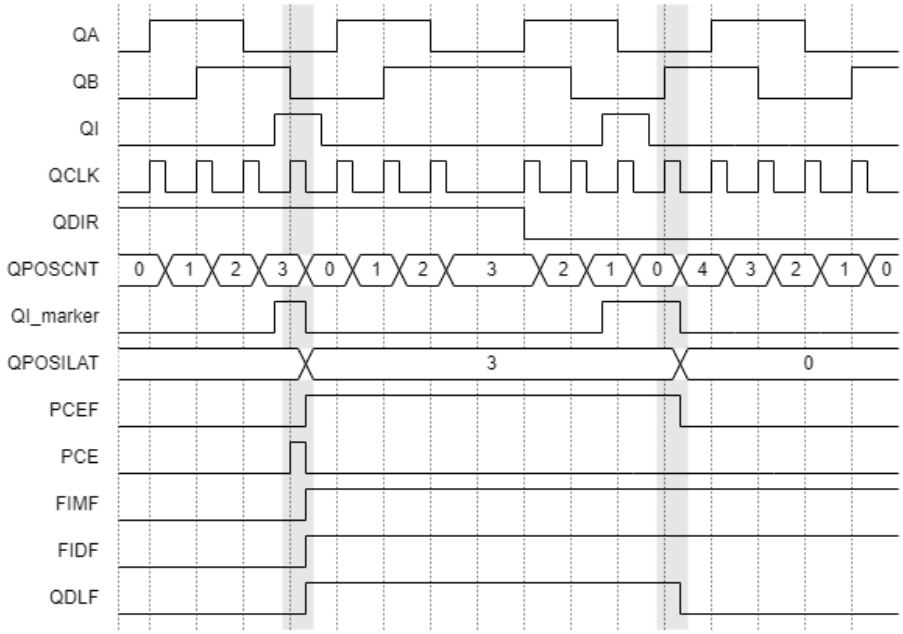



Рисунок 95 – Временная диаграмма сброса и сохранения счетчика положения по маркеру индексации в режиме квадратурного счета QDU

В режиме **сброса счетчика по первому событию полного оборота** (PCRM = 2'b10) счетчик сбрасывается в 0 или устанавливается в значение QPOSMAX (аналогично режиму PCRM = 2'b00) по первому событию полного оборота, после чего блоком квадратурного преобразователя выставляется флаг формирования первого маркера индексации FIMF, и сброс счетчика положения происходит аналогично режиму сброса по достижении максимального значения PCRM = 2'b01 до тех пор, пока установлен флаг FIMF.

В режиме **сброса по временным отсчетам** (PCRM = 2'b11) сброс/установка (в зависимости от направления счета) происходит по прошествии времени, определенного блоком таймера временных отчетов UTIME (по сигналу UTOUT). Данный режим может быть использован для вычисления скорости вращения датчика.

#### 13.3.2.2 Режимы инициализации счетчика положения

Режимы инициализации счетчика положения определяются полями IEI (инициализация по событию полного оборота), SEI (инициализация по событию стробирования) и SWI (программная инициализация) регистра QEPCNTR. Счетчик инициализируется значением регистра QPOSINIT. Каждый из режимов инициализации функционирует независимо.

**Инициализация по событию полного оборота QI (IEI).** Сигнал индексации QI может использоваться для инициализации счетчика по переднему или заднему фронту. Если IEI = 2'b10, то счетчик инициализируется значением QPOSINIT по переднему фронту сигнала индексации. Аналогично, если IEI = 2'b11, то счетчик инициализируется значением QPOSINIT по заднему фронту сигнала индексации. При других значениях поля IEI инициализация по событию полного оборота не осуществляется.

**Инициализация по событию стробирования (SEI).** Если SEI = 2'b10, то счетчик инициализируется значением QPOSINIT по переднему фронту сигнала стробирования QS. Если SEI = 2'b11, то счетчик инициализируется значением QPOSINIT по переднему фронту сигнала стробирования QS при счете вверх (QDIR = 1) и по заднему фронту сигнала QS при счете вниз (QDIR = 0). При других значениях поля SEI инициализация по событию стробирования не осуществляется.

**Программная инициализация (SWI).** Счетчик инициализируется значением QPOSINIT при SWI = 1. Бит не сбрасывается автоматически. Осуществляется непрерывная инициализация счетчика, пока бит SWI не будет сброшен в 0.

#### 13.3.2.3 Режимы сохранения значения счетчика положения

Режимы сохранения значения счетчика положения определяются полями IEL и SEL регистра QEPCNTR. Каждый из режимов сохранения функционирует независимо.

**Сохранение по событию полного оборота (IEL).** Сохранение осуществляется по фронту (IEL = 2'b01) или срезу (IEL = 2'b10) сигнала QI, а также по маркеру индексации (IEL = 2'b11, рисунок 95) в регистр QPOSILAT. В результате сохранения значения в регистр QPOSILAT формируется флаг сохранения значения IEL в регистре QFLG.

В случае, если выбран режим сброса по событию полного оборота PCRM = 2'b00, сохранение по событию полного оборота всегда осуществляется по маркеру индексации (аналогично IEL = 2'b11), при этом значение поля IEL игнорируется. В других случаях при значении IEL = 2'b00 сохранение по событию полного оборота не осуществляется.

**Сохранение по фронту сигнала стробирования (SEL).** Значение счетчика сохраняется в регистр QPOSSLAT по каждому переднему фронту сигнала QS, если SEL = 0 (захват по заднему фронту может быть реализован инверсией сигнала стробирования квадратурным преобразователем). Если SEL = 1, сохранение в QPOSSLAT происходит по переднему фронту QS при прямом направлении вращения и по заднему фронту при вращении в обратном направлении. При каждом сохранении значения счетчика в регистр QPOSSLAT устанавливается флаг прерывания SEL в регистре QFLG.

**Сохранение по временным отсчетам** от модуля таймера временных отчетов UTIME (по сигналу UTOUT) осуществляется в регистр QPOSLAT.

#### 13.3.2.1 Компаратор счетчика положения

Компаратор осуществляет сравнение значения счетчика положения QPOSCNT со значением в регистре QPOSCMP и при равенстве значений формирует внешний синхросигнал PCSOUT и (опционально) прерывание PCM. Управление компаратором осуществляется через регистр QPOSCNTR. Работа компаратора разрешается установкой бита PCE. Регистр QPOSCMP может использовать режим отложенной записи (управляется битом PCSHDW), когда отложенное значение записывается во внутренний регистр сравнения из регистра QPOSCMP по событиям, определяемым битом PCLOSD – при

совпадении QPOSCNT = 0 или при совпадении результатов сравнения. При использовании режима отложенной записи по событию записи взводится флаг готовности к отложенной записи PCR в регистре QFLG. Если режим отложенной записи не используется, запись осуществляется напрямую в QPOSCMP (в таком случае флаг PCR также не используется).

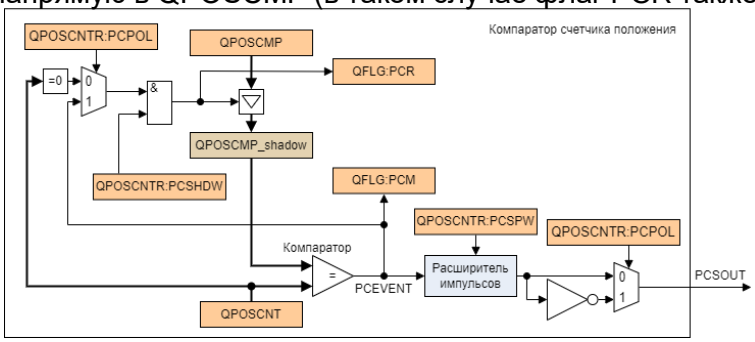



Рисунок 96– Структурная схема компаратора счетчика положения

Флаг успешного сравнения PCM в регистре QFLG устанавливается, когда QPOSCNT = QPOSCMP (по фронту сигнала PCEVENT), при этом также формируется синхроимпульс требуемой длительности для извещения внешнего устройства (сигнал PCSOUT). Длительность сигнала PCSOUT управляется полем PCSPW и составляет от 1*4 до 4096*4 периодов частоты QEP_CLK. В случае нового успешного сравнения при уже активном импульсе PCSOUT будет сгенерирован новый импульс заданной длины от нового события сравнения. Сигнал PCSOUT может быть инвертирован с помощью бита PCPOL.

### 13.3.3 *Блок захвата фронтов QCAP*

Блок захвата фронтов предназначен для измерения времени периода сигнала счета QCLK, что может быть использовано для вычисления скорости медленно вращающегося вала.

Управление блоком захвата фронтов осуществляется через регистр QCAPCNTR. Работа блока разрешается установкой бита CEN. Блок-схема приведена на рисунке 97.

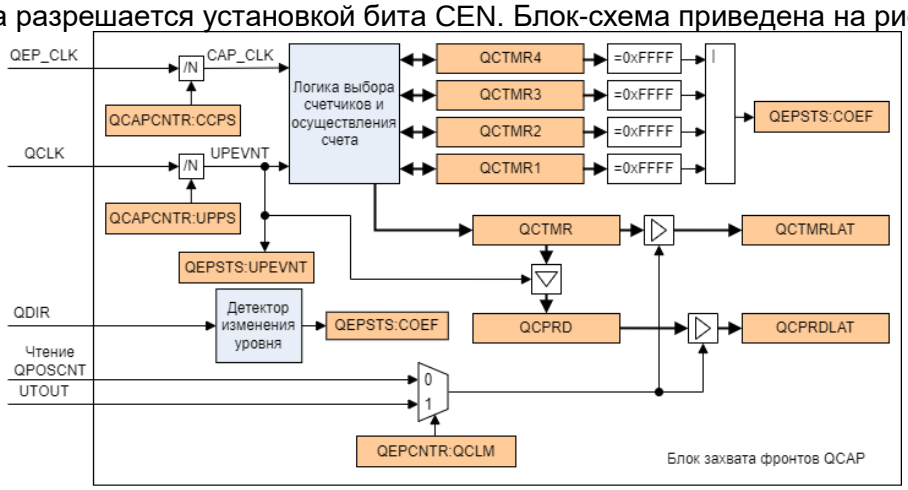



Рисунок 97 – Структурная схема блока захвата фронтов

16-разрядные счетчики блока захвата фронтов QCTMR1-QCTMR4 декрементируются на частоте QEP_CLK, которая может быть прорежена с помощью поля CCPS. Значение одного из счетчиков QCTMR1-QCTMR4 в зависимости от режима работы дублируется в регистр таймера захвата QCTMR. По сигналу UPEVNT – прореженному сигналу QCLK на определяемое полем UPPS значение – осуществляется сохранение значения QCTMR в регистр QCPRD, при этом взводится флаг UPEVNT в регистре QEPSTS для индикации, что в QCPRD сохранено новое значение QCTMR.

Значения таймера захвата QCTMR и регистра периода QCPRD могут быть сохранены в регистры QCTMRLAT и QCPRDLAT по чтению регистра QPOSCNT или срабатыванию блока таймера временных отсчетов UTIME (определяется битом QCLM регистра QEPCNTR).

Результат вычисления скорости вала с помощью блока захвата фронтов можно считать достоверным, если число периодов частоты QEP_CLK между квадратурными событиями не более 65535 (то есть не превышает разрядности счетчиков) и в процессе счета не изменилось направление вращения. По переполнению счетчиков QCTMR1-QCTMR4 взводится флаг COEF. Если в процессе счета QCTMR1-QCTMR4 изменяется направление движения вала, формируется флаг изменения направления вращения при захвате фронтов – CDEF.

Для более точного определения времени между квадратурными событиями реализован режим захвата нескольких периодов сигнала счета QCLK с помощью бита BM. Режим захвата нескольких периодов доступен только при прореживании сигнала QCLK не более чем в два раза (UPPS ≤ 1). В этом режиме для счета используются регистры QCTMR1-QCTMR4. Когда режим захвата нескольких периодов квадратурных событий не используется, счет осуществляется только счетчиком QCTMR1, значение которого дублируется в QCTMR.

При отсутствии коэффициента прореживания (UPPS = 0) счет в режиме захвата нескольких периодов осуществляется счетчиками QCTMR1-QCTMR4, сигнал UPEVNT формируется по каждому квадратурному событию. Счет осуществляется следующим образом: после включения режима захвата нескольких периодов осуществляется счет в каждый из регистров QCTMR1-QCTMR4. В регистр QCTMR дублируется значение QCTMR1. По событию UPEVNT регистр QCTMR1 сбрасывается, в регистр QCTMR дублируется значение счетчика QCTMR2. По следующему событию UPEVNT сбрасывается счетчик QCTMR2, в регистр QCTMR дублируется значение счетчика QCTMR3, и так далее. Сброс текущего счетчика и дублирование значения следующего счетчика в регистр таймера захвата QCTMR осуществляются циклически – после достижения QCTMR4 следующим регистром является QCTMR1 (рисунок 98). Таким образом, в регистр QCPRD по каждому событию UPEVNT сохраняется среднее значение по четырем периодам сигнала QCLK. Первые четыре захваченных в QCPRD значения после включения режима захвата нескольких периодов не содержат действительных результатов по четырем периодам QCLK, так как осуществляется накопление данных.

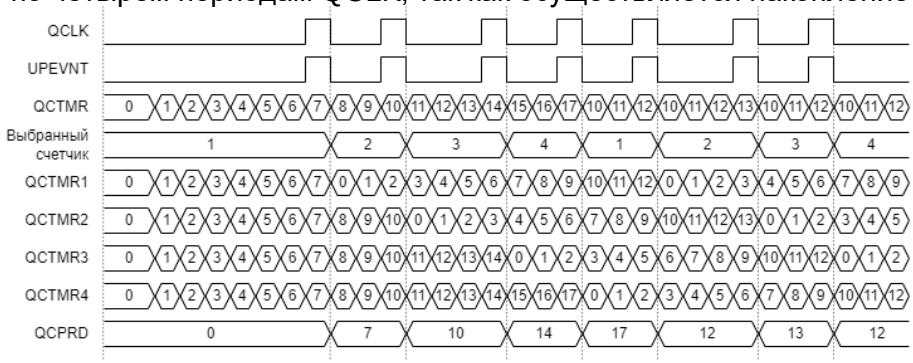



Рисунок 98 – Временная диаграмма работы блока захвата фронтов в режиме захвата нескольких периодов, UPPS = 0

При использовании коэффициента прореживания UPPS = 1 счет в режиме захвата нескольких периодов осуществляется регистрами QCTMR1 и QCTMR3, сигнал UPEVNT формируется по каждому второму квадратурному событию. Функционирование полностью аналогично работе режима без прореживания, но с учетом факта прореживания каждого второго сигнала и дублирования значений только счетчиков QCTMR1 и QCTMR3 в регистр таймера захвата QCTMR (рисунок 99). Первые два захваченных в QCPRD значения после включения режима захвата нескольких периодов не содержат действительные результаты по четырем периодам QCLK, так как осуществляется накопление данных.

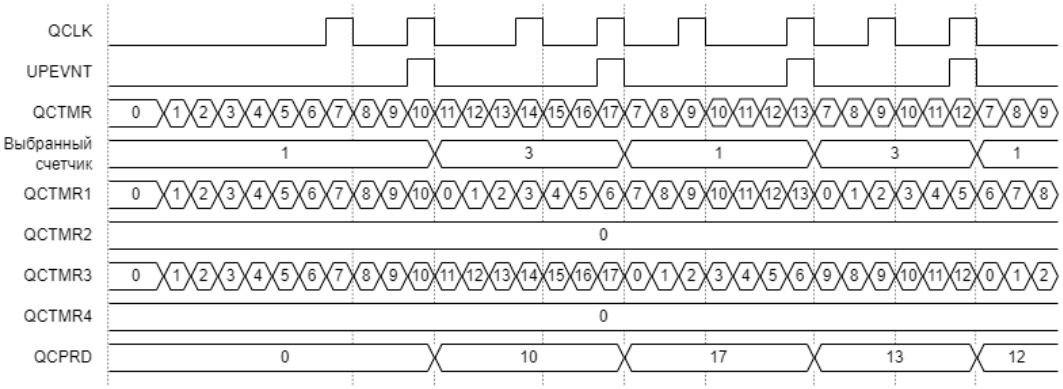



Рисунок 99 – Временная диаграмма работы блока захвата фронтов в режиме захвата нескольких периодов, UPPS = 1

Значения регистров таймера захвата QCTMR и периода захвата QCPRD сохраняются в регистры QCTMRLAT и QCPRDLAT соответственно при чтении регистра QPOSCNT по системным шинам APB или по событию таймера временных отсчетов UTIME (выбор события сохранения в регистры осуществляется битом QCLM регистра QEPCNTR).

### 13.3.4 *Блок таймера временных отсчетов UTIME*

Блок таймера временных отсчетов предназначен для периодического оповещения системы о необходимости начать измерение скорости вращения вала. Представляет собой 32-разрядный таймер, инкрементирующийся на частоте QEP_CLK. Работа блока таймера временных отсчетов разрешается битом UTE регистра QEPCNTR. Блок-схема приведена на рисунке 100.

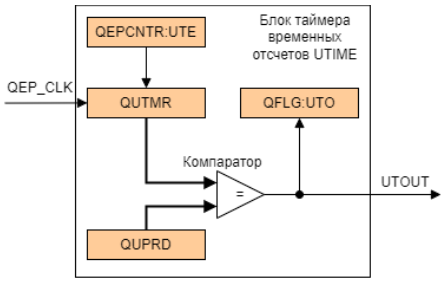



Рисунок 100 – Структурная схема блока таймера временных отсчетов

При равенстве значения таймера временных отсчетов QUTMR значению периода QUPRD формируются флаг прерывания UTO и сигнал UTOUT.

По сигналу UTOUT осуществляется:

- сброс счетчика QPOSCNT блока PCCU, если сконфигурирован сброс от таймера временных отсчетов;
- сохранение значения регистра QPOSCNT блока PCCU в регистр QPOSLAT;
- сохранение регистров таймера захвата QCTMR и периода захвата QCPRD блока UTIME в регистры QCTMRLAT и QCPRDLAT, если сконфигурировано сохранение по событию от блока UTIME;
- формирование прерывания, если разрешено прерывание UTO в регистре QEINT.

### 13.3.5 *Блок сторожевого таймера QWDOG*

Блок сторожевого таймера предназначен для детектирования отсутствия переключения сигнала счета QCLK в течение времени, определяемого значением QWDTMR, и представляет собой 16-разрядный счетчик, инкрементирующийся на деленной на 64 частоте QEP_CLK и сбрасываемый сигналом QCLK. Структурная схема приведена на рисунке 101. Работа блока разрешается установкой бита WDE регистра QEPCNTR.

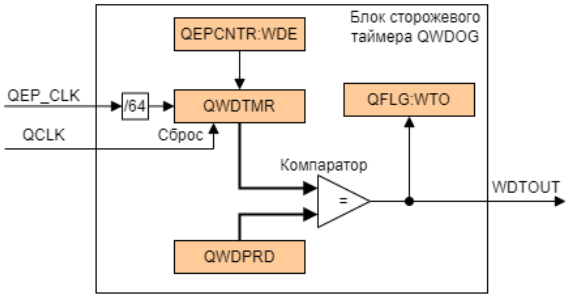



Рисунок 101 – Структурная схема блока сторожевого таймера

При равенстве значения сторожевого таймера QWDTMR значению периода QWDPRD формируются флаг прерывания WTO и сигнал WDTOUT.

### 13.3.6 *Прерывания и флаги состояния*

Контроллер квадратурных преобразователей может формировать сигнал запроса прерывания INT_QEP по следующим 11 событиям:

- PHE – ошибка фазы входных сигналов квадратурного декодера QDU;
- PCE – ошибка значения счетчика положения PCCU при сбросе по событию полного оборота;
- QDC – смена направления движения счетчика положения PCCU;
- PCO – переполнение счетчика положения PCCU (QPOSCNT ≥ QPOSMAX) при счете вверх;
- PCU – переполнение счетчика положения PCCU (QPOSCNT = 0) при счете вниз;
- PCR – готовность к отложенной записи значения сравнения из регистра QPOSCMP компаратора положения;
- PCM – успешное сравнение значения QPOSCNT счетчика таймера PCCU с значением сравнения QPOSCMP (QPOSCNT = QPOSCMP);
- SEL – сохранение значения QPOSCNT счетчика таймера PCCU в регистр QPOSSLAT по фронту сигнала стробирования QS;
- IEL – сохранение значения QPOSCNT счетчика таймера PCCU в регистр QPOSILAT по событию полного оборота;
- UTO – достижение счетчиком QUTMR таймера временных отсчетов UTIME значения периода QUPRD;
- WTO – достижение счетчиком QWDTMR сторожевого таймера QWDOG значения периода QWDPRD.

Выбор событий, формирующих запрос прерывания, осуществляется через регистр разрешения прерывания QEINT. Состояние событий прерывания может быть прочитано через регистр статуса прерывания QFLG (доступен только для чтения). События могут быть принудительно установлены или сброшены через регистры QFRC и QCLR соответственно (регистры доступны только для записи). Состояние активного запроса прерывания отражается с помощью флага INT регистра QFLG, данный флаг может быть только сброшен через регистр QCLR.

Также в процессе работы контроллер квадратурных преобразователей заполняет поля статуса в регистре QEPSTS:

- PCEF – флаг ошибки значения счетчика положения PCCU при сбросе по событию полного оборота, обновляемый по каждому событию полного оборота;
- FIMF – флаг захвата первого индексного маркера;
- FIDF – направление вращения при захвате первого индексного маркера;
- CDEF – флаг изменения направления вращения при захвате фронтов;
- COEF – флаг переполнения счетчика захвата фронтов;
- QDFL – направление вращения при захвате каждого индексного маркера;
- QDF – текущее направление вращения;

– UPEVNT – флаг сохранения нового значения счетчика захвата фронтов в регистр QCPRD.

## 13.4 Контроллер захвата (CAP_CNTR)

Функциональное назначение блоков захвата:
– измерение скорости вращающихся машин;
– измерение времени между импульсами датчиков положения;
– измерение периода и скважности периодических сигналов.

Основные особенности:
– подключение к шине APB;
– 32-разрядный счетчик;
– рабочая частота 150 МГц (максимальное разрешение 6,67 нс);
– четыре 32-разрядных регистра событий захвата (CAP1 – CAP4);
– выбор полярности для событий захвата;
– возможность выдачи прерывания по событиям;
– возможность работы как в однократном, так и циклическом режиме захвата;
– возможность захвата как абсолютных, так и относительных значений;
– формирование сигнала выходной синхронизации;
– возможность работы в режиме вспомогательного ШИМ-генератора (auxiliary PWM, APWM) с возможностью выбора полярности и отложенной загрузки;
– возможность входной синхронизации в режиме APWM.

В микросхеме реализовано 4 блока захвата. Структурная схема блока захвата приведена на рисунке 102.

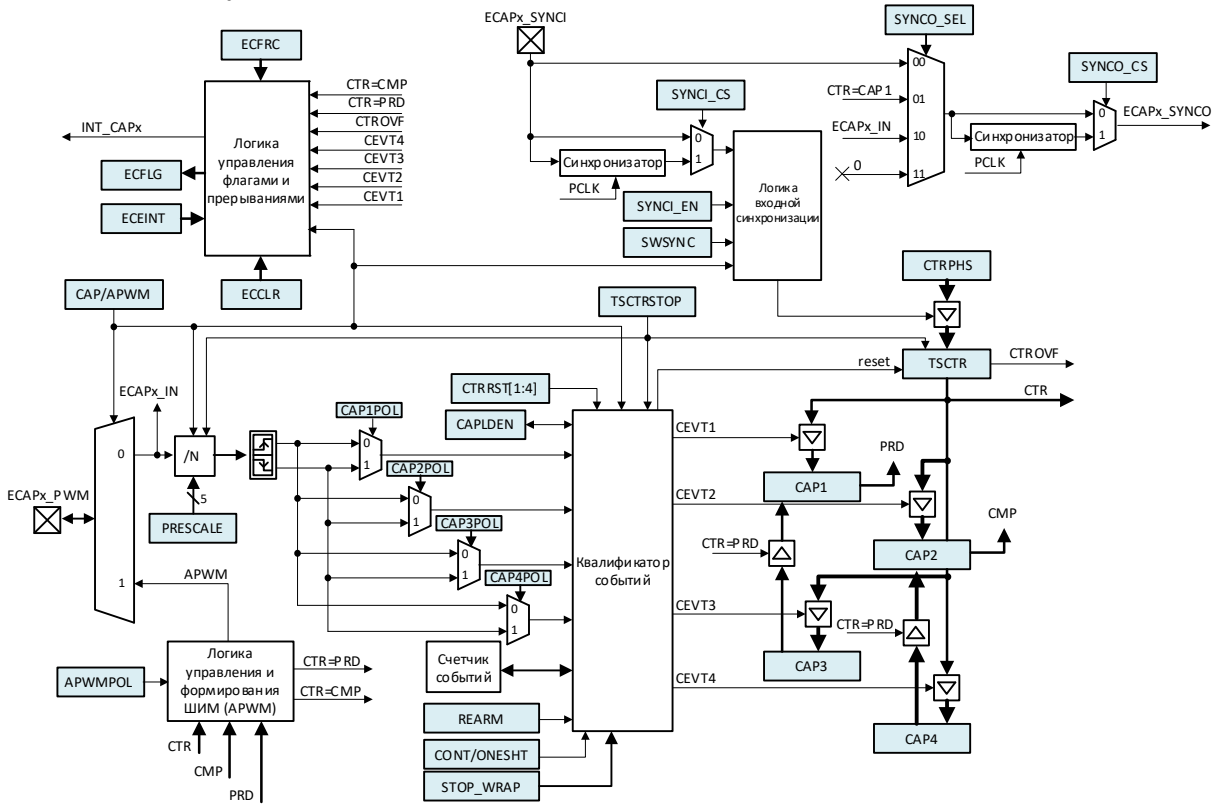



Рисунок 102 – Структурная схема блока захвата CAP

Блок захвата тактируется частотами PCLK и CAP_CLK. Частота PCLK используется при записи в регистры блока по шине APB и для пересинхронизации сигналов синхронизации SYNCI и SYNCO. При записи в регистры блока по шине APB осуществляется пересинхронизация на частоту CAP_CLK.

Каждый из блоков захвата имеет вывод ECAPx_PWM, соединенный с выводом общего назначения микросхемы в цифровой функции ECAPx_PWM. В зависимости от режима работы блока захвата, вывод ECAPx_PWM является входом измеряемого сигнала ECAP_IN или выходом сигнала APWM ШИМ.

### 13.4.1 *Режим захвата внешних событий*

Режим захвата функционирует по умолчанию. Регистры CAP1-CAP4 служат для захвата значения таймера CTR по входным событиям.

#### 13.4.1.1 Делитель входного сигнала и детектор фронтов

В случае, если события приходят слишком часто и требуется уменьшить их частоту, используется предварительный делитель событий (импульсов), осуществляющий прореживание сигнала. В предварительном делителе используется счетчик, который формирует один импульс на каждые PRESCALE+1 импульсов (см. рисунок 103). Делитель программируется битами PRESCALE регистра ECCTL1. Значение 0x00 запрещает прореживание импульсов – в таком случае входной сигнал поступает на детектор фронтов напрямую.

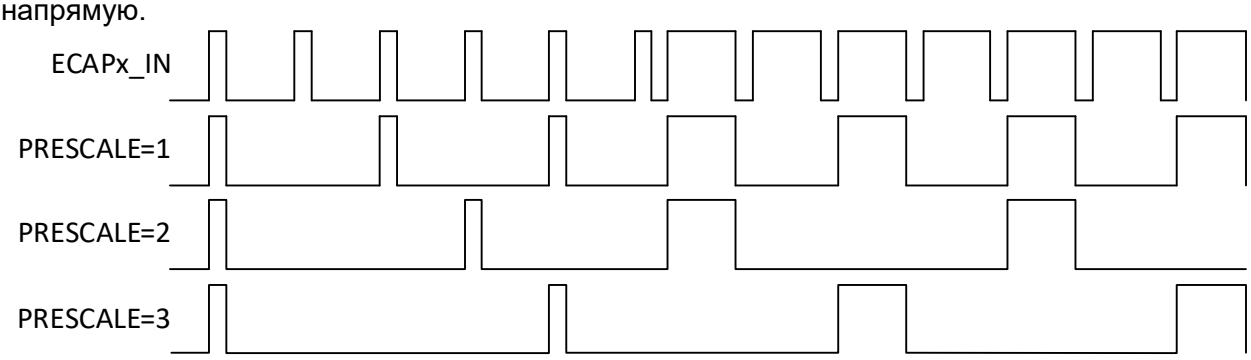



Рисунок 103 – Сигнал на выходе делителя в зависимости от значения PRESCALE

Детектор фронтов определяет фронт сигнала через 3 периода рабочей частоты CAP_CLK после формирования фронта. Необходимо учитывать, что, если после сброса на вход ECAP_xPWM блока поступает постоянный уровень логической 1, то при включении частоты CAP_CLK через 3 периода детектором фронтов будет определен передний фронт.

Детектор фронтов и делитель входного сигнала функционируют всегда при наличии частоты CAP_CLK, если выбран режим захвата и основной таймер не остановлен.

При остановке основного счетчика (бит TSCTRSTOP) или при переходе в режим APWM (бит CAP/APWM) детектор фронтов и делитель входного сигнала сбрасываются.

#### 13.4.1.2 Захват событий и счетчик событий

Для каждого из четырех событий CEVT1-CEVT4 выбирается полярность захватываемого сигнала (передний или задний фронт) битами CAP1POL-CAP4POL.

По событиям CEVT1-CEVT4 осуществляется захват значения основного таймера в соответствующие регистры CAP1-CAP4, если загрузка регистров захвата по событию захвата разрешена битом CAPLDEN регистра ECCTL1. Регистры захвата событий CAP1-CAP4 доступны для записи и для чтения. Если загрузка запрещена (CAPLDEN=0), значения основного таймера не будут захватываться в регистры CAP1-CAP4 по событиям захвата, но сами события CEVT1-CEVT4 будут формироваться.

Для захвата относительных значений времени между событиями возможно осуществлять сброс основного таймера при возникновении событий после захвата событий в регистры CAP1-CAP4. Программно настраивается с помощью битов CTRRST1-CTRRST4, по умолчанию сброс отключен. Сброс основного таймера в режиме захвата по событиям CEVT1-CEVT4 осуществляется, если событие захвачено, разрешена загрузка регистров захвата, установлен соответствующий бит разрешения сброса CTRRST1-CTRRST4 и основной счетчик не остановлен.

Выбор, какое из событий на данный момент должно формироваться, осуществляется на основании значения счетчика событий – значение счетчика

соответствует номеру обработанного события CEVT1-CEVT4. Счетчик событий инкрементируется, если для события с номером, соответствующим текущему значению счетчика, формируется выбранный фронт захватываемого сигнала – при этом инкрементируется также независимо от разрешения загрузки регистров захвата CAPLDEN.

При остановке основного счетчика или при переходе в режим APWM счетчик событий сбрасывается. Также возможно осуществить программный сброс счетчика событий записью 1 в бит REARM регистра ECCTL2 – при этом одновременно аппаратно бит CAPLDEN устанавливается в 1 и разрешается загрузка регистров захвата по событиям захвата. Аппаратно установленный в 1 бит CAPLDEN можно программно сбросить в 0.

#### 13.4.1.3 Режим циклического захвата

Блок захвата может функционировать в режиме циклического и однократного захвата, режим работы выбирается битом CONT/ONESHT регистра ECCTL2. Режим циклического захвата функционирует по умолчанию. При переключении между режимами предделитель входного сигнала, детектор фронтов и счетчик событий не сбрасываются.

В циклическом режиме счетчик событий последовательно обрабатывает входящие события с счетом функционала, описанного в пункте Захват событий и счетчик событий.

#### 13.4.1.4 Режим однократного захвата

Однократный захват выбирается записью единицы в бит CONT/ONESHT регистра ECCTL2. Далее необходимо указать количество подсчитываемых событий (от одного до четырех) с помощью счетчика событий – задается битами STOP_WRAP регистра ECCTL2. После этого следует перезапустить счетчик событий записью единицы в бит REARM, что, в свою очередь, приведет к аппаратной установке бита CAPLDEN.

Как только количество событий (значение счетчика событий) совпадет со значением, задаваемым битами STOP_WRAP, бит CAPLDEN аппаратно сбрасывается.

Для повторного запуска однократного захвата следует повторить последовательность действий.

В режиме однократного захвата управление битом CAPLDEN осуществляется аппаратно, поэтому программно его модифицировать не рекомендуется, так как формирование событий CEVT1-CEVT4 зависит от значения бита CAPLDEN, а значение счетчика событий от него не зависит.

Также не рекомендуется изменять значение STOP_WRAP в процессе осуществления однократного захвата, так как сравнение количества событий с задаваемым битами STOP_WRAP значением осуществляется проверкой равенства.

#### 13.4.1.5 Основной счетчик

Основной счетчик CTR, доступ к которому со стороны системы осуществляется через регистр TSCTR, инкрементируется на частоте CAP_CLK от 0x0000_0000 до 0xFFFF_FFFF, после чего счетчик сбрасывается. Также в режиме захвата основной счетчик может быть сброшен по событиям CEVT1-CEVT4 при соответственно установленным CTRRST1-CTRRST4. Возможна запись значения в счетчик CTR через регистр TSCTR.

Возможно программно запускать и останавливать счетчик CTR с помощью бита TSCTRSTOP регистра ECCTL2. По умолчанию основной счетчик остановлен.

### 13.4.2 *Режим вспомогательного ШИМ (APWM)*

Блок захвата может функционировать в режиме вспомогательного одноканального 32-разрядного генератор ШИМ (auxiliary PWM, APWM). Данный режим выбирается установкой бита CAP/APWM регистра ECCTL2. Вывод ECAPx_PWM блока при этом начинает функционировать как выход.

#### 13.4.2.1 Основной счетчик и регистры CAP1-CAP4

Основной счетчик CTR сохраняет свое значение, которое было у него до перехода в режим APWM. В режиме APWM при работе основной счетчик инкрементируется от 0x0000_0000 до значения, задаваемого регистром CAP1 (PRD). При совпадении значения основного счетчика со значением PRD счетчик сбрасывается. В режиме APWM основной

счетчик также поддерживает отложенную загрузку значения из регистра CTRPHS по событию входной синхронизации и запись в регистр TSCTR основного счетчика. Если в процессе работы основного счетчика записываемое в него значение больше или равно PRD, счетчик также сбрасывается.

Регистры CAP1-CAP4 в режиме APWM выступают в качестве регистров периода и сравнения основного 32-разрядного счетчика.

Регистр CAP1 является регистром периода таймера (PRD), а регистр CAP2 – регистром сравнения (CMP). Регистры CAP3 и CAP4 являются регистрами отложенной загрузки для регистров CAP1 и CAP2, соответственно. Как только значение счетчика таймера достигает значения периода CAP1, возникает событие CTR = PRD, по которому происходит сброс таймера и перегрузка значений из CAP3 и CAP4 в регистры CAP1 и CAP2. До запуска таймера следует инициализировать значения регистров CAP1-CAP4. Регистры CAP1-CAP4 доступны для чтения и записи.

#### 13.4.2.2 Формирование ШИМ-сигнала

После инициализации регистров CAP1 (PRD) и CAP2 (CTR) и запуска счетчика CTR значение основного счетчика сравнивается одновременно со значением PRD регистра периода CAP1 и значением CMP регистра сравнения CAP2.

При CTR≤CMP на выходе блока сигнал ECAP_xPWM находится в нуле. При CTR>CMP сигнал переводится в единицу и удерживается до тех пор, пока основной счетчик не достигнет значения периода. С помощью бита APWMPOL можно изменить полярность выходного сигнала.

При достижении основным счетчиком периода осуществляется запись в регистры CAP1 и CAP2 из регистров отложенной загрузки CAP3-CAP4.

На рисунке 104 представлен пример формирования сигнала ШИМ с использованием отложенной загрузки и разной полярности сигнала.

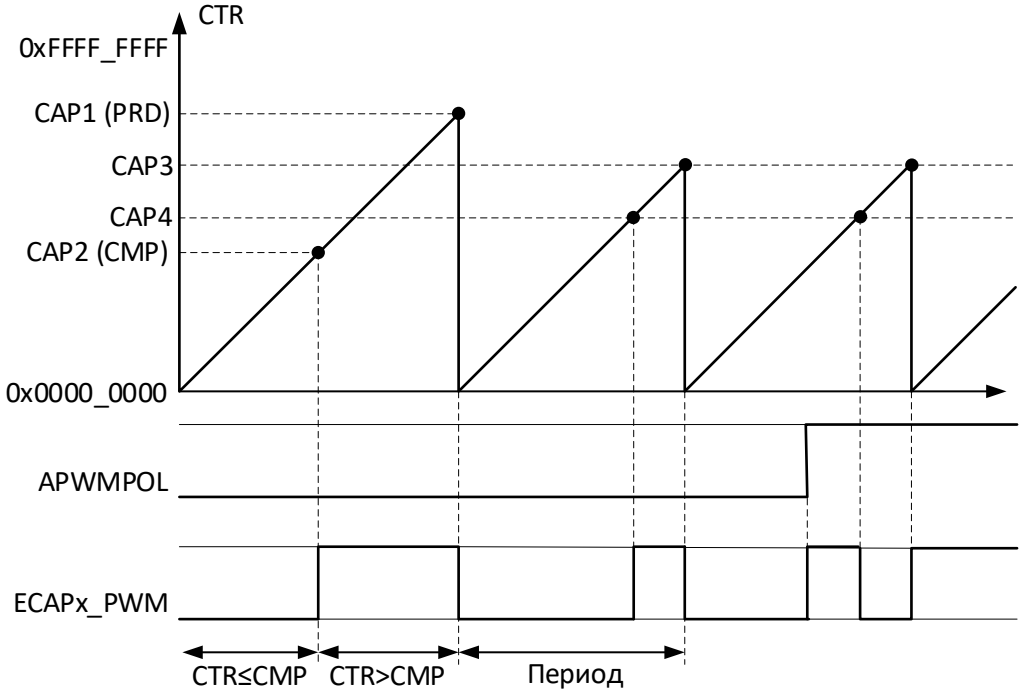



Рисунок 104 – Формирование сигнала ШИМ блоком захвата в режиме APWM

#### 13.4.2.3 Входная и выходная синхронизация

Для синхронизации основного счетчика с другими блоками захвата в микросхеме и другими микросхемами в блоке захвата для режима APWM предусмотрена возможность загрузки значения из регистра CTRPHS в основной счетчик по событию входной синхронизации.

Блок захвата предусматривает прием входной синхронизации (сигнал SYNCI) и формирование выходной синхронизации (сигнал SYNCO). Сигналы входной и выходной

синхронизации блоков захвата в микросхеме включены по цепочке (рисунок 105). Сигнал SYNCI для нулевого блока захвата в цепочке и сигнал SYNCO для последнего блока захвата в цепочке выведены как цифровые функции выводов микросхемы.

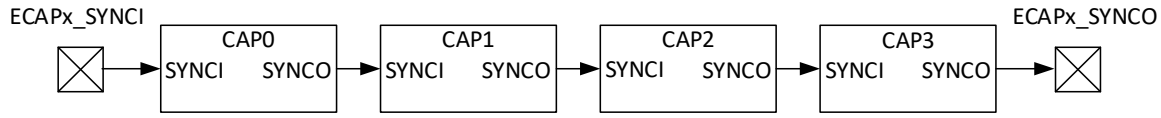



Рисунок 105 – Синхронизация блоков захвата в микросхеме

Управление возможностью входной аппаратной синхронизации осуществляется битом SYNCI_EN регистра ECCTL1. Возможна пересинхронизация входного сигнала SYNCI на частоту PCLK (управляется битом SYNCI_CS регистра ECCTL1). Предусмотрена возможность программной синхронизации с помощью бита SWSYNC регистра ECCTL2. Загрузка из регистра CTRPHS по программной и аппаратной входной синхронизации возможна только в режиме APWM.

Сигнал выходной синхронизации SYNCO может формироваться, в зависимости от поля SYNCO_SEL регистра ECCTL2, из входной синхронизации SYNCI (без учета пересинхронизации), из условия равенства значения счетчика CTR и регистра CAP1 (единица при CTR=CAP1, ноль – в других случаях) и из входного сигнала ECAPx_IN. Выбранный выходной сигнал SYNCO, аналогично SYNCI, также может быть пересинхронизирован на частоту PCLK (управляется битом SYNCO_CS регистра ECCTL1). Выходная синхронизация функционирует в любом режиме работы.

### 13.4.3 *Прерывания*

Блок захвата может формировать запросы прерывания INT_CAPx по 7 возможным событиям:

- CEVT1-CEVT4, события захвата при работе в режиме захвата;
- CTROVF, событие достижения основным счетчиком максимального значения 0xFFFF_FFFF;
- CTR=PRD, событие равенства значения основного счетчика значению регистра CAP1 (PRD) при работе в режиме ШИМ;
- CTR=CMP, событие равенства значения основного счетчика значению регистра CAP2 (CMP) при работе в режиме ШИМ.

Состояние событий может быть прочитано через регистр статуса прерывания ECFLG. Сброс событий осуществляется через регистр сброса прерывания ECCLR. Принудительное программное формирование событий возможно через регистр установки прерывания ECFRC. Регистр статуса доступен только для чтения, регистры сброса и установки не являются самоочищающимися. Также существует возможность указать, по каким из событий разрешено формирование запроса прерывания, с помощью регистра маски источников прерывания ECEINT – по умолчанию для всех событий формирование запроса прерывания запрещено.

# 14 Контроллеры аналоговых и цифро-аналоговых блоков

## 14.1 Контроллер АЦП (ADC_CNTR)

Контроллер АЦП управляет сборкой из двух АЦП, представленных на рисунке 106 как АЦП0 и АЦП1, и обеспечивает следующие возможности:

1 На вход каждого АЦП с помощью мультиплексора MUX может быть мультиплицирован любой из девяти p-каналов и любой из девяти n-каналов (n-канал нужен при использовании дифференциального режима).

2 Для каждого АЦП в контроллере предусмотрен независимый автомат управления (АУ), причем:

– каждый из АУ может автоматически выполнять цепочку преобразований от 1 до 32 преобразований;
– запуск цепочек преобразований возможен как по событию запуска, так и в непрерывном режиме.

3 Запуск цепочки преобразования можно осуществить:

– программно;
– по событию блоков ШИМ;
– по внешним событиям.

4 Для сборки из двух АЦП предусмотрена возможность запуска цепочек преобразования:

– независимо;
– одновременно;
– попеременно;
– со сдвигом на значение, заданное в тактах рабочей частоты АЦП.

5 Результаты цепочек преобразований сборки из двух АЦП раскладываются в 64 независимых регистра.

6 Чтение результатов может производится в одном из двух режимов:

– режим регистров, когда результат каждого преобразования находится в выделенном регистре;
– режим FIFO, когда результаты преобразований читаются последовательно, по одному адресу.

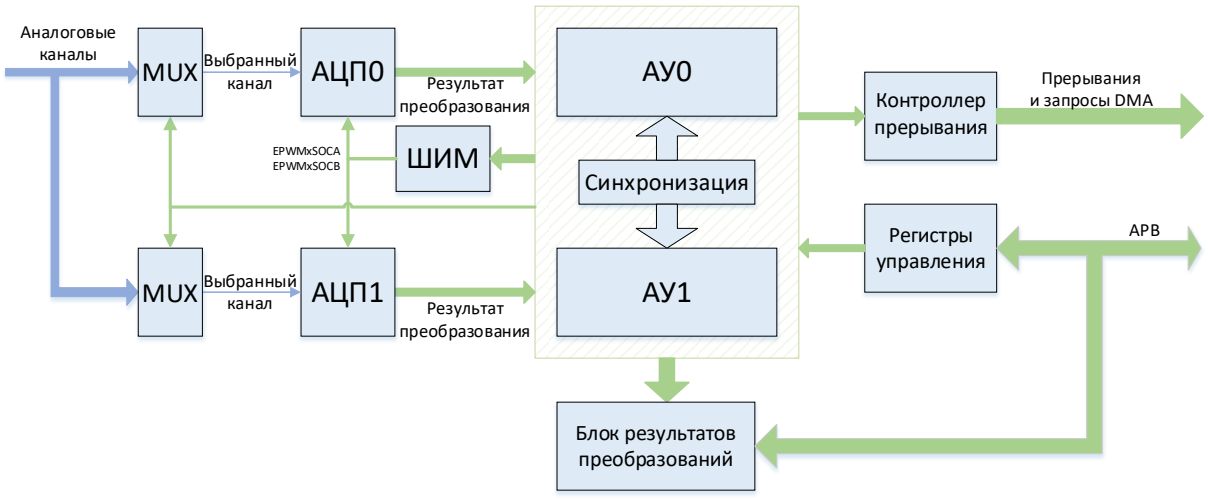



Рисунок 106 – Структурная схема контроллера АЦП

Структура АЦП представлена на рисунке 107 и включает в себя компаратор, два ЦАП на конденсаторах и логику.

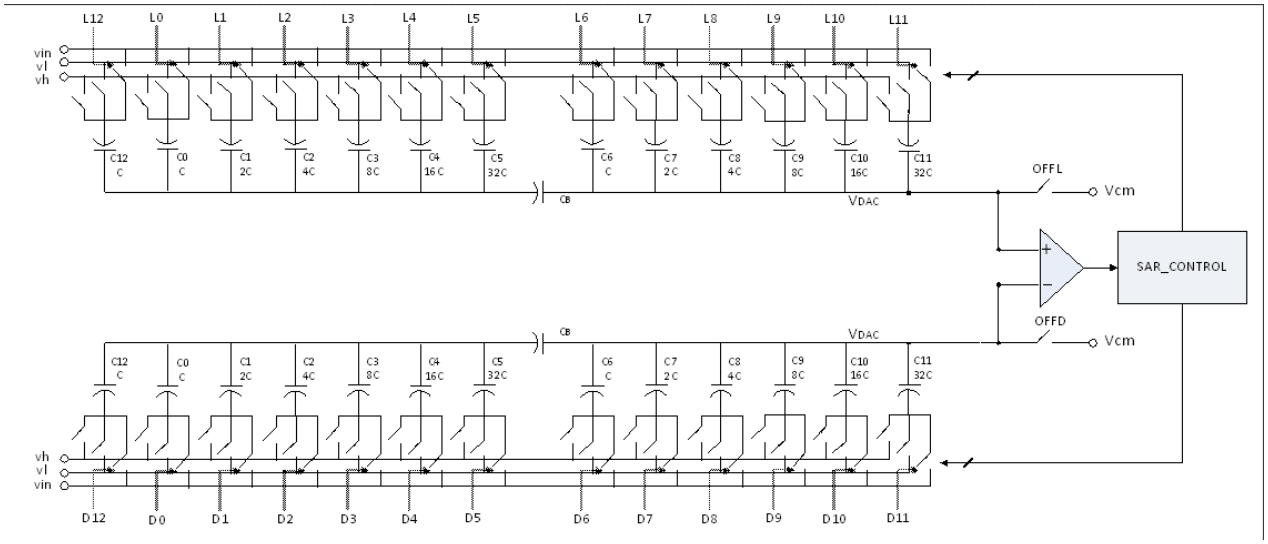



Рисунок 107 – Структура АЦП

### 14.1.1 *Запуск цепочек преобразований*

**Регистр ADCx_CNTR, поле EN_ADCx:**

Перед началом работы с АЦП необходимо установить бит разрешения работы EN_ADCx. При этом происходит включение как цифровых контроллеров, так и аналоговых преобразователей.

**Регистр ADCx_CNTR, поле RST_ADCx:**

Для сброса только цифровых контроллеров реализован бит RST_ADCx.

**Регистр ADCx_CNTR, поле STARTofCONV_ADCx:**

Запуск цепочки преобразований для каждого из АЦП осуществляется установкой бита STARTofCONV_ADCx. Установка бита STARTofCONV_ADCx может быть выполнена как программно, так и аппаратно, по соответствующим сигналам от блоков ШИМ или на выводах микроконтроллера. Алгоритм работы контроллера не зависит от способа установки этого бита.

**Регистр ADCx_CNTR, поле CONV_TRIG_CNTR_ADCx:**

Источник аппаратной установки бита STARTofCONV_ADCx выбирается полем CONV_TRIG_CNTR_ADCx. При появлении на входе соответствующего сигнала, бит STARTofCONV_ADCx устанавливается в единицу автоматически.

**Регистр ADCx_CNTR, поле ETRNL_CONV_MODE_ADCx:**

Преобразования могут выполняться как в непрерывном, так и в однократном режимах, что регулируется битом ETRNL_CONV_MODE_ADCx.

**Регистр ADCx_CNTR, поле MAXCNV_ADCx:**

В однократном режиме контроллер выполняет заданное в поле MAXCNV_ADCx количество преобразований, после чего останавливается. Для запуска следующей цепочки преобразований необходимо снова установить бит STARTofCONV_ADCx. В непрерывном режиме контроллер циклически (снова и снова) выполняет заданное в поле MAXCNV_ADCx количество преобразований, перезаписывая предыдущие результаты. При этом бит STARTofCONV_ADCx остается активным до его выключения и может быть сброшен только программно.

### 14.1.2 *Выбор канала для преобразования*

**Регистр ADCx_CHSEL, поле CONVxx:**

Для каждого АЦП из сборки есть возможность выбора до девяти p- и n-каналов. Регистр ADC0_CHSEL отвечает за выбор каналов во время цепочки преобразований в АЦП0, а регистр ADC1_CHSEL, соответственно, в АЦП1. Максимально может быть выполнено до 32 преобразований на каждом из АЦП, причем выбор p- и n-каналов на каждом из этих преобразований осуществляется полем CONVxx. В каждом из полей старшие четыре бита определяют n-канал, младшие четыре бита определяют p-канал. Если преобразование осуществляется не в дифференциальном режиме, старшие четыре бита не используются. Например, для ADC11 установка в регистре ADC1_CHSEL поля CONV02 в 0010_0001 будет означать, что в цепочке преобразований на третьем по счету преобразовании n-канал будет подключен к PC15, а p-канал к PC13.

Таблица 58 – Таблица распределения каналов по блокам АЦП

| **Номер блока АЦП** | **Тип канала** | **Номер канала** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| ADC00 | CHP | PC10 | PC9 | PC8 | PC6 | PC5 | PC4 | PC3 | PC2 | PC0 |
| | CHN | PC11 | NC | NC | PC7 | NC | NC | NC | NC | PC1 |
| ADC10 | CHP | PC10 | PC9 | PC8 | PC6 | PC5 | PC4 | PC3 | PC2 | PC0 |
| | CHN | PC11 | NC | NC | PC7 | NC | NC | NC | NC | PC1 |
| ADC01 | CHP | NC | PC21 | PC19 | PC18 | PC17 | PC16 | PC14 | PC13 | PC12 |
| | CHN | NC | NC | PC20 | NC | NC | NC | PC15 | NC | NC |
| ADC11 | CHP | NC ТД* | PC21 | PC19 | PC18 | PC17 | PC16 | PC14 | PC13 | PC12 |
| | CHN | NC | NC | PC20 | NC | NC | NC | PC15 | NC | NC |
| ADC02 | CHP | NC | PC31 | PC30 | PC28 | PC27 | PC26 | PC25 | PC23 | PC22 |
| | CHN | NC | NC | NC | PC29 | NC | NC | NC | PC24 | NC |
| ADC12 | CHP | NC ТД* | PC31 | PC30 | PC28 | PC27 | PC26 | PC25 | PC23 | PC22 |
| | CHN | NC | NC | NC | PC29 | NC | NC | NC | PC24 | NC |

* ТД – температурный датчик

Часть каналов блоков АЦП не подключено к выводам микросхемы, в этом случае в соответствующей ячейке таблицы указано значение «NC».

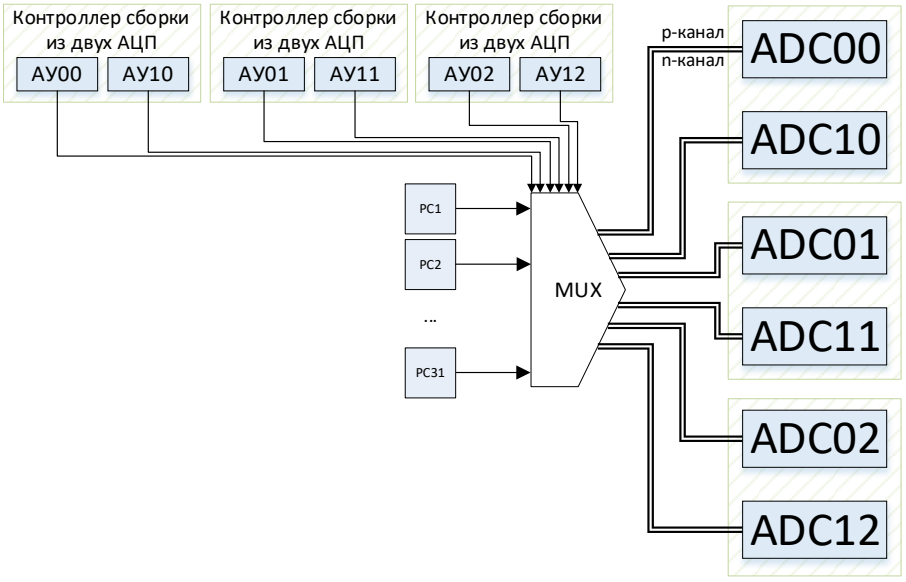



Рисунок 108 – Структурная схема контроллера АЦП

### 14.1.3 *Автомат управления (АУ) последовательностью преобразований*

Автоматы управления запускают автоматические последовательности преобразований с сохранением всех результатов, что позволяет сократить время между

отдельными преобразованиями и освободить вычислительные возможности центрального процессора. В каждом АУ реализовано четыре состояния:

1 Начальное состояние:

После сброса АУ переходит в начальное состояние и ожидает установки бита начала преобразования STARTofCONV_ADCx в регистрах ADC0_CNTR или ADC1_CNTR для АЦП0 и АЦП1 соответственно. Перед запуском преобразования можно настроить синхронизацию, определяемую полем SYNCofCONV_MODE_ADCx регистра ADC_SYNC_ CNTR. Если синхронизация не настроена, флаг SYNCofCONV_in всегда установлен в «1» и не препятствует началу преобразования.

2 Синхронизация:

После получения сигнала STARTofCONV_ADCx автомат переходит в состояние ожидания сигнала синхронизации преобразования. Также в этом состоянии происходит переключение мультиплексора на другой канал, из-за чего полное время заряда конденсатора может быть больше, чем DELAY_TIME_ADC0/DELAY_TIME_ALT. Полное время заряда дополнительно регулируется полями DELAY_TIME_ADC0/DELAY_TIME_ALT регистра ADCx_CNTR.

3 Заряд конденсатора:

Следующим состоянием автомата после получения сигнала синхронизации является состояние заряда входного конденсатора. В этом состоянии запускается счетчик количества тактов преобразования. Максимальное значение счетчика контролируется полями DELAY_TIME_ADC0/DELAY_TIME_ALT регистра ADCx_CNTR.

4 Оцифровка сигнала:

По достижению счетчиком значения DELAY_TIME_ADC0/DELAY_TIME_ALT, требуемого для заряда конденсатора, происходит переход в состояние оцифровки сигнала. По завершинию оцифровки происходит переход либо в начальное состояние, либо в состояние синхронизации в зависимости от режима работы автомата и количества обработанных преобразований. Режим работы автомата (однократный или непрерывный) контролируется битом ETRNL_CONV_MODE_ADCx регистра ADCx_CNTR.

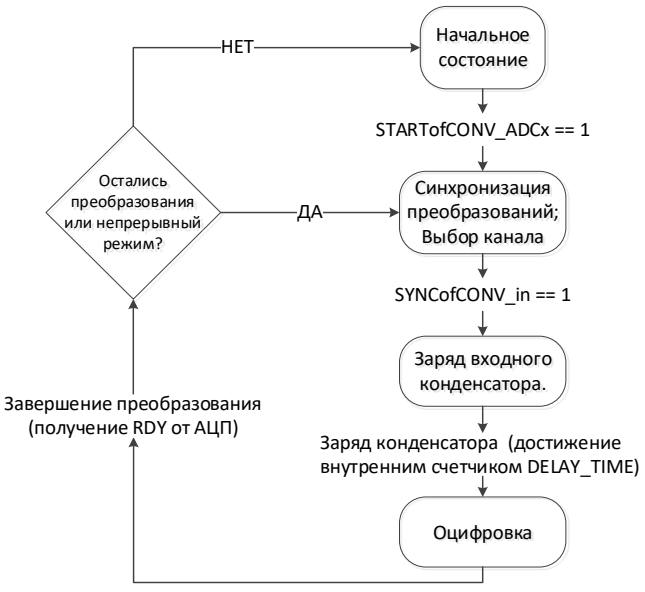



Рисунок 109 – Автомат управления последовательностью преобразований

На рисунке 110 представлена диаграмма задержек сигналов АЦП. На ней указаны следующие задержки:

- Td1 – задержка ожидания сигнала синхронизации. Регулируется регистром ADC_SYNC_CNTR;
- Td2 – заряд конденсатора. Регулируется полями DELAY_TIME_ADC0 /DELAY_TIME_ALT регистра ADCx_CNTR;

- Td3 – время преобразования (от 64 тактов CLK и выше). Регулируется регистром ADC_SYNC_CNTR;
- Td4 – время между запуском цепочек преобразований. Регулируется регистром ADCx_CNTR;
- T_s – время оцифровки сигнала (не меняется и составляет 64 такта CLK).

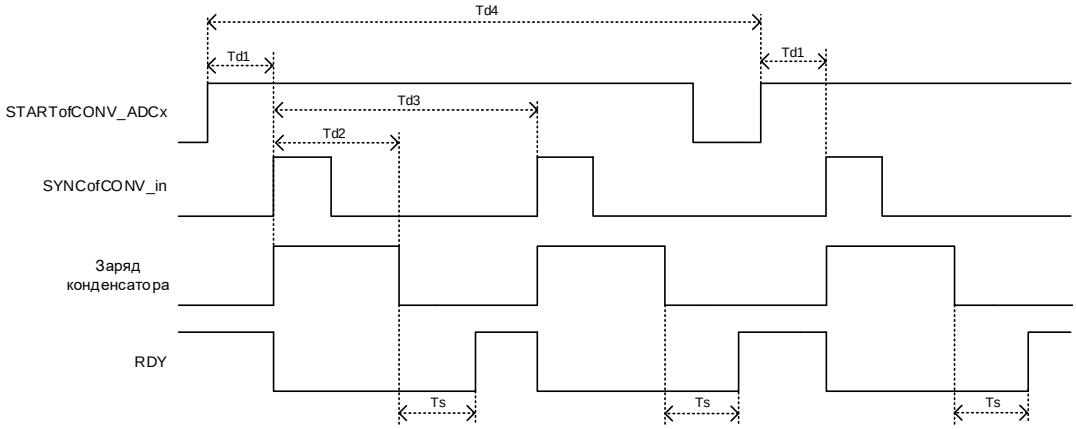



Рисунок 110 – Диаграмма задержек сигналов АЦП

### 14.1.4 *Синхронизация преобразований*

Для расширения возможностей работы со сборкой из 2-х АЦП в контроллере реализованы различные режимы синхронизации запуска преобразований. Для реализации механизма синхронизации в каждом АУ реализованы вход SYNCofCONV_in и выход SYNCofCONV_out:

- Входы SYNCofCONV_in служат для синхронизации запуска АУ1 и АУ2 в контроллере;
- Выход SYNCofCONV_out определяет смещение по тактам начала преобразования соседнего АУ в сборке относительно текущего (в режиме синхронизации с задержкой по фазе).

Синхронизация между АУ контролируется полями SYNCofCONV_in_MODE_ADCx и SYNCofCONV_out_DELAY_TIME_ADCx регистра ADC_SYNC_CNTR и включает в себя следующие режимы:

- Независимый режим (SYNCofCONV_in_MODE_ADCx = 000). Соответствующий этому режиму АУ переходит в состояние заряда конденсатора без задержек;
- синхронные преобразования (SYNCofCONV_in_MODE_ADCx = 011). Если (хоть в одном/в обоих одновременно) из полей SYNCofCONV_in_MODE_ADCx установлено «011», то преобразования обоих АЦП синхронизированы по готовности сигналов SYNCofCONV_in в обоих АУ;
- Попеременные преобразования (SYNCofCONV_in_MODE_ADCx = 100). В этом режиме сигналом старта преобразования для текущего АЦП служит сигнал окончания преобразования соседнего АЦП. Для попеременного выполнения преобразований требуется одновременное включение режима попеременных преобразоываний для обоих АЦП в сборке;
- Преобразования с задержкой по фазе относительно соседнего АЦП (SYNCofCONV_in_MODE_ADCx = 001). В этом режиме сигналом старта преобразования для АЦП является сигнал SYNCofCONV_out соседнего АЦП. В контроллере реализована возможность управления временем выдачи сигнала SYNCofCONV_out для каждого из АЦП в периодах тактового сигнала при помощи поля SYNCofCONV_out_DELAY_TIME_ADCx. Таким образом, возможен режим задержки запуска АЦП, что позволяет реализовать высокопроизводительный конвейерный АЦП;

– Внешняя синхронизация преобразований (SYNCofCONV_in_MODE_ADCx = 010). В этом режиме сигналом старта преобразования для АЦП является входной сигнал SYNCofCONV_in_ext. Сигнал SYNCofCONV_in_ext может быть подан как с внешних выводов, так и с других сборок АЦП.

На выходе каждой сборки есть сигнал ADC_SYNC_OUT, который может быть подан на вход следующей сборки через один из выводов общего назначения, в соответствии с таблицей распределения функций выводов общего назначения. Такой функционал позволит синхронизировать все имеющиеся АЦП.

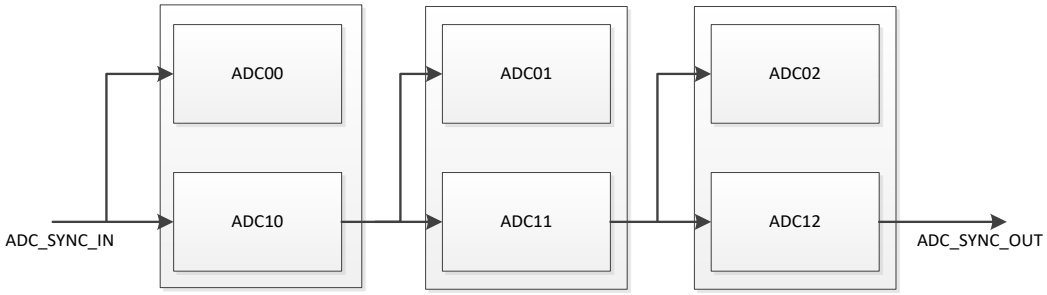



Рисунок 111 – Цепочка синхронизации блоков контроллеров АЦП

### 14.1.5 *Запись результатов преобразований*

Запись результатов преобразований осуществляется в буфер результатов, который может быть организован как пространство по 32 независимых регистра для каждого АЦП, либо как два независимых FIFO буфера. Способ организации буфера результатов контролируется полями BUF_MODE_ADCx регистра ADC_BUF_CNTR и включает в себя два варианта:

– Режим независимых регистров, когда запись результата в регистр происходит по соответствующему номеру в цепочке преобразования, то есть результат первого преобразования в последовательности будет записан в регистр ADCx_RESULT_0, а результат i-го преобразования будет записан в регистр ADCx_RESULT_i. Доступ к регистрам со стороны ЦП осуществляется в этом режиме напрямую, то есть получаемые процессором данные соответствуют адресу читаемого регистра. Например, если поля MAXCNV_ADC0 и MAXCNV_ADC1 регистров ADC0_CNTR и ADC1_CNTR равны 0 и 7, соответственно, то будет выполнено одно преобразование блоком АУ0 и восемь преобразований блоком АУ1, а по завершению цепочки преобразований результаты будут записаны в один регистр ADC0_RESULT_0 и восемь регистров ADC1_RESULT_0,..,ADCx_RESULT_7 буфера результатов.
– Режим FIFO, когда 32-разрядные буфера результатов АУ0 и АУ1 организованы как FIFO, то есть доступ по любому из адресов буфера возможен после чтения стоящих перед ним наиболее ранее записаных адресов.

Регистры результатов ADCx_RESULT_i имеют длину 32-разряда и включают в себя: 12-разрядные результаты преобразования, номер преобразования в цепочке, и время преобразования, выраженное в количестве тактов рабочей частоты АЦП. Формат записи результатов представлен в разделе «Описание регистров блока контроллера АЦП».

### 14.1.6 *Калибровка блоков АЦП*

В блоках АЦП предусмотрена возможность как автоматической калибровки ошибки смещения нуля, так и калибровка вручную. Режим калибровки контролируется битами OFFSET_MODE_ADCx регистра ADC_ANALOG_CNTR и включает в себя два варианта:

- Ручная калибровка смещения нуля (OFFSET_MODE_ADCx = 0). В ручном режиме смещение результата осуществляется на значение, указанное в полях OFFSET_ADCx регистра ADC_ANALOG_CNTR.
- Автоматическая калибровка смещения нуля (OFFSET_MODE_ADCx = 1). В автоматическом режиме необходимо запустить процедуру однократного преобразования, во время которого будет осуществлена автоматическая калибровка, и по окончании которого будут выставлены биты ADCx_CALDONE.

### 14.1.7 *Прерывания и запросы DMA*

В контроллере реализованы следующие сигналы прерываний/запросов DMA регистра ADC_BUF_STATE:

- FIFO_FULL_ADCx – сигнал переполнения буфера FIFO. Возникает в том случае, если в режиме FIFO происходит заполнение буфера, то есть произошло 32 записи, и за это время ни одно из записанных значений не было считано;
- FIFOnEMPTY_ADCx – сигнал наличия данных в буфере FIFO. Возникает, если в буфере есть хотя бы одно несчитанное значение;
- FIFO_LIM_ADCx – сигнал достижения пользовательского предела по заполнению буфера. Сигнал возникает в том случае, если количество несчитанных данных в регистре равно значению поля FIFO_LIM_ADCx_INT_EN регистра ADC_INT_CNTR.

Эти сигналы могут быть как источниками прерываний, так и источниками запросов DMA. В случае ненадобности эти сигналы могут быть замаскированы соответствующими значениями регистров ADC_INT_CNTR и ADC_DMA_CNTR.

### 14.1.8 *Работа с температурными датчиками*

В состав АЦП включены два температурных датчика.

Первый температурный датчик подключен к 8 каналу (CHP) ADC11, второй температурный датчик доступен на 8 канале (CHP) ADC12. Выбор температурного датчика для проведения преобразований может осуществляться также в соответствии с таблицей распределения каналов по блокам АЦП.

Для правильной и стабильной работы термодатчика необходимо настроить блок ИОНТ в основной режим согласно таблице 59. Также для точной работы важно обеспечить время не менее 1 мкс между запуском каждого преобразования.

Параметры температурного датчика не регламентируются. В зависимости от необходимой точности может быть достаточно провести градуировку в двух – трех точках. При необходимости более точных измерений необходимо построить градуировочную таблицу. Градуировка производится индивидуально для каждой микросхемы.

### 14.1.9 *Управление источником опорных напряжений и токов (ИОНТ)*

Перед началом работы с блоками АЦП необходимо задать на них опорные напряжения и токи. См. подраздел 14.4 «Контроллер формирования опорных напряжений и токов».

Дополнительно доступен контроль напряжения, которое формируется на внутреннем ИОН. Для этого необходимо произвести настройку ИОН ADC_BG в основной режим работы, а далее активировать буферизацию напряжения на внешний вывод микросхемы REF_ADCx при помощи поля BUFEN внутри контроллера ADC1 каждой из сборок в зависимости от того, на какой сборке АЦП необходим контроль ИОН.

### 14.1.10 *Входные сопротивление и емкость*

На рисунке 112 представлена упрощенная RC-цепочка на любом из сигнальных входов АЦП.

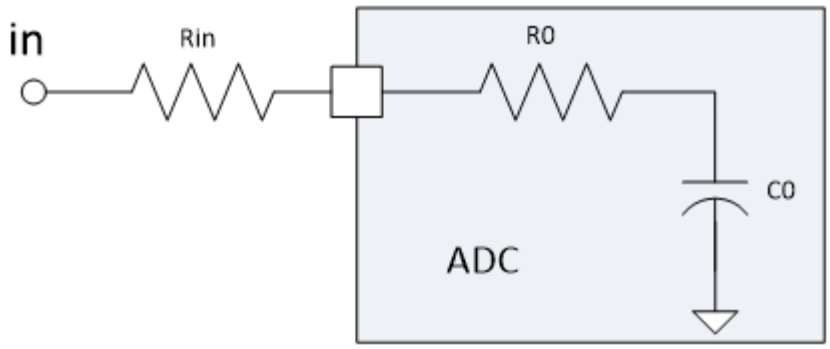



Рисунок 112 – Упрощенная RC-цепочка

Входная емкость C0 равна 14,5 пФ, а внутреннее сопротивление до пада R0 – 750 Ом.

Для обеспечения заявленной точности преобразования (на максимальной частоте семплирования 1 MSPS), необходимо чтобы внешнее сопротивление Rin было не более 800 Ом.

Для обеспечения N-битной точности установления напряжения на входах компаратора, сопротивление ключа должно быть не более

$$\mathrm{Rin} + \mathrm{R0} = \frac{\mathrm{Tsamp}}{[\mathrm{C0} \cdot \ln(2^N)]}, \tag{10}$$

где R0 – эквивалентное внутреннее сопротивление до входов компаратора;

C0 – эквивалентная емкость матрицы конденсаторов;

Tsamp – время фазы выборки (на максимальной частоте).

$$\mathrm{Tsamp} = \mathrm{Tgo} - 52 \bullet \mathrm{Tclk} = 187{,}5\ \text{нс}, \tag{11}$$

где Tclk – период следования тактового сигнала (на максимальной частоте Tclk = 1 мкс/64 = 15,6 нс);

Tgo – период следования сигнала GO (на максимальной частоте всегда GO = 1);

В итоге

$$\mathrm{Rin} \le \frac{\mathrm{Tgo} - 52 \bullet \mathrm{Tclk}}{[\mathrm{C0} \bullet \ln(2^N)]} - \mathrm{R0}. \tag{12}$$

То есть, для обеспечения 12-битной точности на частоте 1 MSPS, сопротивление Rin должно быть не более

$$\mathrm{Rin} = \frac{\mathrm{Tsamp}}{[\mathrm{C0} \bullet \ln(2^{12})]} - \mathrm{R0} = \frac{187{,}5\mathrm{n}}{14{,}52\mathrm{p} \bullet 8{,}318} - 750 = 800\ \text{Ом}. \tag{13}$$

Tclk эквивалентен ADCCLK[2:0];

Tgo задается регистром ADCX_CNTR.

## 14.2 Контроллер ЦАП (DAC_CNTR)

Блок контроллера цифро-аналогового преобразователя позволяет задавать режимы работы 12-разрядного цифро-аналогового преобразователя и передавать данные для преобразований как программно, так и с помощью блока прямого доступа к памяти DMA.

Для корректной работы цифро-аналогового преобразователя должны быть включены соответствующие опорные токи (всегда) и напряжения (при использовании напряжения VREFx в качестве опорного напряжения ЦАП) в блоке ИОНТ ANA_BG.

Передача данных для преобразования осуществляется записью в регистр DAC_DATA. При этом возможна настройка режима обработки данных в одном из режимов:

1 (MODE = 3'b000) Режим задания одиночного значения. В этом режиме данные постоянно записываются и считываются из нулевой ячейки FIFO. Обновление значения на выходе блока происходит только при записи нового значения. При этом происходит обновление счетчика периода. Возможна организация одиночных преобразований, при которой при получении прерывания о достижении периода счета DAC в буфер заносится новое значение для преобразования.
2 (MODE = 3'b001) Режим FIFO с периодическим обновлением. В этом режиме данные записываются в очередь буфера FIFO размером в 32 значения. Данные на выходе ЦАП обновляются при достижении счетчиком периода обновления данных DAC_CNT значения, задаваемого регистром DAC_PRD. При достижении последнего записанного в буфер FIFO значения обновление значений на выходе ЦАП прекращается.
3 (MODE = 3'b010) Режим FIFO с обновлением данных от внешнего сигнала. В этом режиме данные также записываются в очередь буфера FIFO. Выдача очередного значения происходит по событию фронта сигнала внешней синхронизации DAC_SYNC.

Режим MODE = 3'b011 аналогичен режиму MODE = 3'b010. Режимы MODE = 3'b1xx аналогичны режимам MODE = 3'b0xx с тем отличием, что при достижении последнего записанного в буфер FIFO значения обновление значений на выходе ЦАП начинается с нулевого значения (циклично).

32-битный счетчик периода обновления данных DAC_CNT инкрементируется на системной частоте PCLK и обнуляется при достижении периода DAC_PRD.

Бит DAC_EN_D включает работу цифрового контроллера и не влияет на работу аналогового блока преобразователя. При DAC_EN_D = 0 цифровой контроллер выключен, внутренние регистры контроллера остаются в своих предыдущих состояниях, счетчик DAC_CNT не инкрементируется и не обнуляется при равенстве значения периоду DAC_PRD, данные в буфер FIFO не записываются и не обновляются. При этом, если разрешена работа аналогового контроллера (бит DAC_EN_A), то преобразователем будут обрабатываться данные, которые были в буфере на момент выключения цифрового контроллера.

Бит DAC_EN_A включает работу аналогового преобразователя. При DAC_EN_A = 0 аналоговый преобразователь выключен, преобразования прекращаются независимо от состояния цифрового контроллера.

В режиме сброса (бит DAC_RST) все регистры контроллера принимают свое начальное значение, буфер FIFO сбрасывается. Бит самоочищаемый.

Для оповещения процессора о состоянии контроллера ЦАП в блоке реализована возможность формирования сигнала запроса на прерывание INT_FIFO_DACx по одному из трех событий:

- буфер FIFO не полон;
- буфер FIFO пуст;
- количество записей в буфере FIFO ниже предела заполнения, указанного в поле FIFO_LIM регистра FIFO_CNTR.

Маскирование (разрешение или запрет) соответствующих прерываний определяется состоянием регистра DAC_IE.

Также возможно формирования сигнала запроса на прерывание INT_DACx по следующим событиям:

- достижение счетчиком периода обновления данных DAC_CNT значения периода DAC_PRD в режимах, отличных от режима FIFO с обновлением

данных от внешнего сигнала (формирование сигнала запроса прерывания маскируется в регистре DAC_IE);
- фронт сигнала внешней синхронизации в режиме FIFO с обновлением данных от внешнего сигнала (формирование сигнала запроса прерывания маскируется в регистре DAC_IE);
- запись 1 в поле DAC_INT регистра DAC_STS (формирование сигнала запроса прерывания не маскируется).

Контроллер ЦАП может формировать запросы к контроллеру DMA по следующим событиям:
- буфер FIFO не полон;
- буфер FIFO пуст;
- количество записей в буфере FIFO ниже предела заполнения, указанного в поле FIFO_LIM регистра FIFO_CNTR.
- достижение счетчиком периода обновления данных DAC_CNT значения периода DAC_PRD.

Разрешение или запрет соответствующих запросов к контроллеру DMA определяется состоянием регистра DAC_RE.

## 14.3 Контроллер компаратора (COMP_CNTR)

Блок компаратора предназначен для сравнения аналоговых значений. Каждый блок контроллера компаратора управляет одним физическим компаратором. На вход компаратора могут быть поданы данные как с внешних источников, так и с выхода ЦАП.

Выход каждого ЦАП заведён на соответствующий входной канал каждого компаратора:

DAC0 –> COMP0;
DAC1 –> COMP1;
DAC2 –> COMP2;
DAC3 –> COMP3.

Бит CMP_ON включает схему аналогового компаратора. Поля CMP_SELP и CMP_SELN регистра CMP_CNTR позволяют выбрать один из входов для P и N входов компаратора.

Результатом работы компаратора является цепь cmpres, значение которой равно 1, если напряжение на входе P больше напряжения на входе N, и равно нулю в обратном случае. Бит INV регистра CMP_CNTR позволяет инвертировать входное значение цепи cmpres для удобства подключения каналов ко входам компараторов. По фронту цепи результата работы cmpout компаратора в триггере CMP_RES_IS фиксируется единица. Выход триггера последовательно передается на вход триггера, работающего на частоте PCLK.

Значение этого триггера, объединенное по «И» с сигналами разрешения прерываний IE и разрешения ошибки для ШИМ PE, является источником соответствующих событий. Сброс значений может быть осуществлен чтением регистра, если включен бит CMP_CLREN.

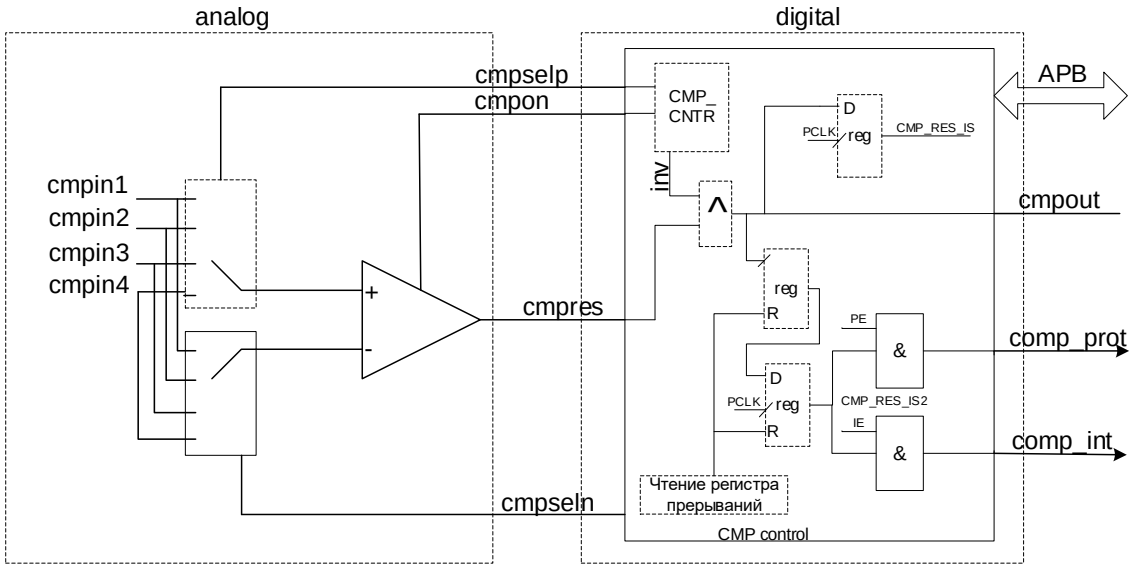
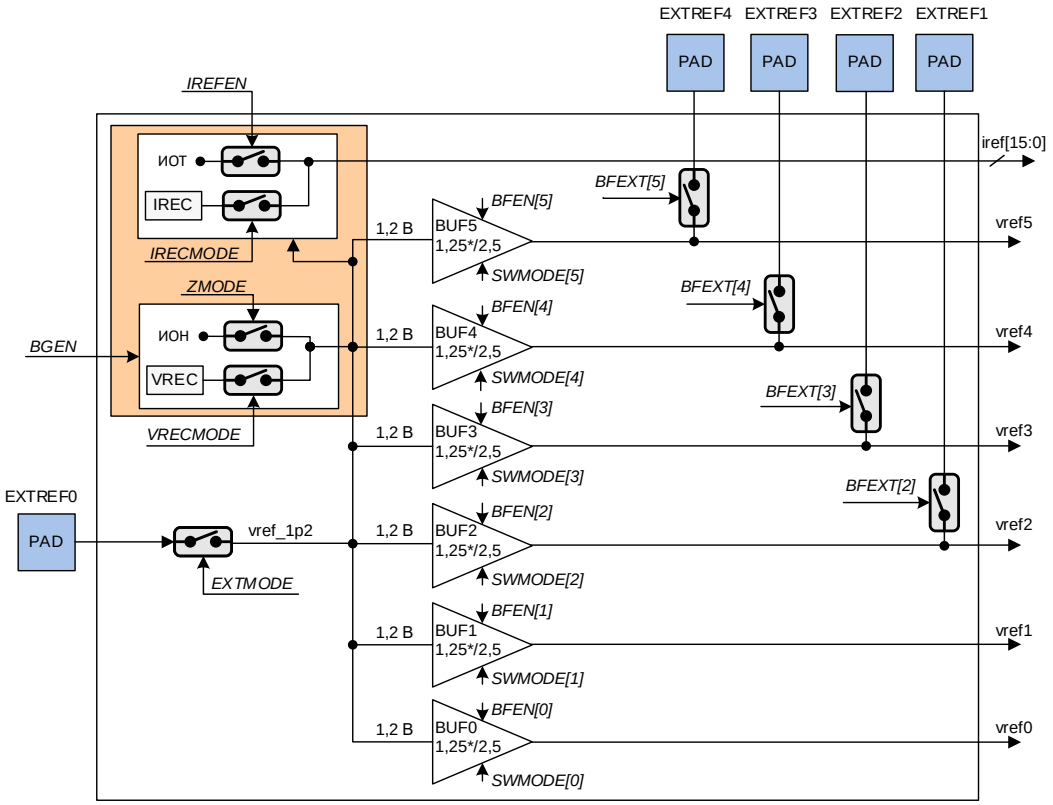



Рисунок 113 – Структурная схема компаратора

## 14.4 Контроллер формирования опорных напряжений и токов (ADC_BG_CNTR, ANA_BG_CNTR)

Для формирования опорных напряжений и токов для аналоговых блоков в микросхеме реализовано два блока формирования опорных напряжений и токов. Каждый из блоков способен формировать до шести опорных напряжений и до 16 опорных токов. Структурная схема блока приведена на рисунке 114.




* Используется для тестовых целей

Рисунок 114 – Структурная схема блока формирования опорных напряжений и токов

Блок ADC_BG используется для формирования опорных напряжений и токов для блоков АЦП, блок ANA_BG позволяет формировать опорные напряжения и токи для блоков умножителей частоты, ЦАП, компараторов и блоков ШИМ высокого разрешения HRPWM.

Для каждого из блоков формирования опорных напряжений реализован отдельный регистр управления ADC_BG_CNTR и ANA_BG_CNTR. Бит BGEN обеспечивает включение ядра блока формирования опорных напряжений.

После активации ядра источника опорных напряжений установка бита BG_IREFEN разрешает выдачу всех 16 токов номиналом по 10 мкА. Опорные токи источника ADC_BG используются блоками АЦП. Опорные токи источника ANA_BG используются блоками умножителей частоты PLLx, компараторов, ЦАП и ШИМ высокого разрешения HRPWM. Таким образом, перед использованием этих блоков необходимо активировать источник тока соответствующего блока формирования опорных напряжений.

Биты BG_BFEN[5:0] позволяют активировать шесть внутренних буферов опорных напряжений блока. Возможно формирование четырех опорных напряжений с внешних выводов микросхемы с функцией BG_EXTREF[4:1] и опорного напряжения BandGap vref_1p2 с внешних выводов с функцией BG_EXTREF0.

Выходные напряжения блока ADC_BG используются следующим образом:

- опорное напряжение VREF5 не используется, активируется битом BFEN5;
- опорное напряжение VREF4 не используется, активируется битом BFEN4;
- опорное напряжение VREF3 не используется, активируется битом BFEN3;
- опорное напряжение VREF2 использует ADC2, активируется битом BFEN2;

- опорное напряжение VREF1 использует ADC1, активируется битом BFEN1;
- опорное напряжение VREF0 использует ADC0, активируется битом BFEN0.

Выходные напряжения блока ANA_BG используются следующим образом:

- опорное напряжение VREF5 использует DAC3, активируется битом BFEN5;
- опорное напряжение VREF4 использует DAC2, активируется битом BFEN4;
- опорное напряжение VREF3 использует DAC1, активируется битом BFEN3;
- опорное напряжение VREF2 использует DAC0, активируется битом BFEN2;
- опорное напряжение VREF1 не используется, активируется битом BFEN1;
- опорное напряжение VREF0 не используется, активируется битом BFEN0.

Блоки компаратора, умножителей частоты и ШИМ высокого разрешения опорных напряжений не требуют.

Основные варианты работы блоков формирования опорных напряжений и токов в зависимости от конфигурации согласно таблице 59.

Таблица 59 – Режимы работы блоков формирования опорных напряжений и токов

| Сигналы\Режимы | Основной режим | Откл. сост. | Режим внешнего vref_1p2 | Режим внешних vref | Режим резервного напряжения | Режим резервного тока | Тест опоры | Тест vref |
|---|---|---|---|---|---|---|---|---|
| BG_BGEN | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| BG_IREFEN | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| BG_BFEN[5:0] | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| BG_SWMODE[5:0] | 0/1 | 0 | 0/1 | 0 | 0/1 | 0/1 | 0 | 0/1 |
| BG_ZMODE | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| BG_VRECMODE | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| BG_IRECMODE | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| BG_EXTMODE | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| BG_BFEXT[5:2] | 0 | 0 | 0 | 0/1 | 0 | 0 | 0 | 1 |
| Выходное напряжение блока, vref# | 1,25 В*/ 2,5 В | ↓0 | Экв. $V_{ext}$ | = $V_{ext}$ | ~1,25 В*/ ~2,5 В | 1,25 В*/ 2,5 В | ↓0 | 1,25 В*/ 2,5 В |
| Выходные токи блока | 10 мкА | | 10 мкА | 10 мкА | 10 мкА | ~10 мкА | | ↓0 |
| * Используется для тестовых целей | | | | | | | | |

1. Основной режим. В данном режиме функционируют все подблоки, ключи входных связей разорваны. Системы резервирования неактивны. Формируются номинальные выходные напряжения и токи. В данном режиме для каждого из шести выходных напряжений (vref#) возможно выбрать уровень 1,25 (используется для тестовых целей) или 2,5 В.
2. Отключенное состояние. Все подблоки отключены, выводы опорных напряжений находятся в Z-состоянии, потребление блока минимизировано до токов утечки.

3. Режим внешнего опорного напряжения vref_1p2. Данный режим предполагает формирование напряжения BandGap (vref_1p2) извне (вывод микросхемы с функцией BG_EXTREF0). Оно является опорным для выходных буферов и источника опорных токов. В этом режиме BandGap, необходимый для формирования опорных токов активен, но его выход переведен в Z-состояние. Ключ, формирующий напряжение vref_1p2, открыт. Для каждого из шести выходных напряжений (vref#) возможно выбрать уровень 1,25 (используется для тестовых целей) или 2,5 В.

4. Режим внешних vref. Данный режим предполагает использование внешних опорных напряжений для выводов vref[5:2]. В этом режиме BandGap, необходимый для формирования опорных токов, активен. Внешние опорные напряжения должны быть

поданы на нужные выводы микросхемы, соответствующие vref[5:2] (в соответствии с таблицей портов).

**Важно!** При подключении внешнего опорного напряжения соответствующий буфер должен быть отключен, чтобы его выход был в Z-состоянии.

5. Режим резервного напряжения опоры. Данный режим включает резервное формирования опорного напряжения для встроенного ИОН, формирующий опорные токи и напряжения для выходных буферов и источника опорных токов. Также формируются температурно-зависимые опорные токи для выходных буферов и источника опорного тока. Режим может быть применен в случае неспособности внутреннего ИОН формировать опорное напряжение (например, в случае не запуска). Опорное напряжение формируется на резистивном делителе между напряжением питания и землей с малой точностью, температурной зависимостью, а также зависимостью от напряжения питания. Таким же образом формируются и температурно-зависимые токи. В данном режиме должен быть подан сигнал отключения ИОН и перевод выхода в Z-состояние.

6. Режим резервных токов. Данный режим включает резервное формирования опорных токов для ИОТ. Режим может быть применен в случае неспособности внутреннего ИОТ формировать опорные токи. Опорные токи формируются на резистивном делителе между напряжением питания и землей с малой точностью, температурной зависимостью, а также зависимостью от напряжения питания. В данном режиме должен быть подан сигнал отключения ИОТ. Остальные блоки могут работать в обычном режиме.

7. Режимы тестирования внутреннего опорного напряжения vref_1p2 и выходных опорных напряжений vref#. Предполагает замер уровня опорного напряжения в узле vref_1p2 и/или выводах vref#. Для этого измеряемый блок должен работать в обычном режиме, на ключ соответствующего внешнего подключения подан сигнал. Необходимо учитывать, что между выводом микросхемы с функцией EXTREF0 (внешнее vref_1p2) и внутренним узлом vref_1p2 включен резистор ~82 КОм.

# 15 Блоки контроллеров интерфейсов передачи данных

## 15.1 Контроллер SSP (SSP_CNTR)

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 32-разрядных ячеек данных) из параллельного в последовательный формат;
- преобразование данных из последовательного в параллельный формат и их запись в аналогичный буфер FIFO приемника (восемь 32-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;
- наличие данных в FIFO приемника;
- отсутствие данных в FIFO передатчика;
- отсутствие данных в сдвиговом регистре передатчика.

Основные сведения о модуле представлены в следующих разделах:

- характеристики интерфейса SPI;
- характеристики интерфейса Microwire;
- характеристики интерфейса SSI.

Основные характеристики модуля SSP:

- функционирование как в ведущем, так и в ведомом режиме;
- программное управление скоростью обмена;
- состоит из независимых буферов приема и передачи (восемь ячеек по 32 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 32 бит;
- независимое маскирование прерываний от буфера FIFO передатчика, буфера FIFO приемника, по переполнению буфера приемника, по наличию данных в FIFO приемника, по отсутствию данных в FIFO передатчика, а также по отсутствию данных в сдвиговом регистре передатчика;
- доступна возможность тестирования по шлейфу, соединяющему вход с выходом;
- поддержка прямого доступа к памяти (DMA).

### 15.1.1 *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 32 бит;
- сброс FIFO передатчика;
- маскирование прерываний.

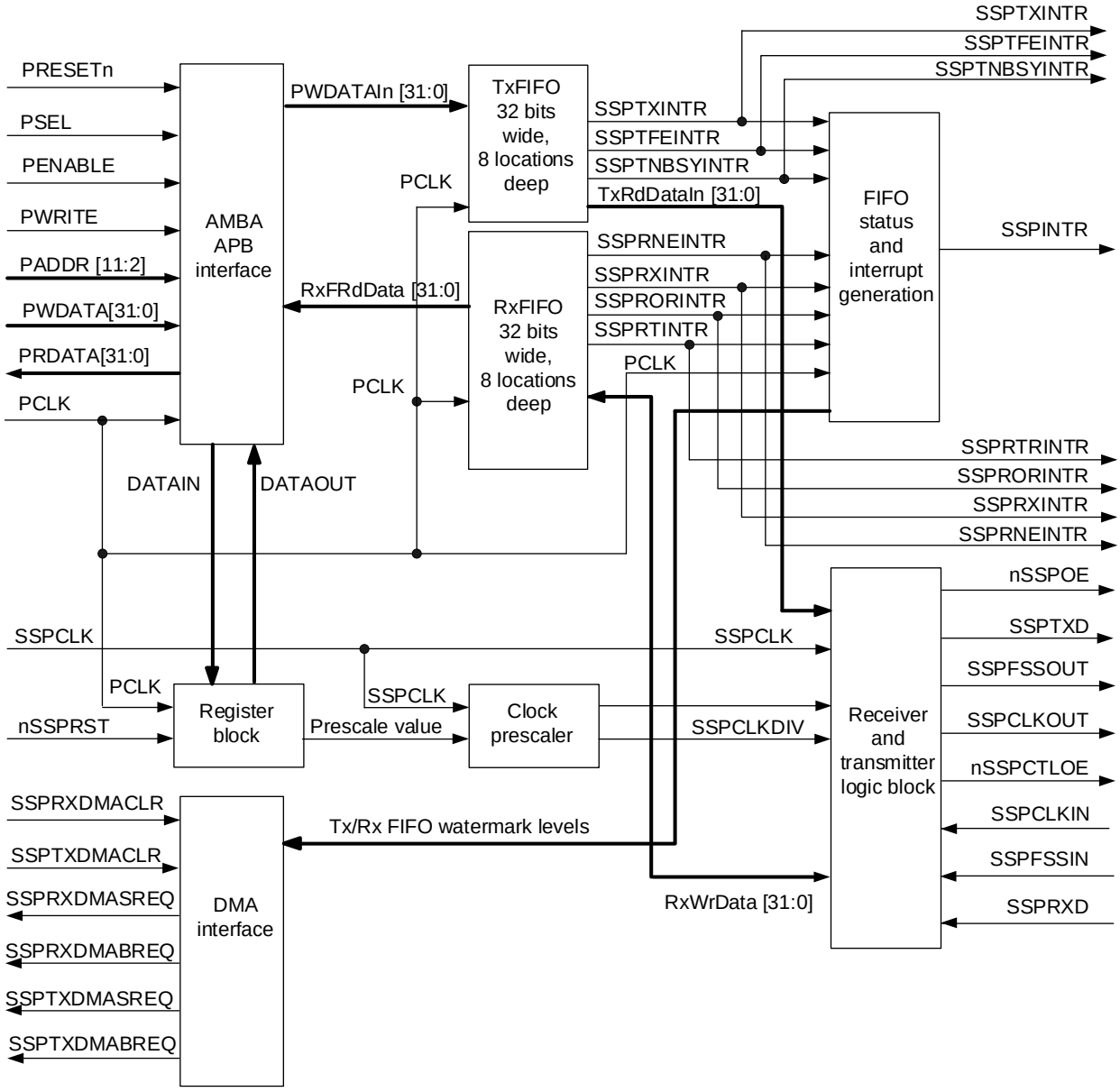



Рисунок 115 – Структурная схема модуля SSP

#### 15.1.1.1 Характеристики интерфейса SPI

Последовательный синхронный интерфейс SPI фирмы Motorola обеспечивает:

- полнодуплексный обмен данными по четырехпроводной линии;
- программное задание фазы и полярности тактового сигнала.

#### 15.1.1.2 Характеристики интерфейса Microwire

Интерфейс Microwire фирмы National Semiconductor обеспечивает полудуплексный обмен данными с использованием восьмибитных управляющих последовательностей.

#### 15.1.1.3 Характеристики интерфейса SSI

Интерфейс SSI фирмы Texas Instruments обеспечивает:

- полнодуплексный обмен данными по четырехпроводной линии;
- возможность перевода линии передачи данных в третье (высокоимпедансное) состояние.

### 15.1.2 *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 32 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 следующие независимо маскируемые 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;
- SSPRNEINTR – наличие данных в FIFO приемника;
- SSPTFEINTR – отсутствие данных в FIFO передатчика;
- SSPTNBSYINTR – отсутствие данных в сдвиговом регистре передатчика.

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).

#### 15.1.2.1 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.

#### 15.1.2.2 Буфер FIFO передатчика

Буфер передатчика имеет ширину 32 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. Присутствует возможность программного сброса с помощью сигнала RESTxFIFO из управляющего регистра CR1.

#### 15.1.2.3 Буфер FIFO приемника

Буфер приемника имеет ширину 32 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.

#### 15.1.2.4 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. Затем 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.

#### 15.1.2.5 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.

#### 15.1.2.6 Интерфейс прямого доступа к памяти

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

#### 15.1.2.7 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.

#### 15.1.2.8 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 32-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.

#### 15.1.2.9 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

$$FSSPCLK \leq FPCLK.$$

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:

$$\begin{cases} FSSPCLK_{min} \geq 2 \times FSSPCLKOUT_{max} \ (master\ mode) \\ FSSPCLK_{min} \geq 12 \times FSSPCLKIN_{max} \ (slave\ mode) \end{cases}$$

При активации сигнала SSPFRX в регистре управления CR0, это соотношение приобретает вид:

$$\begin{cases} FSSPCLK_{min} \geq 2 \times FSSPCLKOUT_{max} \ (master\ mode) \\ FSSPCLK_{min} \geq 4 \times FSSPCLKIN_{max} \ (slave\ mode) \end{cases}$$

Аналогично, максимальная допустимая частота сигнала SSPCLK определяется следующей системой соотношений, которые должны выполняться одновременно:

$$\begin{cases} FSSPCLK_{max} \leq 254 \times 256 \times FSSPCLKOUT_{min} \ (master\ mode) \\ FSSPCLK_{max} \leq 254 \times 256 \times FSSPCLKIN_{min} \ (slave\ mode). \end{cases}$$

#### 15.1.2.10 Программирование регистра управления CR0

Регистр CR0 предназначен для:
- установки скорости информационного обмена;
- выбора одного из трех протоколов обмена данными;
- выбора размера слова данных;
- активации быстрого режима работы ведомого устройства.

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.

Сигнал SSPFRX используется для активации у ведомого модуля SPI быстрого режима работы и включения синхронизации сигнала RXD.

Для протокола SPI фирмы Motorola также задаются полярность и фаза сигнала (биты SPH и SPO).

#### 15.1.2.11 Программирование регистра управления CR1

Регистр SSPCR1 предназначен для:
- выбора ведущего или ведомого режима функционирования приемо-передатчика;
- включения режима проверки канала по шлейфу;
- разрешения или запрещения работы модуля;
- программного сброса FIFO передатчика.

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).

Для программного сброса FIFO передатчика необходимо установить в 1 бит RESTxFIFO. После сброса сигнал RESTxFIFO автоматически сбросится в 0.

#### 15.1.2.12 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

$$FSSPCLKOUT = \frac{FSSPCLK}{CPSDVR \cdot (1+SCR)}.$$

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.

#### 15.1.2.13 Формат информационного кадра

Каждый информационный кадр содержит в зависимости от запрограммированного значения от 4 до 32 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 восьмибитная управляющая последовательность. В течение всего времени передачи этой последовательности приемник не обрабатывает каких-либо входных данных. После того как сигнал передан и декодирован ведомым устройством, оно выдерживает паузу в один тактовый интервал после передачи последнего бита управляющей последовательности, после чего передает в адрес ведущего устройства запрошенные данные. Длительность блока данных от ведомого устройства может составлять от 4 до 32 бит, таким образом общая длительность информационного кадра составляет от 13 до 41 бит.

### 15.1.2.14 Формат синхронного обмена SSI фирмы TexasInstruments

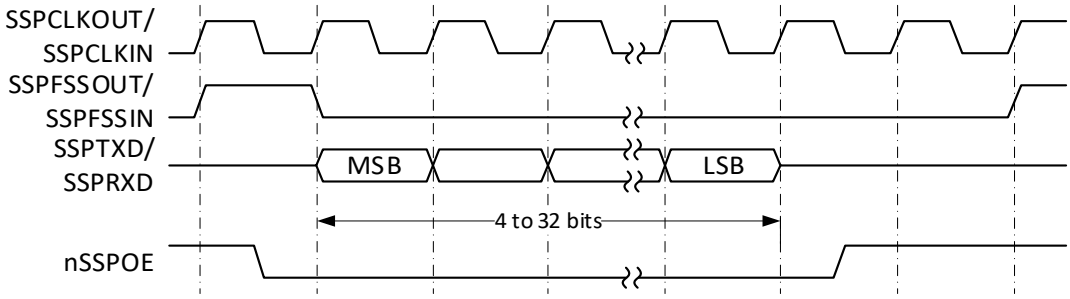



Рисунок 116 – Формат синхронного обмена протокола SSI (единичный обмен)

В данном режиме, при неактивном приемопередатчике SSP сигналы SSP_CLK и SSP_FSS переводятся в низкий логический уровень, а линия передачи данных SSP_TXD поддерживается в третьем состоянии.

После появления хотя бы одного элемента в буфере FIFO передатчика сигнал SSP_FSS переводится в высокий логический уровень на время, соответствующее одному периоду сигнала SSP_CLK. Значение из буфера FIFO при этом переносится в сдвиговый регистр блока передатчика. По следующему переднему фронту сигнала SSP_CLK, старший значащий разряд информационного кадра (4 – 32 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.

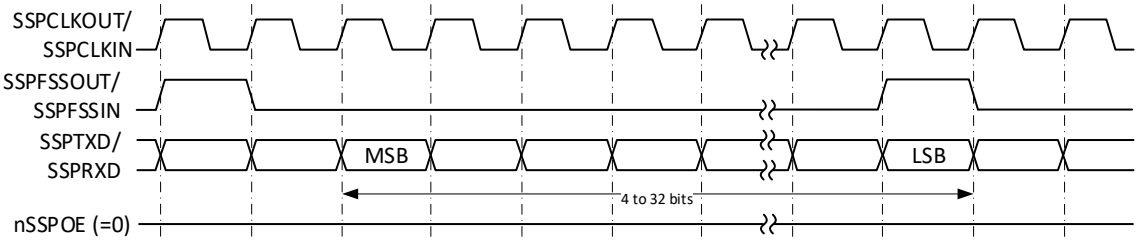



Рисунок 117 – Формат синхронного обмена протокола SSI (непрерывный обмен)

### 15.1.2.15 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.

#### 15.1.2.16 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 118, 119.

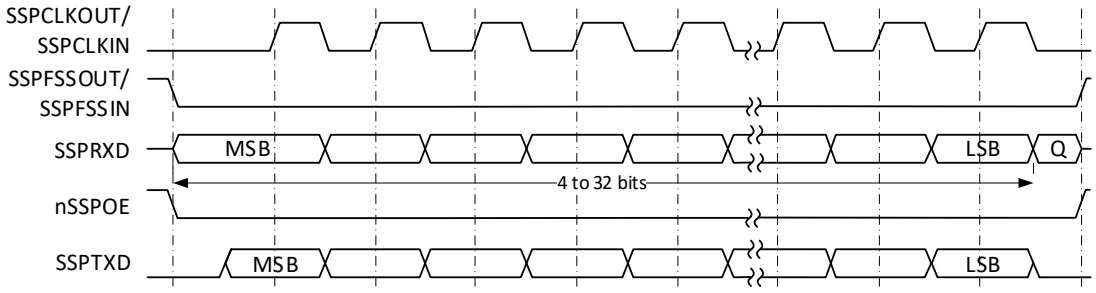



Рисунок 118 – Формат синхронного обмена протокола SPI, SPO=0, SPH=0 (одиночный обмен)

Примечание – На рисунке 118 буквой Q обозначен сигнал с неопределенным уровнем.

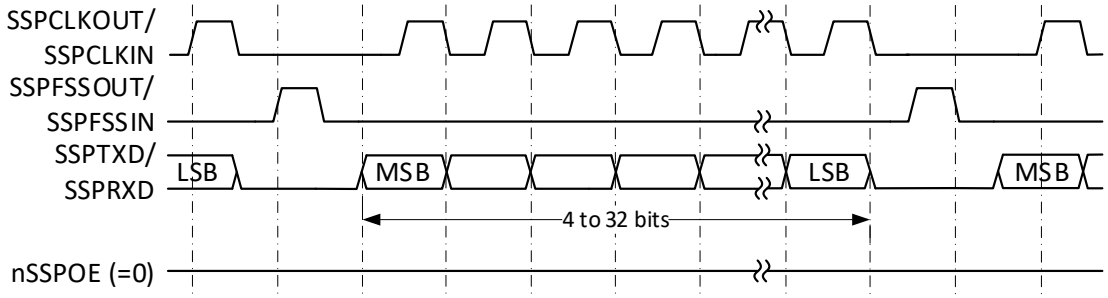



Рисунок 119 – Формат синхронного обмена протокола SPI, SPO=0, SPH=0 (непрерывный обмен)

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.

#### 15.1.2.17 Формат синхронного обмена SPI фирмы Motorola, SPO=0, SPH=1

Временные диаграммы последовательного синхронного обмена в режиме SPI с SPO=0, SPH=1 приведены на рисунке 120.

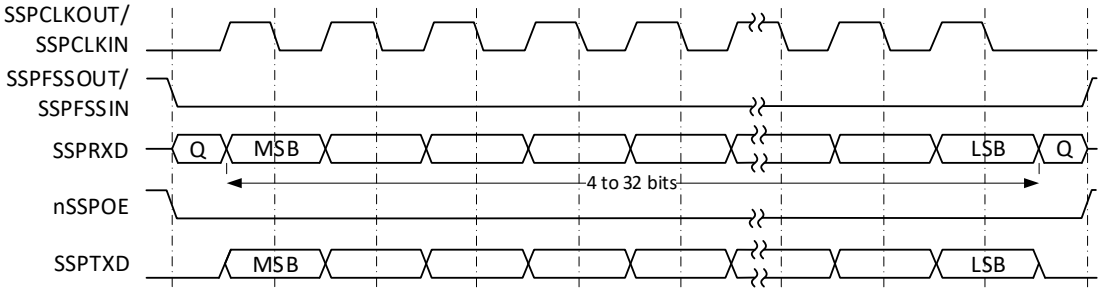



Рисунок 120 – Формат синхронного обмена протокола SPI, SPO=0, SPH=1

Примечание – На рисунке 120 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.

#### 15.1.2.18 Формат синхронного обмена SPI фирмы Motorola, SPO=1, SPH=0

Временные диаграммы последовательного синхронного обмена в режиме SPI с SPO = 1, SPH = 0 показаны ниже (Рисунок 121, Рисунок 122).

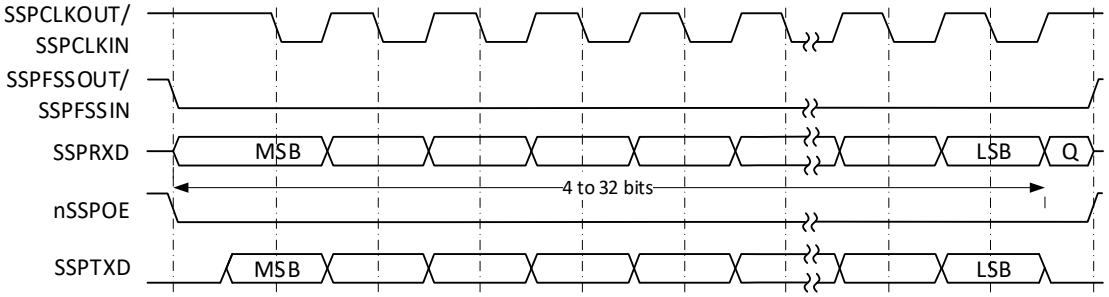


Рисунок 121 – Формат синхронного обмена протокола SPI, SPO=1, SPH=0 (одиночный обмен)

Примечание – На рисунке 121 буквой Q обозначен сигнал с неопределенным уровнем.

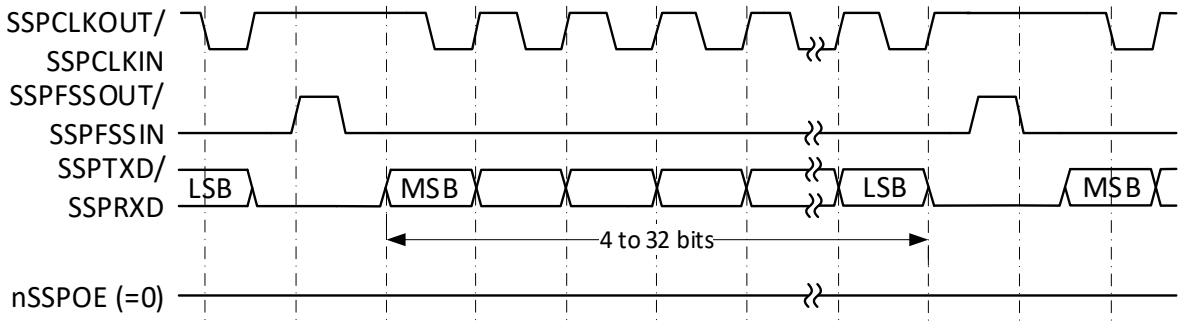



Рисунок 122 – Формат синхронного обмена протокола SPI, SPO=1, SPH=0 (непрерывный обмен)

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.

#### 15.1.2.19 Формат синхронного обмена SPI фирмы Motorola, SPO=1, SPH=1

Временные диаграммы последовательного синхронного обмена в режиме SPI с SPO = 1, SPH = 1 приведены на рисунке 123.

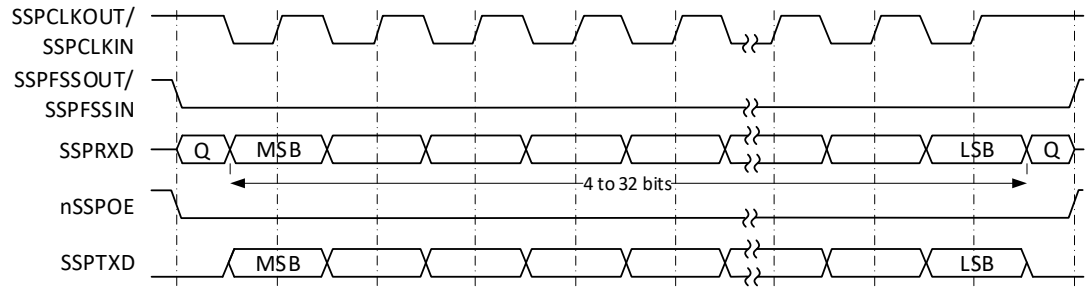



Рисунок 123 – Формат синхронного обмена протокола SPI, SPO = 1, SPH = 1

Примечание – На рисунке 123 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.

#### 15.1.2.20 Формат синхронного обмена Microwire фирмы NationalSemiconductor

Временные диаграммы последовательного синхронного обмена в режиме Microwire показаны на рисунках ниже (Рисунок 124, Рисунок 125).

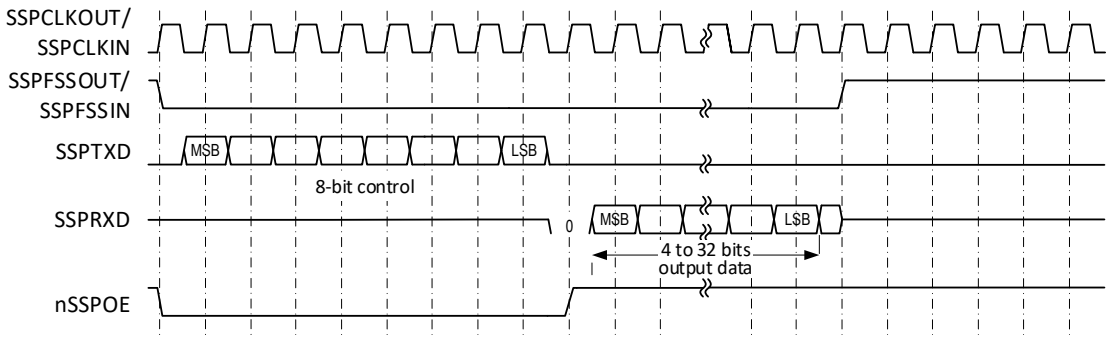



Рисунок 124 – Формат синхронного обмена протокола Microwire (одиночный обмен)

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-ми 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 32 бит, таким образом, общая длительность информационного кадра составляет от 13 до 41 бит.

В данном режиме во время ожидания приемопередатчика:

- сигнал SSP_CLK имеет низкий логический уровень;
- сигнал SSP_FSS имеет высокий логический уровень;
- сигнал SSP_TXD переводится в высокоимпедансное состояние.

Переход в режим информационного обмена происходит после записи управляющего байта в буфер FIFO передатчика. По заднему фронту сигнала SSP_FSS данные из буфера переносятся в регистр сдвига блока передатчика, откуда, начиная со старшего значащего разряда, последовательно выдаются в линию SSP_TXD. Линия SSP_FSS остается в низком логическом уровне в течение всей передачи кадра. Линия SSP_RXD при этом находится в высокоимпедансном состоянии.

Внешнее ведомое устройство осуществляет прием бит данных по переднему фронту сигнала SSP_CLK. По окончании приема последнего бита управляющей последовательности она декодируется в течение одного тактового интервала, после чего ведомое устройство передает запрошенные данные в адрес модуля SSP. Биты данных выдаются в линию SSP_RXD по заднему фронту сигнала SSP_CLK. Ведущее устройство, в свою очередь, регистрирует их по переднему фронту этого тактового сигнала. В случае одиночного информационного обмена по окончании приема последнего бита слова данных сигнал SSP_FSS переводится в высокий уровень на время, соответствующее одному тактовому интервалу, что служит командой для переноса принятого слова данных из регистра сдвига в буфер FIFO приемника.

Примечание – Внешнее устройство может перевести линию приемника в третье состояние по заднему фронту сигнала SSP_CLK после приема последнего бита слова данных, либо после перевода линии SSP_FSS в высокий логический уровень.

Непрерывный обмен данными начинается и заканчивается также, как и одиночный обмен. Однако линия SSP_FSS удерживается в низком логическом уровне в течение всего сеанса передачи данных. Управляющий байт следующего информационного кадра передается сразу же после приема младшего значащего разряда текущего кадра. Данные из сдвигового регистра передаются в буфер приемника, после регистрации младшего разряда очередного слова по заднему фронту сигнала SSP_CLK.

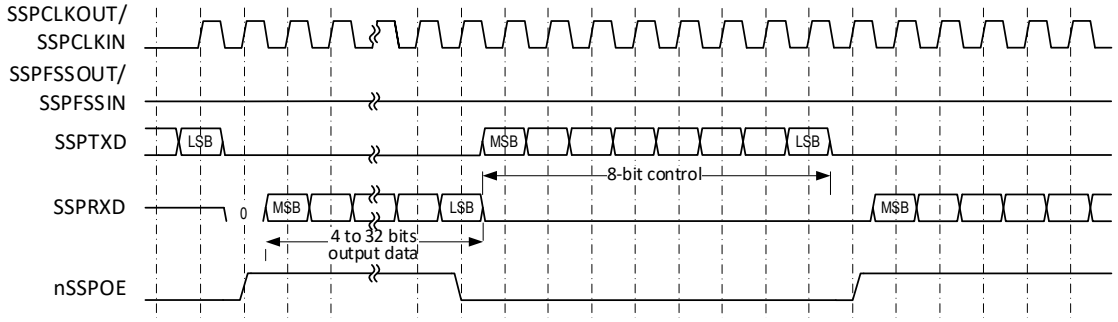



Рисунок 125 – Формат синхронного обмена протокола Microwire (непрерывный обмен)

***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.

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.

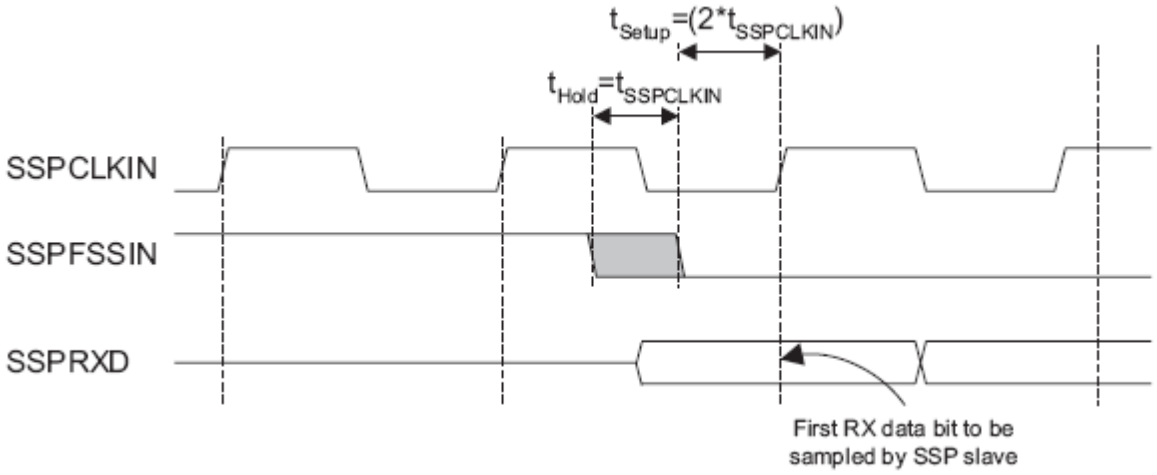



Рисунок 126 – Формат Microwire, требования к времени установки и удержания сигнала

#### 15.1.2.21 Примеры конфигурации модуля в ведущем и ведомом режимах

На рисунках ниже (рисунки 127 – 129) 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 приёмопередатчик должен быть изначально сконфигурирован в одном из этих режимов.

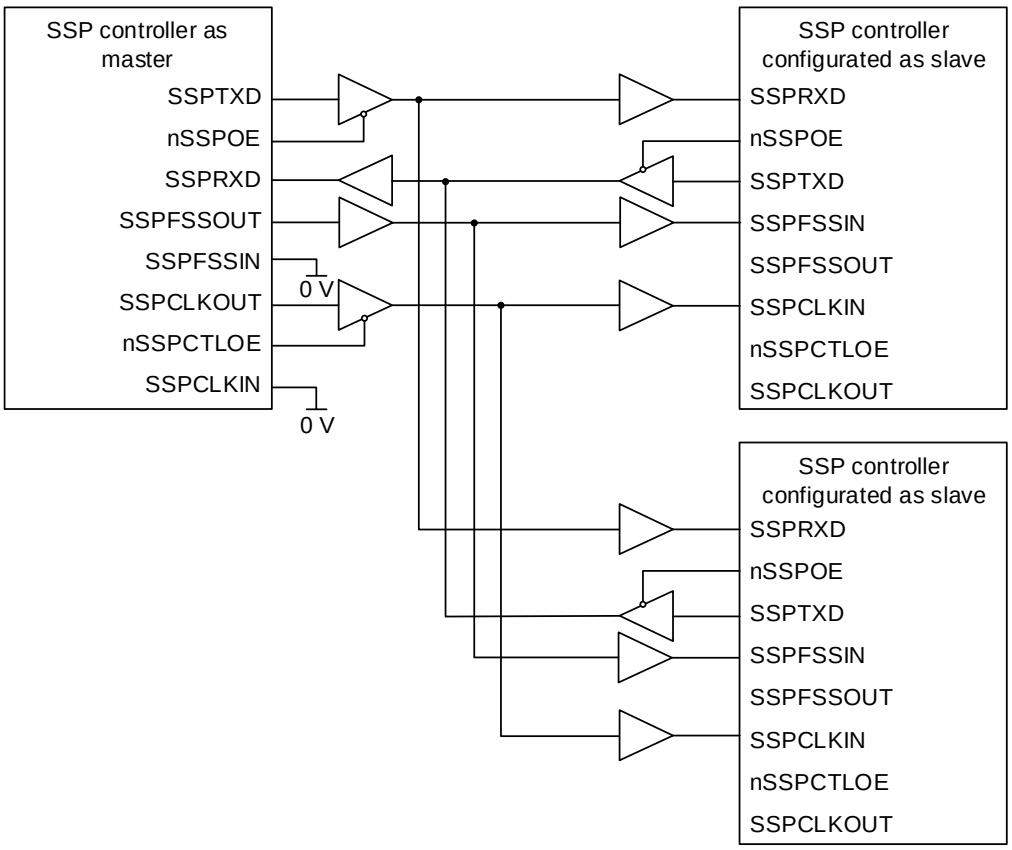



Рисунок 127 – Ведущее устройство SSP подключено к двум ведомым

Рисунок 127 показывает совместную работу трех модулей SSP, один из которых сконфигурирован в качестве ведущего, а два – в качестве ведомых устройств. Ведущее устройство способно передавать данные по кругу 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.

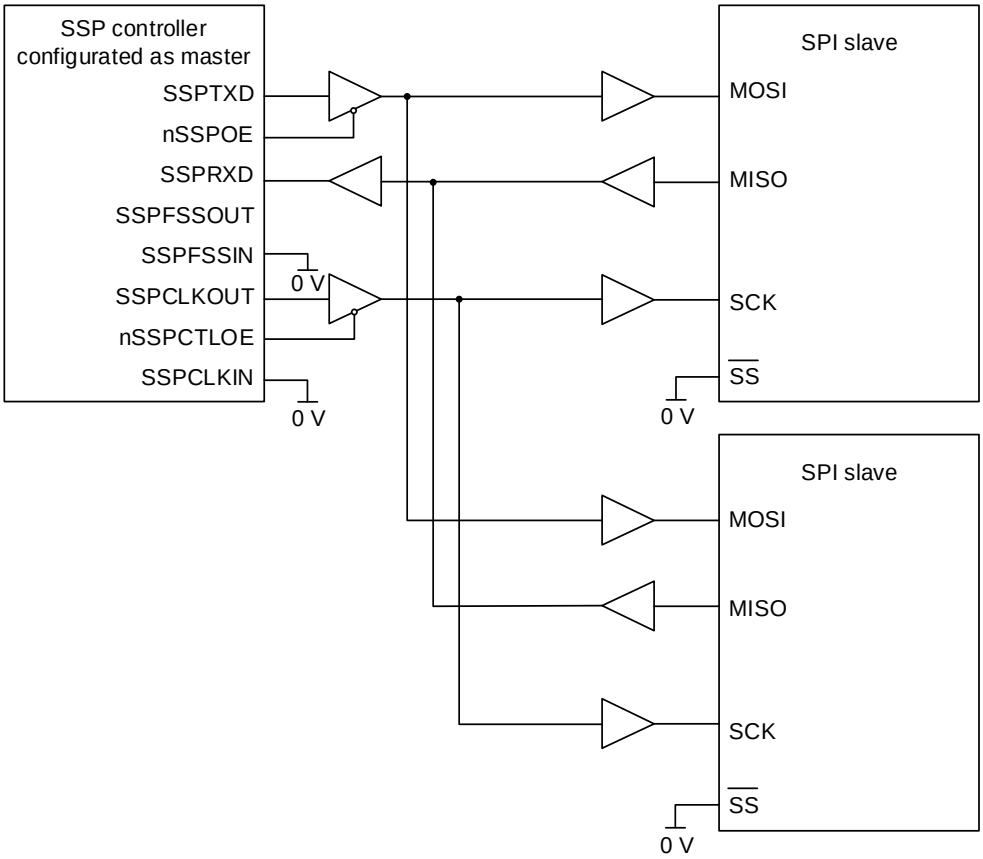



Рисунок 128 – Ведущее устройство SSP подключено к двум ведомым, поддерживающим SPI

Рисунок 128 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 по кругу по линии SSP_TXD. Ответная передача данных на входную линию SSP_RXD ведущего устройства, одновременно осуществляется только одним из ведомых по соответствующей линии MISO.

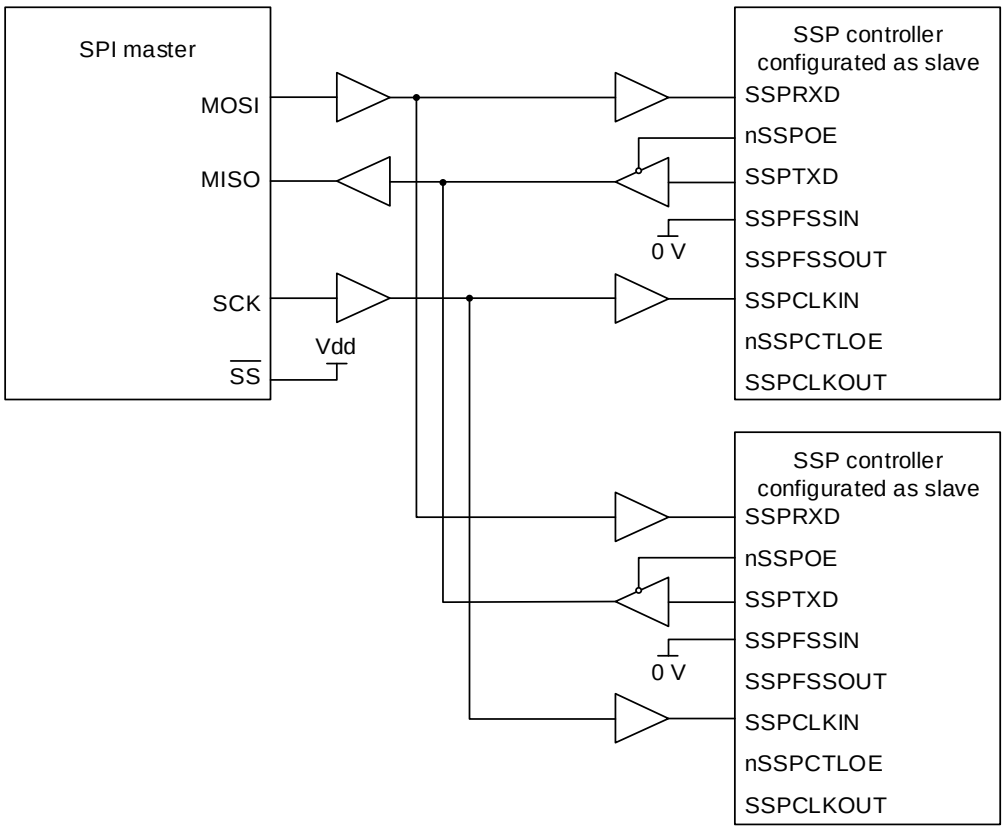



Рисунок 129 – Ведущее устройство, протокол SPI, подключено к двум ведомым модулям SSP

Рисунок 129 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 по кругу 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.

### 15.1.3 *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 60 приведены значения порогов заполнения буферов приемника и передатчика, необходимых для срабатывания запросов блочного обмена DMABREQ.

Таблица 60 – 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 130 показывает временные диаграммы одноэлементного и блочного запросов DMA, в том числе действие сигнала DMACLR. Все сигналы должны быть синхронизированы с PCLK.

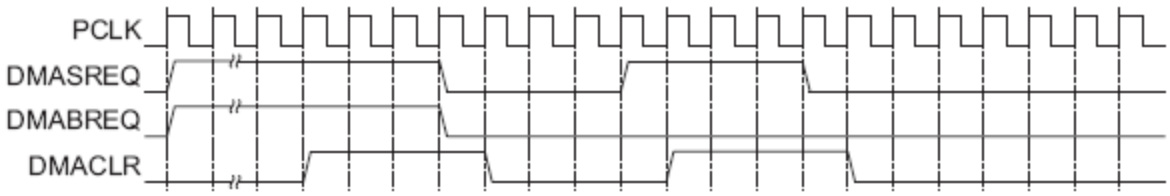



Рисунок 130 – Временные диаграммы обмена в режиме DMA

### 15.1.4 *Прерывания*

В модуле предусмотрено семь 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;
- SSPRNEINTR – наличие данных в FIFO приемника;
- SSPTFEINTR – отсутствие данных в FIFO передатчика;
- SSPTNBSYINTR – отсутствие данных в сдвиговом регистре передатчика;
- SSPINTR – логическое ИЛИ сигналов SSPRXINTR, SSPTXINTR, SSPRTINTR, SSPRORINTR, SSPRNEINTR, SSPTFEINTR и SSPTNBSYINTR.

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.

Признаки возникновения каждого из условий прерывания можно считать либо из регистра прерываний SSPRIS, либо из маскированного регистра прерываний SSPMIS.

#### 15.1.4.1 SSPRXINTR

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

#### 15.1.4.2 SSPTXINTR

Прерывание по заполнению буфера FIFO передатчика. Формируется в случае, если буфер передатчика содержит четыре или менее корректных слов данных.

Состояние прерывания не зависит от значения сигнала разрешения работы модуля SSP. Это позволяет организовать взаимодействие программного обеспечения с передатчиком одним из двух способов. Во-первых, можно записать данные в буфер заблаговременно, перед активизацией передатчика и разрешения прерываний. Во-вторых, можно предварительно разрешить работу модуля и формирование прерываний и заполнять буфер передатчика в ходе работы процедуры обслуживания прерываний.

#### 15.1.4.3 SSPRORINTR

Прерывание по переполнению буфера FIFO приемника формируется в случае, если буфер уже заполнен и блоком приемника осуществлена попытка записать в него еще одно слово. При этом принятое слово данных регистрируется в регистре сдвига приемника, но в буфер приемника не заносится.

#### 15.1.4.4 SSPRTINTR

Прерывание по таймауту приемника возникает в случае, если буфер FIFO приемника не пуст, и на вход приемника не поступало новых данных в течение времени таймаута, равного 32 тактам частоты SSPCLKOUT (для ведущего и ведомого режимов работы). Данный механизм гарантирует, что пользователь будет знать о наличии в буфере приемника необработанных данных.

Прерывание по таймауту снимается либо после считывания данных из буфера приемника до его опустошения, либо после приема новых слов данных по входной линии SSP_RXD. Кроме того, оно может быть снято путем записи 1 в бит RTIC регистра сброса прерывания SSPTICR.

#### 15.1.4.5 SSPRNEINTR

Прерывание по наличию данных в FIFO приемника. Сигнал формируется при записи хотя бы одного слова в FIFO приемника и сохраняется до тех пор, пока все данные не будут считаны.

#### 15.1.4.6 SSPTFEINTR

Прерывание по отсутствию данных в FIFO передатчика. Находится в активном уровне до тех пор, пока в FIFO не будет записано хотя бы одно слово. После передачи всех данных возвращается в активный уровень.

#### 15.1.4.7 SSPTNBSYINTR

Прерывание по отсутствию данных в сдвиговом регистре передатчика. Прерывание активно, когда модуль SSP не передаёт данные.

#### 15.1.4.8 SSPINTR

Все описанные сигналы запроса на прерывание скомбинированы в общую линию путем объединения по схеме ИЛИ сигналов SSPRXINTR, SSPTXINTR, SSPRNEINTR, SSPTFEINTR, SSPTNBSYINTR, SSPRTINTR и SSPRORINTR с учетом маскирования. Общий выход может быть подключен к системному контроллеру прерываний, что позволит ввести дополнительное маскирование запросов на уровне периферийных устройств.

## 15.2 Контроллер UART (UART_CNTR)

Модуль универсального асинхронного приемопередатчика (UART – Universal Synchronous Asynchronous Receiver Transmitter) является периферийным устройством 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).

### 15.2.1 *Основные сведения*

Основные сведения о модуле представлены в следующих разделах:

- основные характеристики;
- программируемые параметры;
- отличия от приемопередатчика 16C650.

### 15.2.2 *Основные характеристики модуля UART*

Модуль может быть запрограммирован для использования, как в качестве универсального асинхронного приемопередатчика, так и для инфракрасного обмена данными (SIR).

Содержит независимые буферы приема (16x13) и передачи (16x9) 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 также 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, 8 или 9 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.

### 15.2.3 *Программируемые параметры*

Следующие ключевые параметры могут быть заданы программно:
- скорость передачи данных – целая и дробная часть числа;
- количество бит данных;
- количество стоповых бит;
- режим контроля четности;
- разрешение или запрет использования буферов FIFO (глубина очереди данных – 32 элемента или один элемент, соответственно);
- порог срабатывания прерывания по заполнению буферов FIFO (1/8, 1/4, 1/2, 3/4 и 7/8);
- частота внутреннего тактового генератора (номинальное значение – 1,8432 МГц) может быть задана в диапазоне 1,42 – 2,12 МГц для обеспечения возможности формирования бит данных с укороченной длительностью в режиме пониженного энергопотребления;
- режим аппаратного управления потоком данных.

#### 15.2.3.1 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.

#### 15.2.3.2 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 работы 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.

### 15.2.4 *Описание функционирования блока UART*

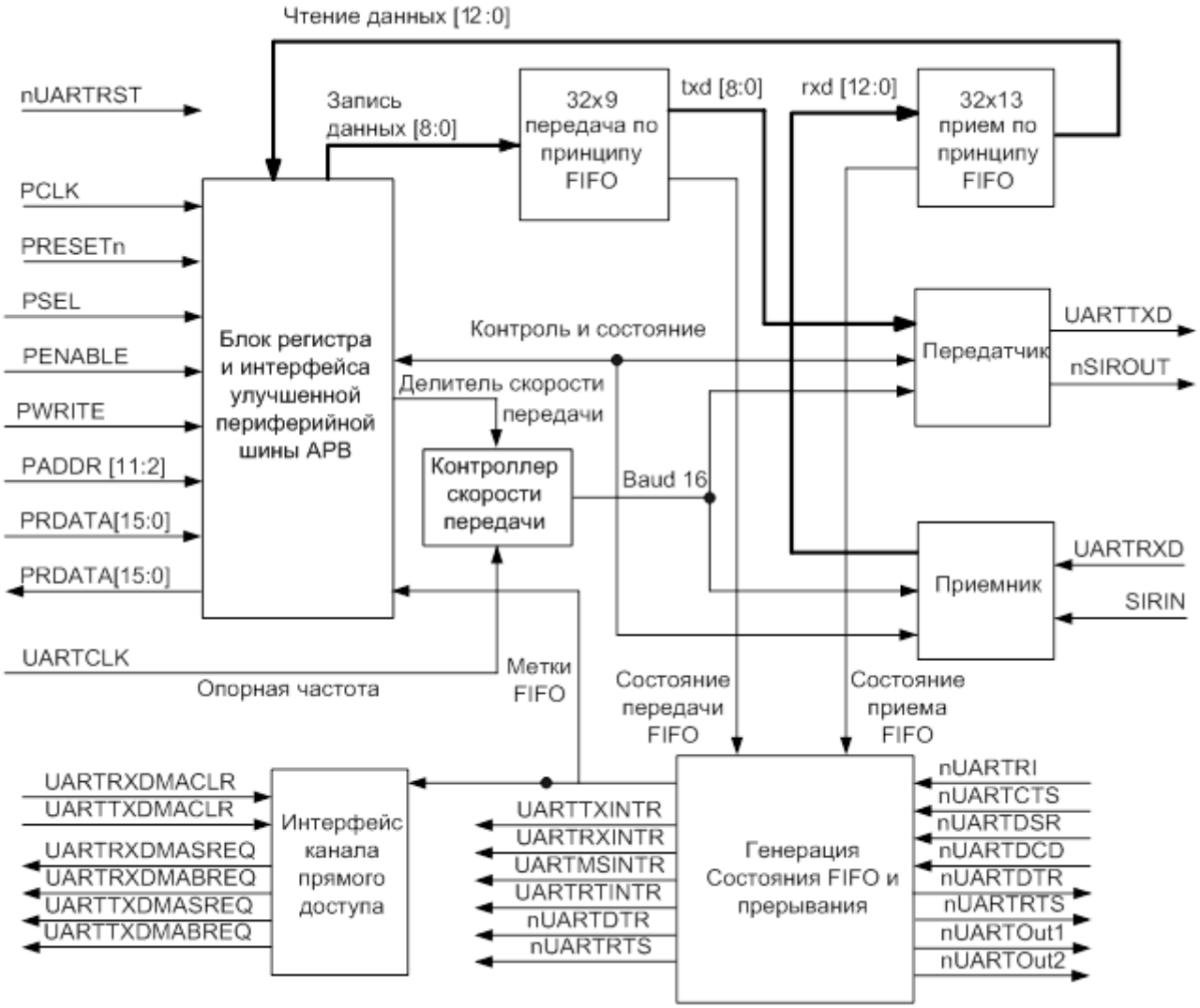



Рисунок 131 – Блок-схема универсального асинхронного приёмопередатчика (UART)

#### 15.2.4.1 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.

#### 15.2.4.2 Буфер FIFO передатчика

Буфер передатчика имеет ширину 9 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.

#### 15.2.4.3 Буфер FIFO приемника

Буфер приемника имеет ширину 13 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.

#### 15.2.4.4 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.

#### 15.2.4.5 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.

#### 15.2.4.6 Блок формирования прерываний

Контроллер генерирует независимые маскируемые прерывания с активным высоким уровнем. Кроме того, формируется комбинированное прерывание путем объединения указанных независимых прерываний по схеме ИЛИ.

Комбинированный сигнал прерывания может быть подан на внешний контроллер прерываний системы, при этом появится дополнительная возможность маскирования устройства в целом, что облегчает построение модульных драйверов устройств.

Другой подход состоит в подаче на системный контроллер прерываний независимых линий запроса на прерывание от приемопередатчика. В этом случае процедура обработки сможет одновременно считать информацию обо всех источниках прерывания. Данный подход привлекателен в случае, если скорость доступа к регистрам периферийных устройств значительно превышает тактовую частоту центрального процессора в системе реального времени.

Для более подробной информации см. подраздел «Прерывания».

### 15.2.5 *Интерфейс прямого доступа к памяти*

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

### 15.2.6 *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.

### 15.2.7 *Описание функционирования ИК кодека IrDASIR*

Структурная схема кодека представлена на рисунке 132.

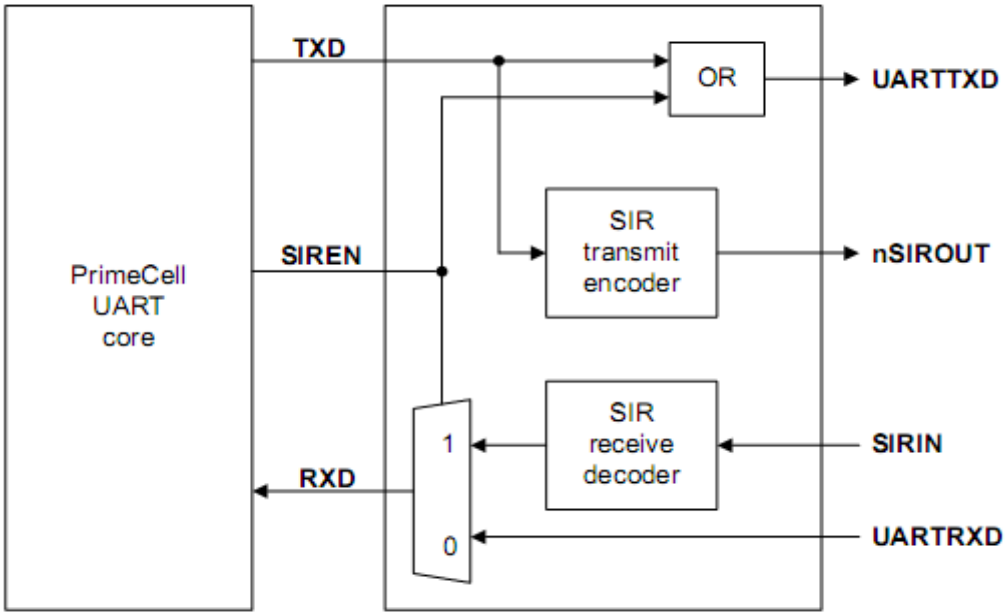



Рисунок 132 – Структурная схема кодека IrDA

#### 15.2.7.1 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.

#### 15.2.7.2 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.

### 15.2.8 *Описание работы UART*

#### 15.2.8.1 Сброс модуля

Приемопередатчик и кодек могут быть сброшены общим сигналом сброса процессора.

#### 15.2.8.2 Тактовые сигналы

Частота тактового сигнала UARTCLK должна обеспечивать поддержку требуемого диапазона скоростей передачи данных:

$$F_{UARTCLK}(min) \geq 16 \times baud_rate_{max};$$

$$F_{UARTCLK}(max) \leq 16 \times 65535 \times baud_rate_{min}.$$

Например, для поддержки скорости передачи данных в диапазоне от 110 до 460800 Бод частота UARTCLK должна находиться в интервале от 7,3728 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.

$$F_{UARTCLK} \leq {}^{5}/_{3} \times F_{CPU_CLK}.$$

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.

#### 15.2.8.3 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.

#### 15.2.8.4 Коэффициент деления частоты

Коэффициент деления для формирования скорости передачи данных состоит из 22-х бит, при этом 16 бит выделено для представления его целой части, а 6 бит – дробной части. Возможность задания нецелых коэффициентов деления позволяет осуществлять обмен данными со стандартными информационными скоростями, при этом используя в качестве UARTCLK тактовый сигнал с произвольной частотой более 3,6864 МГц.

Целая часть коэффициента деления записывается в 16-битный регистр UARTIBRD. Шестиразрядная дробная часть записывается в регистр UARTFBRD. Значение коэффициента деления связано с содержимым указанных регистров следующим образом

$$\text{Коэффициент деления} = \frac{UARTCLK}{(16 \cdot \text{скорость передачи данных})} = BRD_I + BRD_F,$$

где BRD_I – целая часть, а BRD_F – дробная часть коэффициента деления.




Рисунок 133 – 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:

$$M = \lfloor BRD_F \cdot 2^n + 0.5 \rfloor,$$

где ⌊ ⌋ – 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.

#### 15.2.8.5 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.

Данные для передачи 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. Сигнал BUSY переходит в низкое состояние 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 состояния 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), то 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 (Таблица 61).

#### 15.2.8.6 Биты ошибки

Три бита признаков ошибки, ассоциированные с принятым символом данных, заносятся в разряды [11…9] слова данных в буфере FIFO приемника. Также предусмотрен признак ошибки переполнения буфера FIFO в разряде 12 слова данных.

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

Таблица 61 – Назначение бит слова данных в FIFO-буфере приемника

| Бит буфера FIFO | Назначение |
|---|---|
| 12 | Признак переполнения буфера |
| 11 | Ошибка – «разрыв линии» |
| 10 | Ошибка проверки на четность |
| 9 | Ошибка формирования кадра |
| 8…0 | Принятые данные |

#### 15.2.8.7 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.

#### 15.2.8.8 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.

#### 15.2.8.9 Проверка по шлейфу

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

### 15.2.9 *Работа кодека ИК обмена данными IrDASIR*

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:

$$1{,}42\ \text{МГц} < F_{IrLPBaud16} < 2{,}12\ \text{МГц}.$$

#### 15.2.9.1 Проверка по шлейфу

Проверка по шлейфу выполняется после установки в 1 бита LBE регистра управления контроллером UARTCR с одновременной установкой в 1 бита SIRTEST регистра управления тестированием UARTTCR.

В этом режиме данные, передаваемые на выход nSIROUT, должны подаваться на вход SIRIN.

Примечание – Проверка по шлейфу – это единственный случай использования тестового регистра в нормальном режиме функционирования модуля.

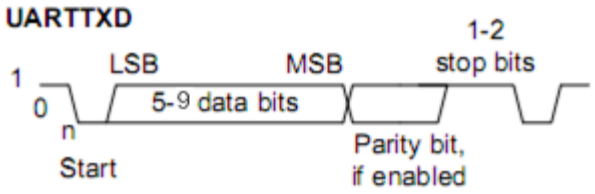



Рисунок 134 – Кадр передачи данных

### 15.2.10 *Модуляция данных IrDA*

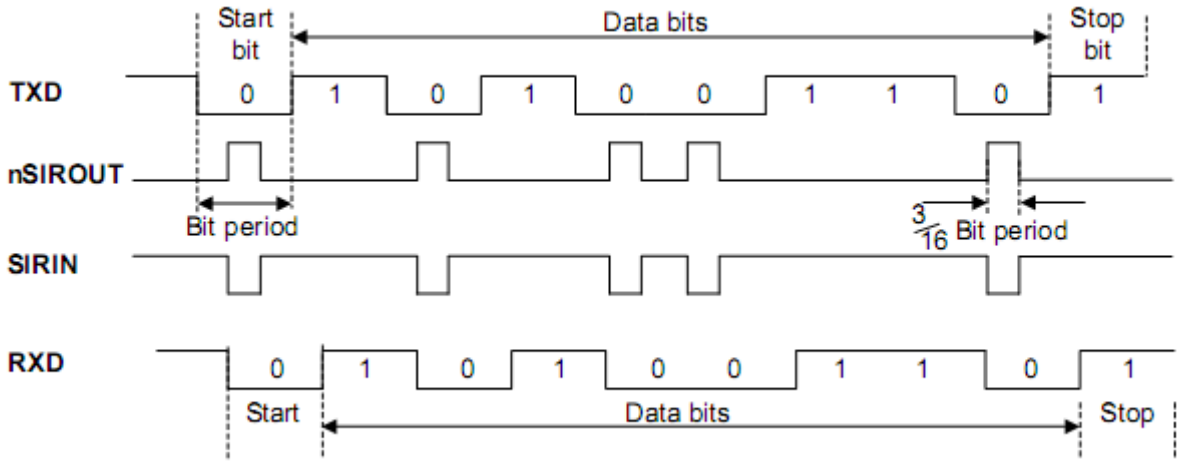



Рисунок 135 – Модуляция данных IrDA

### 15.2.11 *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.

Назначение сигналов в режимах DTE и DCE представлено в таблице 62.

Таблица 62 – 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 |

### 15.2.11.1 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 136.

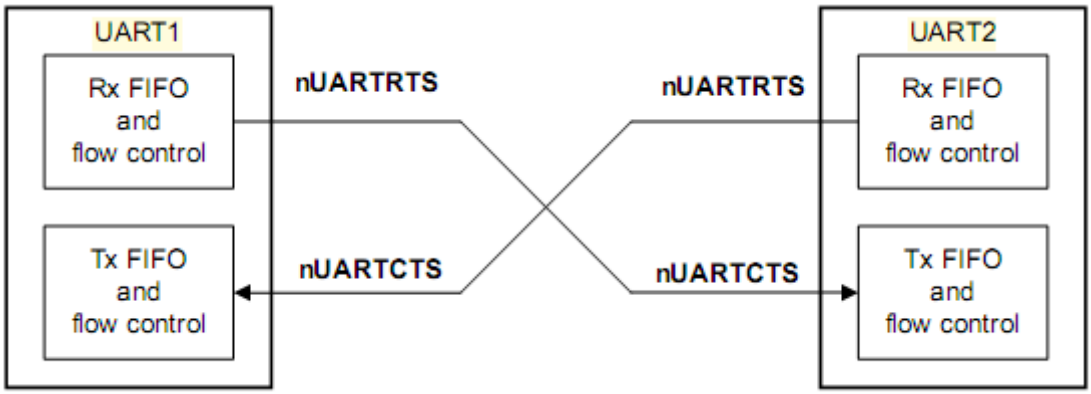



Рисунок 136 – Взаимодействие двух устройств последовательной связи с аппаратным управлением потоком данных

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 показывает необходимые установки для различных режимов управления потоком данных.

Таблица 63 – 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.

### 15.2.11.2 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 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, при этом работа приемопередатчика UART разрешена, прием будет осуществляться до полного заполнения буфера FIFO, либо до завершения передачи данных.

#### 15.2.11.3 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.

### 15.2.12 *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 13 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 взаимоисключаемыми, они могут быть инициированы одновременно. Например, в случае, если заполнение данными буфера приемника превышает пороговое значение, формируется как сигнал запроса одноэлементного обмена, так и сигнал запроса блочного

обмена данными. В случае, если количество данных в буфере приема меньше порогового значения формируется только запрос одноэлементного обмена. Это бывает полезно в ситуациях, при которых объем данных меньше размера блока. Пусть, например, нужно принять 19 символов, а порог заполнения буфера FIFO установлен равным четырем. Тогда контроллер DMA осуществит четыре передачи блоков по четыре символа, а оставшиеся три символа передаст в ходе трех одноэлементных обменов.

Примечание – Для оставшихся трех символов контроллер UART не может инициировать процедуру блочного обмена.

Каждый инициированный приемопередатчиком сигнал запроса DMA остается активным до момента его сброса соответствующим сигналом DMACLR.

После снятия сигнала сброса модуль приемопередатчика вновь получает возможность сформировать запрос на DMA в случае выполнения описанных выше условий. Все запросы DMA снимаются после запрета работы приемопередатчика, а также в случае установки в ноль бита управления DMA TXDMAE или RXDMAE в регистре управления DMA UARTDMACR.

В случае запрета буферов FIFO устройство способно передавать и принимать только одиночные символы; как следствие, контроллер может инициировать DMA только в одноэлементном режиме. При этом модуль в состоянии формировать только сигналы управления DMA UARTRXDMASREQ и UARTTXDMASREQ. Для информации о запрете буферов FIFO см. описание регистра управления линией UARTLCR_H.

Когда буферы FIFO включены, обмен данными может производиться в ходе как одноэлементных, так и блочных передач данных, в зависимости от установленной величины порога заполнения буферов и их фактического заполнения. Таблица показывает значения параметров срабатывания запросов блочного обмена UARTRXDMABREQ и UARTTXDMABREQ в зависимости от порога заполнения буфера.

Таблица 64 – Параметры срабатывания запросов блочного обмена данными в режиме DMA

| **Пороговый уровень** | **Длина блока обмена данными** | |
|---|---|---|
| | **Буфер передатчика (количество незаполненных ячеек)** | **Буфер приемника (количество заполненных ячеек)** |
| 1/8 | 28 | 4 |
| 1/4 | 24 | 8 |
| 1/2 | 16 | 16 |
| 3/4 | 8 | 24 |
| 7/8 | 4 | 28 |

В регистре управления DMA UARTDMACR предусмотрен бит DMAONERR, который позволяет запретить DMA от приемника в случае активного состояния линии прерывания по обнаружению ошибки UARTEINTR. При этом соответствующие линии запроса DMA – UARTRXDMASREQ и UARTRXDMABREQ переводятся в неактивное состояние (маскируются) до сброса UARTEINTR. На линии запроса DMA, обслуживающие передатчик, состояние UARTEINTR не влияет.

На рисунке 137 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.

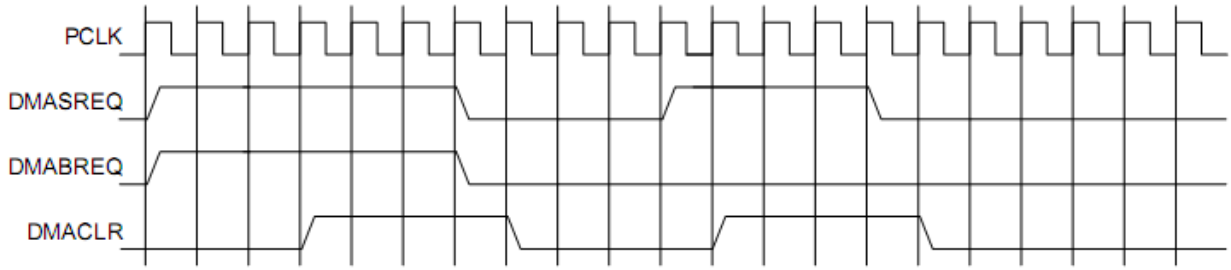



Рисунок 137 – Временные диаграммы одноэлементного и блочного запросов DMA

### 15.2.13 *Прерывания*

В модуле предусмотрено 14 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.
- UARTRNEINTR – FIFO приемника не пусто.
- UARTTFEINTR – FIFO передатчика пусто.
- UARTTNBSYINTR – сдвиговый регистр передатчика пуст.
- 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, UARTEINTR, UARTRNEINTR, UARTTFEINTR и UARTTNBSYINTR.

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.

Прерывание UARTRNEINTR возникает в случае, если FIFO приемника получает хотя бы одно слово данных, то есть становится не пусто.

Прерывание UARTTFEINTR выставляется, если FIFO передатчика не имеет никаких данных, то есть пусто в данный момент.

UARTTNBSYINTR выставляется в том случае, если сдвиговый регистр передатчика пуст. Прерывание активно, когда контроллер UART не передаёт данные.

Признаки возникновения каждого из условий прерывания можно считать либо из регистра прерываний UARTRIS, либо из маскированного регистра прерываний UARTMIS.

#### 15.2.13.1 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.

#### 15.2.13.2 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.

#### 15.2.13.3 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 основано на переходе через пороговое значение, а не на состоянии заполненности буфера FIFO передатчика относительно порогового значения. 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.

#### 15.2.13.4 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.

#### 15.2.13.5 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.

#### 15.2.13.6 UARTINTR

Все описанные сигналы запроса на прерывание скомбинированы в общую линию путем объединения по схеме ИЛИ сигналов UARTRXINTR, UARTTXINTR, UARTRTINTR, UARTMSINTR, UARTEINTR, UARTRNEINTR, UARTTFEINTR и UARTTNBSYINTR с учетом маскирования. Общий выход может быть подключен к системному контроллеру прерываний, что позволит ввести дополнительное маскирование запросов на уровне периферийных устройств.

## 15.3 Контроллер Ethernet (ETHERNETMAC_CNTR)

Функции контроллера Ethernet MAC уровня:

– формирование пакета уровня звена данных протоколов Ethernet/IEEE802.3 и передача его на физический уровень;
– прием с физического уровня и разбор пакета уровня звена данных протоколов Ethernet/IEEE802.3.

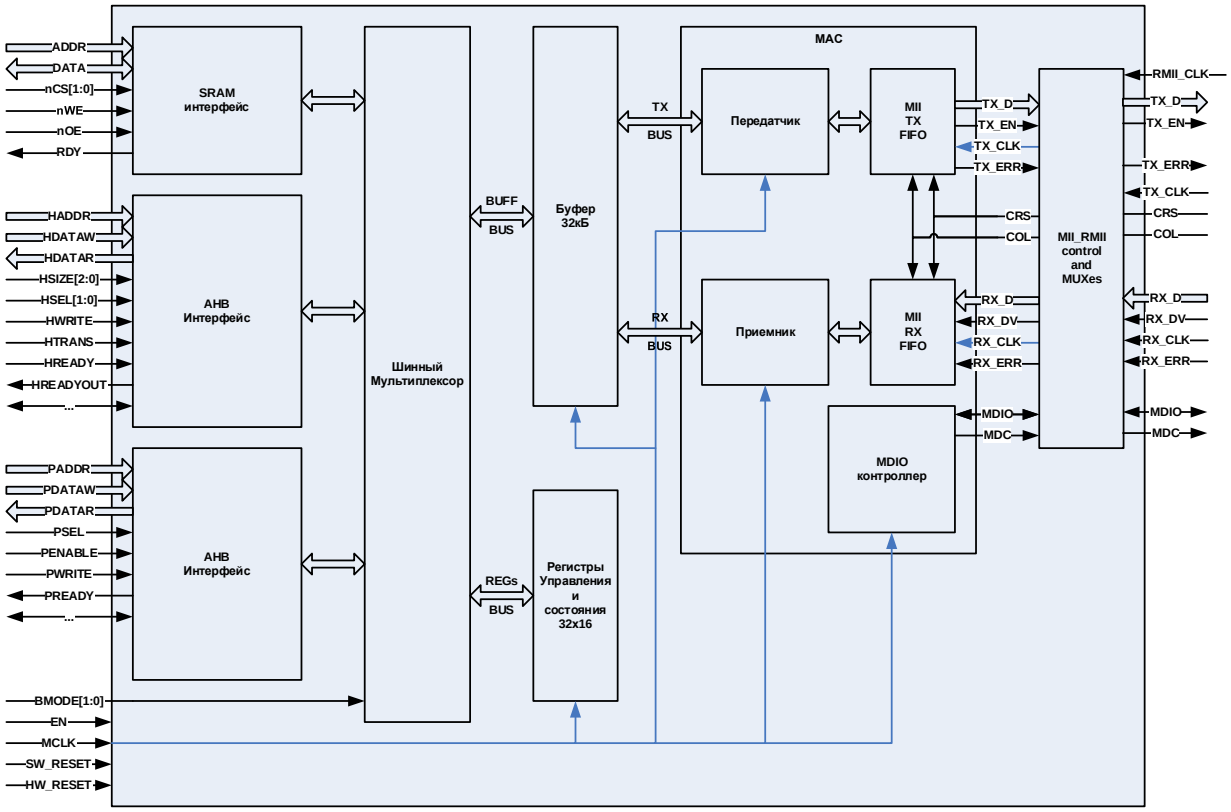



Рисунок 138 – Структурная схема контроллера Ethernet MAC

Работа блока возможна в полно- и полудуплексном режимах. Переключение режима осуществляется битом G_CFG.HD_EN («1» - полудуплексный режим работы).

Также в общее управление блоком входит разрешение обработки пакета PAUSE (G_CFG.PAUSE_EN), управление размером окна распознаваний коллизий (G_CFG.ColWnd). И режимом работы буферов приемника и передатчика (G_CFG.BUFF_MODE). Режимы работы буферов линейный (BUFF_MODE=00b), автоматический (BUFF_MODE=01b) и режим FIFO (BUFF_MODE=10b). Режимы различаются способом обработки указателей границ пустых и полных областей буферов приемника и передатчика. В линейном режиме границы начала пустой области буфера приемника (head_R) и передатчика (tail_X) определяются записью в соответствующие регистры управления. В автоматическом режиме эти границы определяются автоматически – по адресам последнего чтения или записи в соответствующие области. В режиме FIFO границы определяются автоматически, при этом обращение к приемнику производится через адрес 0x0000, а к передатчику через адрес 0x0004.

### 15.3.1 *Передача пакета*

Для передачи пакета(ов) необходима предварительная настройка блока передатчика MAC. Настройка параметров работы передатчика осуществляется посредством регистра управления передатчика X_CFG:

– разрешение работы передатчика (бит EN);
– порядок следования байт в буфере (бит BE);
– порядок следования бит в байте (бит MSB1st);

- выбор события при передаче, выводимого на вывод EVNT[1] (поле EVNT_MODE);
- управление дополнением пакета до минимальной длины PAD-ами (бит PAD_EN);
- управление дополнением пакета преамбулой (бит PRE_EN), SFD добавляется в любом случае;
- управление дополнением пакета, автоматически подсчитываемым полем CRC (CRC_EN);
- управление интервалом между отправлением пакетов (бит IPG_EN);
- управление максимальным числом повторений (поле RtryCnt).

Далее необходимо записать пакет для передачи в буфер передатчика.

Пакет для передачи содержит три поля (все поля должны быть выровнены по границе слова буферного ОЗУ):

- поле управления передачей пакета;
- собственно данные пакета уровня звена данных;
- поле состояния передачи пакета.

Поле управления содержит количество передаваемых данных пакета в байтах. Поле состояния заполняется по завершении процедуры отправки пакета (успешной или нет) и содержит статусную информацию по отправке пакета: о наличии ошибок при его передаче, о количестве попыток передачи пакета и прочее.

Если выбран линейный режим работы буферов, то после помещения пакета для передачи, необходимо записать в регистр управления tail_X границу пустой области (адрес, следующий за последним словом пакета).

По завершении передачи пакета, блок MAC выставит один из флагов прерываний передатчика.

Во время отправки пакета или по его завершении выставляется событие, запрограммированное в поле X_CFG.EVNT_MODE.

### 15.3.2 *Принцип работы передатчика*

Передатчик начинает работать, прочитав ненулевое поле длины из буфера передатчика. Для этого необходимо чтобы буфер передатчика был не пуст (ненулевая разница между значениями head_X и tail_X) и передатчику было разрешено работать (X_CFG.EN=1). Прочитав слово управления, передатчик перемещает указатель head_X на первое слово пакета данных. При получении управляющего слова, в передатчике также фиксируется вся управляющая информация для работы передатчика, препятствуя срыву передачи текущего пакета и позволяя сменить настройки для отправки следующего пакета во время передачи текущего.

По завершении передачи в слово, следующее за последним словом данных, записывается статусная информация отправки пакета.

### 15.3.3 *Прием пакета*

Для приема пакета(ов) необходима предварительная настройка блока приемника MAC. Настройка параметров работы приемника осуществляется посредством регистра управления приемника R_CFG:

- разрешение работы приемника (бит EN);
- порядок следования байт в буфере (бит BE);
- порядок следования бит в байте (бит MSB1st);
- выбор события при передаче, выводимого на вывод EVNT[0] (поле EVNT_MODE);
- управление разрешением приема пакетов:
- длины меньше минимально разрешенной (SF_EN);
- длины больше максимально разрешенной (LF_EN);
- пакетов управления (CF_EN);

- пакетов, содержащих ошибки (EF_EN);
- управление фильтрацией по MAC-адресу:
- разрешение приема пакетов с заданным MAC-адресом (UCA_EN);
- разрешение приема пакетов с широковещательным MAC-адресом (BCA_EN);
- разрешение приема пакетов с групповым MAC-адресом (MCA_EN);
- разрешение приема пакетов с любым MAC-адресом (AC_EN).

Для приема пакета необходимо, чтобы в буфере приемника было достаточно пустого места для принимаемого пакета.

Принятый пакет содержит два поля (все поля выровнены по границе слова буферного ОЗУ):

- поле состояния приема пакета;
- собственно данные пакета уровня звена данных.

Поле состояния заполняется по успешном завершении процедуры приема пакета и содержит количество байтов в пакете (включая заголовок пакет уровня звена дынных) а также статусную информацию по приему пакета, о наличии ошибок при приеме.

Если выбран линейный режим работы буферов, то начало области, свободной для приема данных, указывается в регистре head_R.

По завершении приема пакета, блок MAC выставит один из флагов прерываний приемника.

Во время приема пакета или по его завершении выставляется событие, запрограммированное в поле R_CFG.EVNT_MODE.

### 15.3.4 *Принцип работы приемника*

Приемник начинает работать сразу же после разрешения работы приемника в регистре R_CFG (R_CFG.EN = 1), после обнаружения свободного места в буфере приемника. Обнаружив наличие свободного места, приемник фиксирует всю управляющую информацию для работы приемника, препятствуя срыву приема изменениями настроек и позволяя сменить настройки для приема следующего пакета, и переходит в режим ожидания данных на входе, после поступления данных – в режим приема. По завершении приема в слово, следующее за последним словом данных, записывается статусная информация по приему пакета. Пакеты, отброшенные по причине ошибок в них, или не прошедшие фильтрацию по MAC-адресу, переводят приемник в режим ожидания нового пакета, т.о. не изменяя общего состояния приемника, лишь изменяя состояние регистра флагов прерываний.

### 15.3.5 *Линейный режим работы буферов*

Данный режим включается сбросом поля BUFF_MODE регистра G_CFG (G_CFG.BUFF_MODE = 2’b00). В данном режиме всё управление границами свободных областей в буферах осуществляется вручную.

### 15.3.6 *Автоматический режим работы буферов*

Для включения данного режима необходимо установить значение 1 в поле BUFF_MODE регистра G_CFG (G_CFG.BUFF_MODE = 2’b01). В данном режиме в буфере автоматически отслеживаются указатели границ достоверных данных для передачи и приема по адресу записи в буфер передатчика и адресу чтения из буфера приемника. В данном режиме нет необходимости ручного управления границей свободного места в приемнике и передатчике через запись в соответствующие регистры. Это позволяет упростить алгоритм запуска передачи и приема и передавать данные одновременно с их помещением в буфер передатчика. Граница достоверных данных в буфере приемника перемещается по завершении приема пакета и, т.о., данный режим не допускает одновременного приема пакета и его чтения из буфера. Для обеспечения корректной

работы в этом режиме необходимо активировать биты G_CFG.DBG_XF_EN и G_CFG.DBG_RF_EN.

### 15.3.7 *Режим FIFO работы буферов*

Режим FIFO отличается от предыдущих режимов полностью автоматическим отслеживанием данных в буфере. В данный режим модель переводится установкой значения 2 в поле BUFF_MODE регистра G_CFG (G_CFG.BUFF_MODE = 2'b10). В данном режиме чтение/запись в буферы статусной и управляющей информации, а также данных осуществляются через один адрес – 0x0000 для приемника и 0x0004 для передатчика. Для обеспечения корректной работы в этом режиме необходимо активировать биты G_CFG.DBG_XF_EN и G_CFG.DBG_RF_EN. Данный режим позволяет работать на максимальной скорости, если инструментальные средства управляющего контроллера не обеспечивают режимы адресации с автоинкрементом и циклической буферизацией.

### 15.3.8 *События приемника и передатчика*

В блоке MAC присутствуют 2 вывода индикации событий: события передатчика (EVNT[1]) и приемника (EVNT[0]). Основным назначением этих выводов является информирование управляющего процессора или DMA-контроллера о наличии данных для перемещения. Выводы EVNT программируемые. В качестве источника события могут быть выбраны:

- состояние буфера;
- начало приема/передачи пакета;
- завершение приема/передачи пакета;
- перемещение слова из/в буфера.

Первое событие предназначено для непрерывного обмена информацией с контролем состояния приема/передачи пакетов при использовании высокоуровневых протоколов. Следующие два события обеспечивают прием и передачу пакетов в интерактивном режиме с непосредственным контролем их приема и передачи. Последнее событие предназначено для непосредственного контроля записи/чтения слов данных в/из буферов, и основное назначение этого события – режим отладки.

### 15.3.9 *Прерывания*

Прием и передача пакетов сопровождаются не только формированием событий, но и формированием прерываний, которые служат для непосредственного отражения оперативной статусной информации о состоянии Ethernet-контроллера. Вывод прерывания один, он обеспечивает общую индикацию наличия флагов в регистре флагов прерываний (IFR), разрешенных регистром маски прерываний (IMR). Все прерывания маскируемые. 1 в бите регистра маски прерываний (IMR) разрешает соответствующее прерывание, 0 – запрещает соответствующее прерывание. Прерывания делятся на три группы:

- прерывания MDIO интерфейса;
- прерывания передатчика;
- прерывания приемника.

Прерывания MDIO интерфейса информируют о завершении затребованной операции по MDIO интерфейсу.

Прерывания передатчика показывают состояние отправки пакетов, включая информацию об успешной отправке или наличии ошибок.

Прерывания приемника отражают состояние приема пакета, включая информацию о приеме пакета без ошибок или наличии ошибок при приеме.

Все флаги прерываний кумулятивные. Сброс флагов производится чтением регистра, если бит RCLR_EN регистра G_CFGI установлен (G_CFGI. RCLR_EN = 1), или записью 1 в соответствующий разряд регистра IFR.

### 15.3.10 *Режим детерминированного времени доставки*

Данный режим является расширением стандарта IEEE 802.3/Ethernet для обеспечения детерминированного времени доставки. Режим включается установкой бита DTRM_EN регистра G_CFGI (G_CFGI.DTRM_EN = 1). Данный режим может использоваться только в полнодуплексном режиме работы (значение бита G_CFGI.HD_EN = 1 блокирует данный режим).

В данном режиме для начала передачи пакета выделяется интервал, размером задаваемым регистром JitterWnd (размер окна = JitterWnd + 1), с периодом задаваемым регистром BAG (период = BAG + 1). Единица измерения периода и размера джиттера задается регистром PSC в тактах основной частоты работы блока (размер единицы = PSC + 1).

### 15.3.11 *Режим КЗ*

В блоке MAC для целей тестирования алгоритмов обработки данных предусмотрен режим короткого замыкания (КЗ). В данном режиме выход передатчика переключается на вход приемника. Также, в данном режиме блок принудительно переводится в полнодуплексный режим работы.

### 15.3.12 *Режим RMIInMII*

Выбор режима работы интерфейса физического уровня между MII и RMII осуществляется битом RMIInMII регистра GCNTR. В том случае, если выбран режим MII (пор умолчанию), блок преобразователя RMIInMII control and MUXes передает сигналы со входов MII_RX_D, MII_RX_DV, MII_RX_ERR, MII_RX_CLK, MII_TX_D, MII_TX_EN, MII_TX_ERR, MII_TX_CLK, MII_COL, MII_CRS в неизменном виде на MAC. В этом режиме необходимо подключать все сигналы.

В том случае, если выбран режим RMII преобразовываются блоком RMIInMII control and MUXes в соответствии со спецификацией на RMII интерфейс.

При этом выходные сигналы MII интерфейса блока преобразуются в сигналы RMII в соответствии с таблицей 65.

Таблица 65 – Соответсвие выводов интерфейсов RMII и MII

| **Номер вывода модуля** | **Выводы модуля** | **Функция вывода в режиме MII** | **Функция вывода в режиме RMII** |
|---|---|---|---|
| 1 | MII_RX_D[3] | MII_RX_D[3] | Не используется |
| 2 | MII_RX_D[2] | MII_RX_D[2] | Не используется |
| 3 | MII_RX_D[1] | MII_RX_D[1] | RMII_RX_D[1] |
| 4 | MII_RX_D[0] | MII_RX_D[0] | MII_RX_D[0] |
| 5 | MII_RX_DV | MII_RX_DV | RMII_CRS_DV |
| 6 | MII_RX_ERR | MII_RX_ERR | Не используется |
| 7 | MII_RX_CLK | MII_RX_CLK | Не используется |
| 8 | MII_TX_D[3] | MII_TX_D[3] | Не используется |
| 9 | MII_TX_D[2] | MII_TX_D[2] | Не используется |
| 10 | MII_TX_D[1] | MII_TX_D[1] | RMII_TX_D[1] |
| 11 | MII_TX_D[0] | MII_TX_D[0] | RMII_TX_D[0] |
| 12 | MII_TX_EN | MII_TX_EN | RMII_TX_EN |
| 13 | MII_TX_ERR | Не используется | Не используется |
| 14 | MII_TX_CLK | MII_TX_CLK | Не используется |
| 15 | MII_COL | MII_COL | Не используется |
| 16 | MII_CRS | MII_CRS | Не используется |
| 17 | RMII_CLK | Не используется | REF_CLK 50 МГц |

## 15.4 Контроллер CAN (CAN_CNTR)

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.

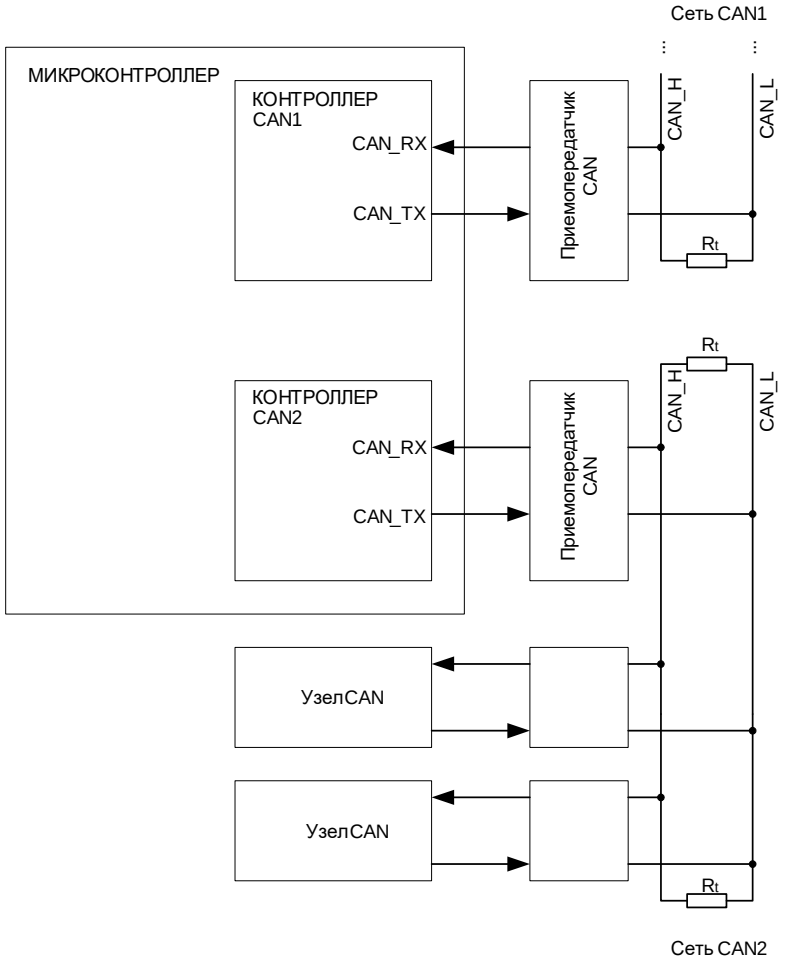



Рисунок 139 – Структурная блок-схема организации сети CAN

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 узла и данные. Все сообщения в протоколе CAN довольно короткие и могут содержать не более восьми байтов данных. При возникновении коллизий (одновременная передача сообщений различными узлами) при передаче идентификатора, происходит арбитраж и узел с большим номером идентификатора уступает сеть узлу с меньшим номером идентификатора.

Особенности:

- поддержка CAN протокола версии CAN2.0 A и B;
- скорость передачи до 1 Мбит/с;
- 32 буфера приема/передачи;
- поддержка приоритетов сообщений;
- 32 фильтра приема;
- маскирование прерываний.

### 15.4.1 *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.

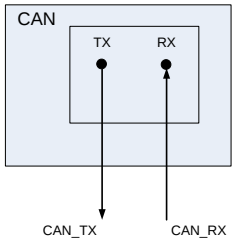



Рисунок 140 – Режим нормальной передачи

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

***Режим работы только на прием - ReceiveOnlyMode (регистр CAN_STATUS: ROM = 1, STM = 0)***

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

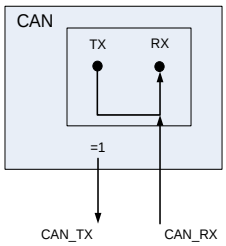



Рисунок 141 – Режим работы только на прием - ReceiveOnlyMode

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

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

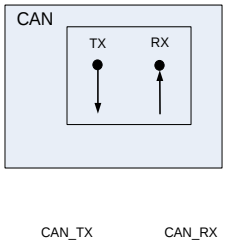



Рисунок 142 – Режим самотестирования - SelfTestMode

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

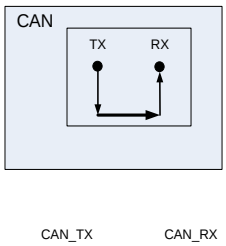



Рисунок 143 – Режим инициализации для задания параметров связи

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 К1986ВЕ9х имеется 32 регистра фильтров и 32 регистра масок, которые можно использовать для блокировки всех CAN-сообщений, кроме избранных сообщений или групп сообщений.

### 15.4.2 *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**. Пакет данных 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 наличие двух или более узлов 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, обеспечивается гарантированность доставки сообщений.

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

### 15.4.3 *Структура пакета данных (DataFrame)*

Пакет данных состоит из 7 различных полей:

- "началопакета" (SOF-startofframe);
- "поле арбитража" (arbitrationfield);
- "поле контроля" (controlfield);
- "поле данных" (datafield);
- "поле CRC" (CRCfield);
- "поле подтверждения" (ACKfield);
- "конец пакета" (endofframe).

Поле данных может иметь нулевую длину.

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

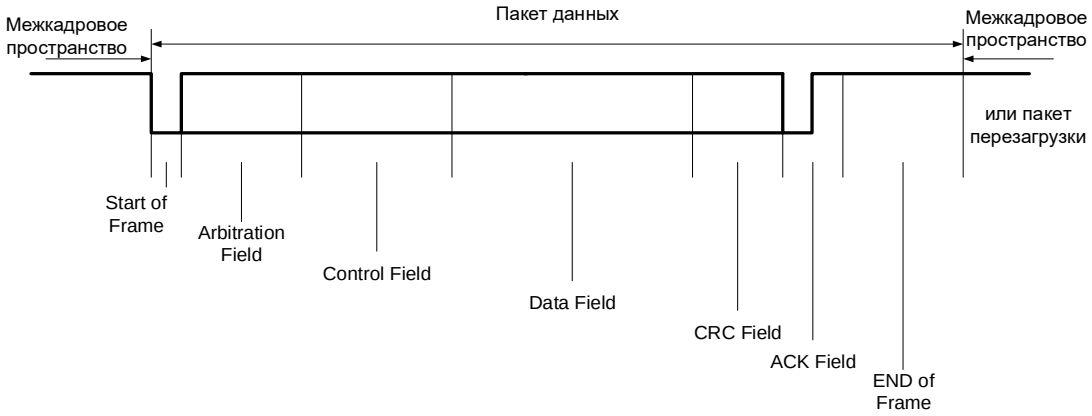



Рисунок 144 – Пакет сообщения CAN

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

#### 15.4.3.1 Начало пакета (Startofframe)

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.

#### 15.4.3.2 Поле арбитража (Arbitrationfield)

Формат поля арбитража отличается для стандартного и расширенного форматов:

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

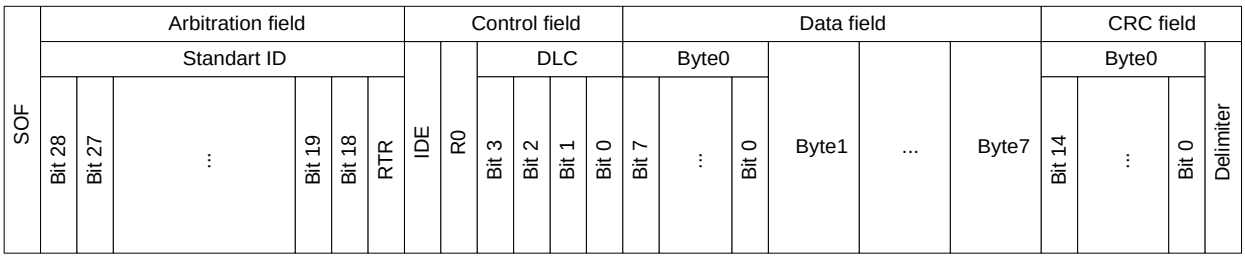



Рисунок 145 – Структура стандартного пакета данных

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

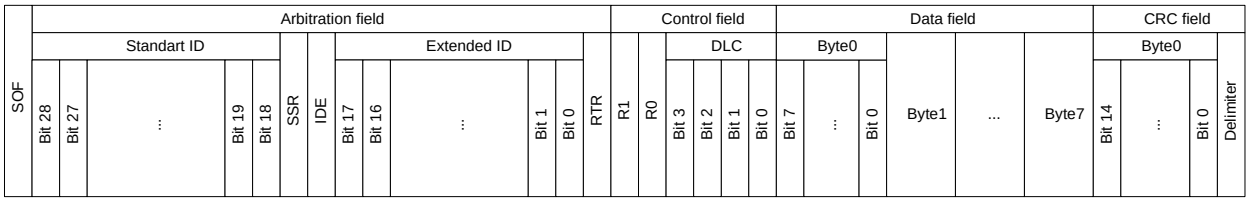



Рисунок 146 – Структура расширенного пакета данных

#### 15.4.3.3 Идентификатор

Идентификатор – стандартный формат. Длина идентификатора – 11 бит и соответствует StandartID в расширенном формате. Эти биты передаются в порядке Bit28…Bit18. Самый младший бит – Bit18. Семь 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:

– **StandartID – 11 бит**
– **ExtendedID – 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.

#### 15.4.3.4 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.

#### 15.4.3.5 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.

#### 15.4.3.6 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.

#### 15.4.3.7 Поле управления (Controlfield)

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.

#### 15.4.3.8 Код длины данных (Datalengthcode)

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.

#### 15.4.3.9 Поле данных (Datafield)

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.

#### 15.4.3.10 Поле CRC (CRCfield)

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.

#### 15.4.3.11 Поле подтверждения (ACKfield)

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".

#### 15.4.3.12 Конецпакета (End of frame)

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

### 15.4.4 *Структура пакета удаленного запроса данных (Remoteframe)*

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.

### 15.4.5 *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 147 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 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.

### 15.4.6 *Инициализация*

Перед началом работы с контроллерами CAN в первую очередь должны быть заданы параметры их тактового сигнала.

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.

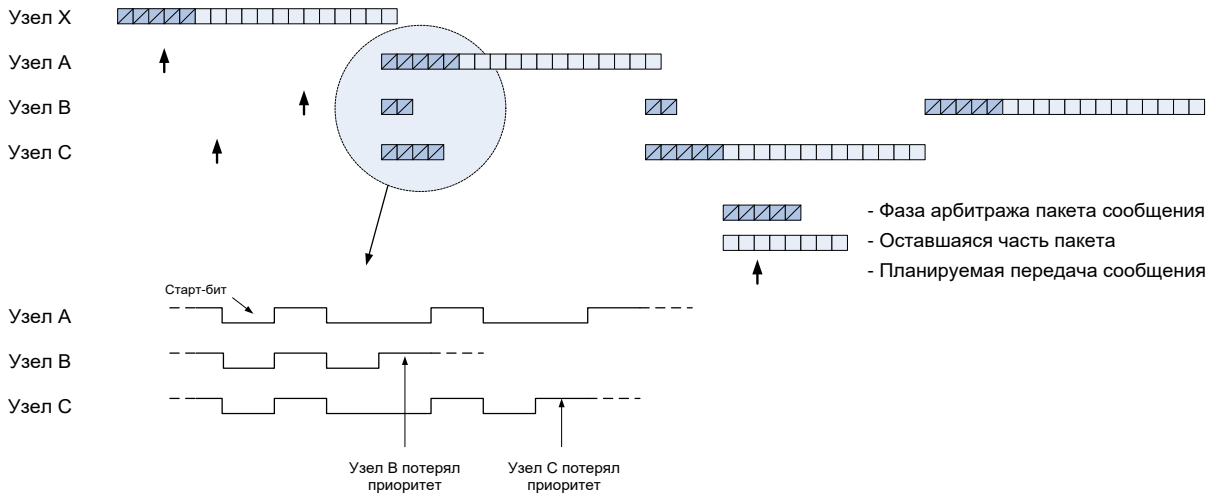



Рисунок 147 – Арбитраж на шине CAN

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

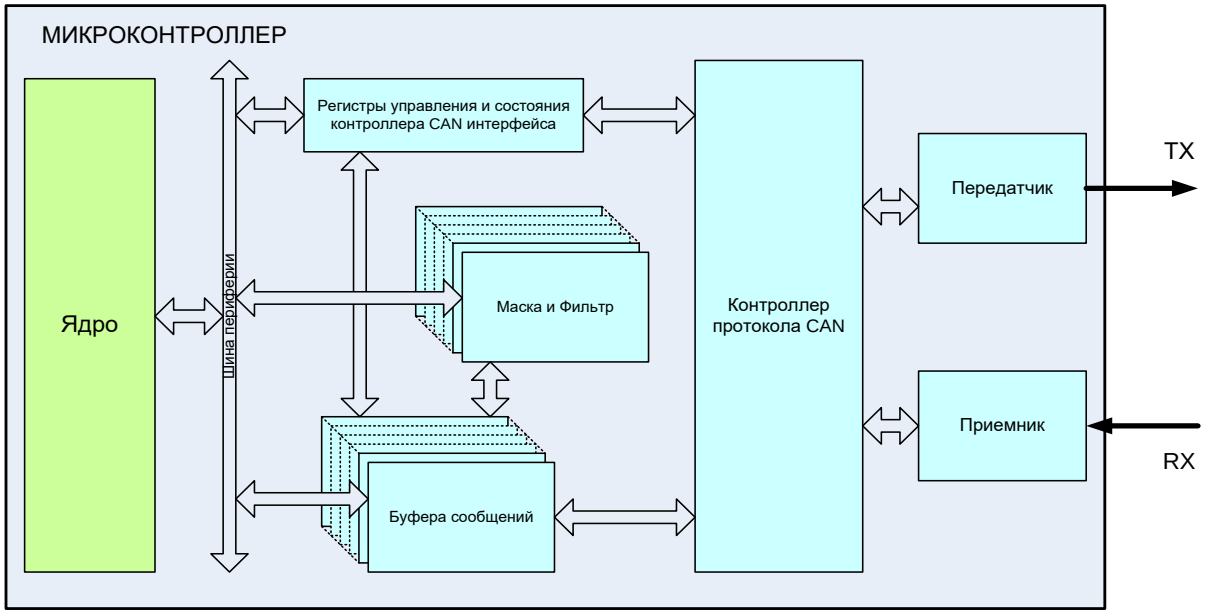



Рисунок 148 – Структурная блок-схема контроллера CAN

### 15.4.7 *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 выставлен бит PRIOR_0, 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.

### 15.4.8 *Передача сообщений по RemoteTransmitRequest (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. Если RTR-запрос удовлетворяет механизму фильтрации нескольких буферов, то ответы будут отправлены из каждого из этих буферов.

### 15.4.9 *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.

### 15.4.10 *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, то оно будет проигнорировано. Если сообщение может быть принято более чем одним буфером, то 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.

### 15.4.11 *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. 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.

### 15.4.12 *Задание скорости передачи и момента сэмплирования*

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 засинхронизироваться 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). Битовыйинтервалсостоитиз 4 частей:

- SynchronizationSegment (Sync_Seg)
- PropagationTimeSegment (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 прескалером BRP со значением от 1 до 65536

TQ (μs) = ((BRP+1))/CANCLK (MHz)

или

TQ (μs) = ((BRP+1)) * Tclk (μs).

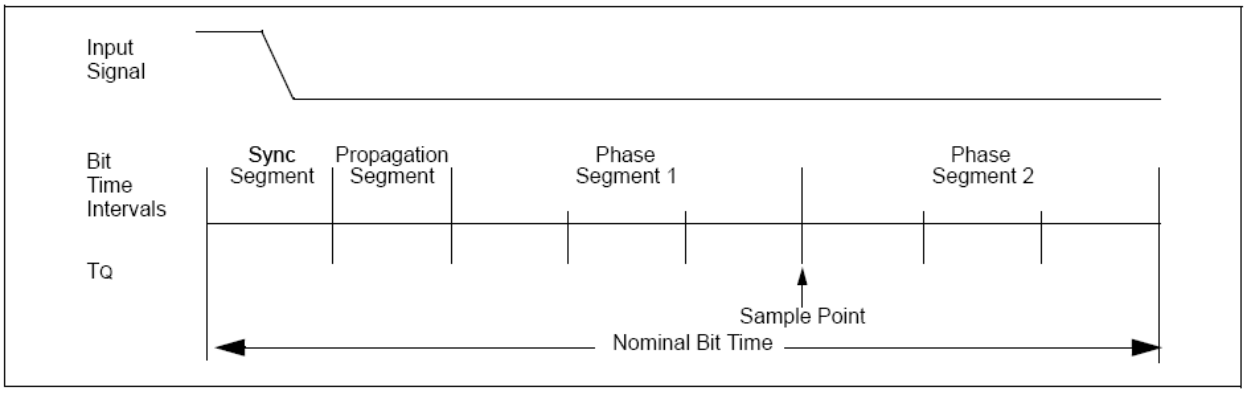



Рисунок 149 – Структура битового интервала

***SynchronizationSegment***

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

***PropagationTimeSegment***

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

***PhaseBufferSegments***

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

### 15.4.13 *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.

***HardSynchronization***

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).

### 15.4.14 *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.

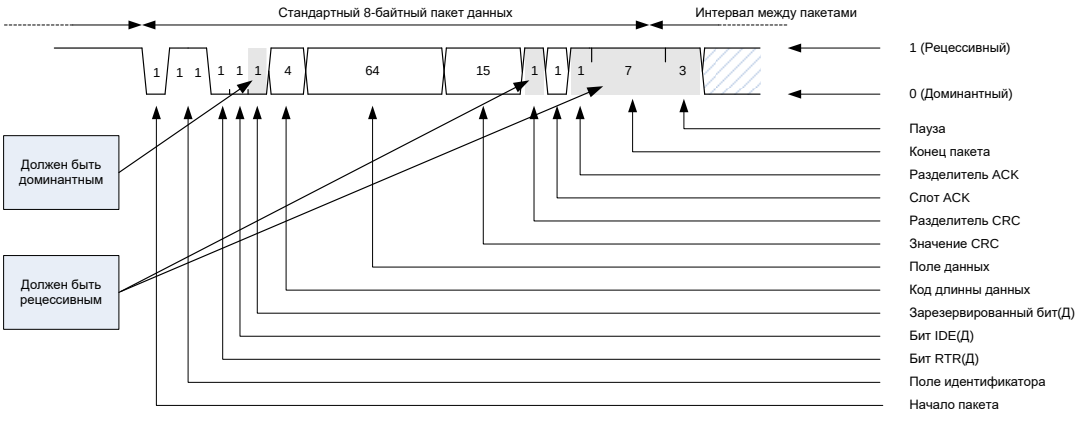



Рисунок 150 – Контроль формата пакета

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.

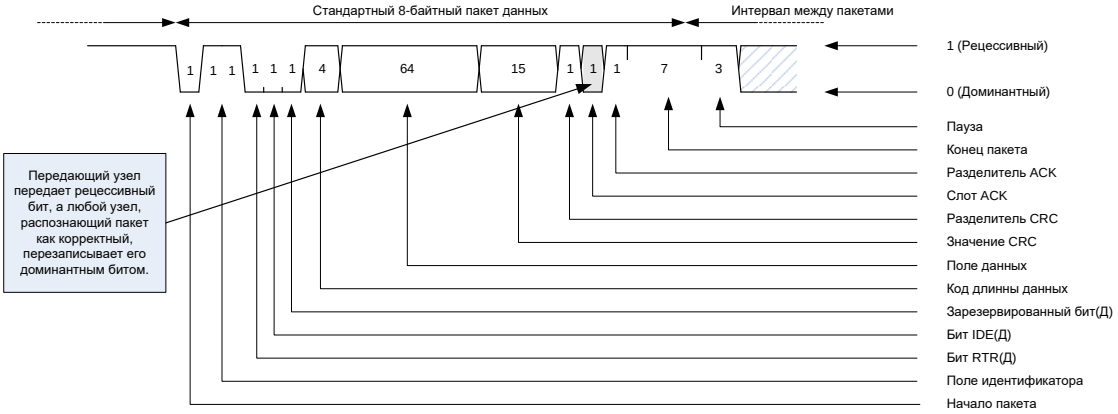



Рисунок 151 – Контроль подтверждения

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.

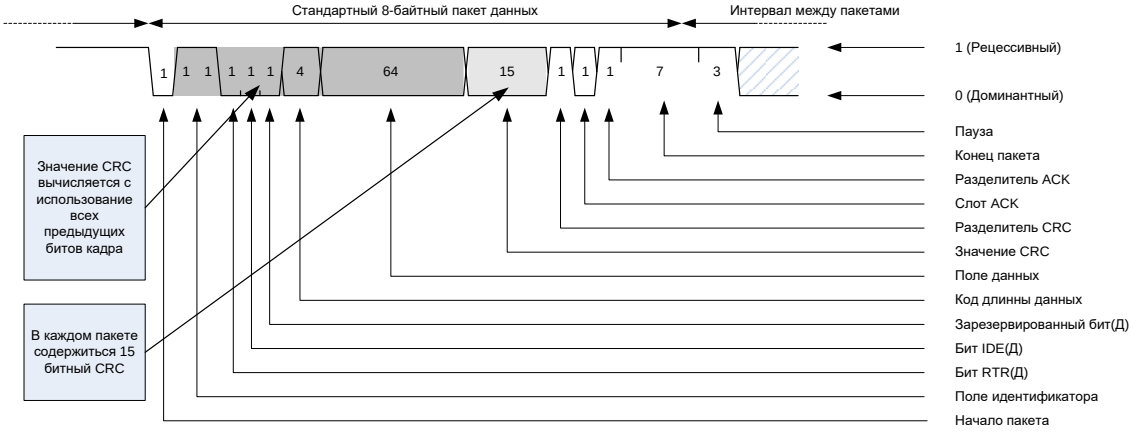



Рисунок 152 – Контроль CRC

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 пакетаBIT_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.

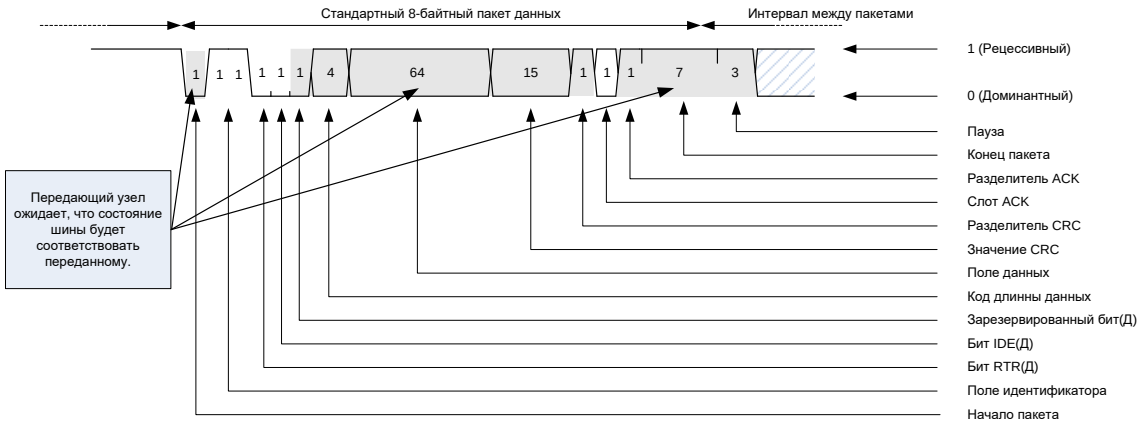



Рисунок 153 – Контроль передаваемых бит

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.

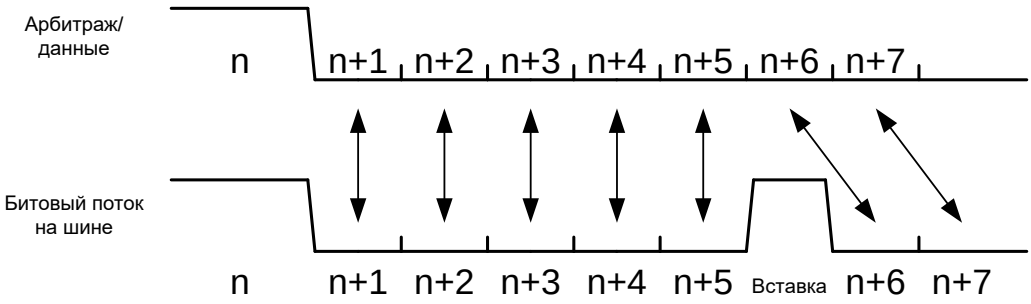



Рисунок 154 – Битстаффинг

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 необходимо вмешательство процессора, который повторно инициализирует контроллер и подключает его обратно к шине. Текущий статус состояния контроллера можно посмотреть в регистре статуса контроллера STATUS.

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.

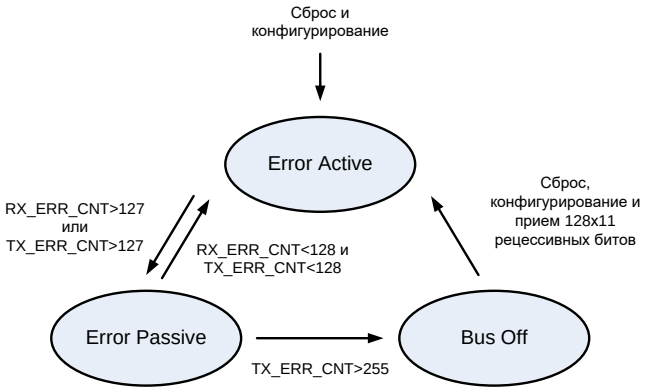



Рисунок 155 – Счетчики ошибок

### 15.4.15 *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).

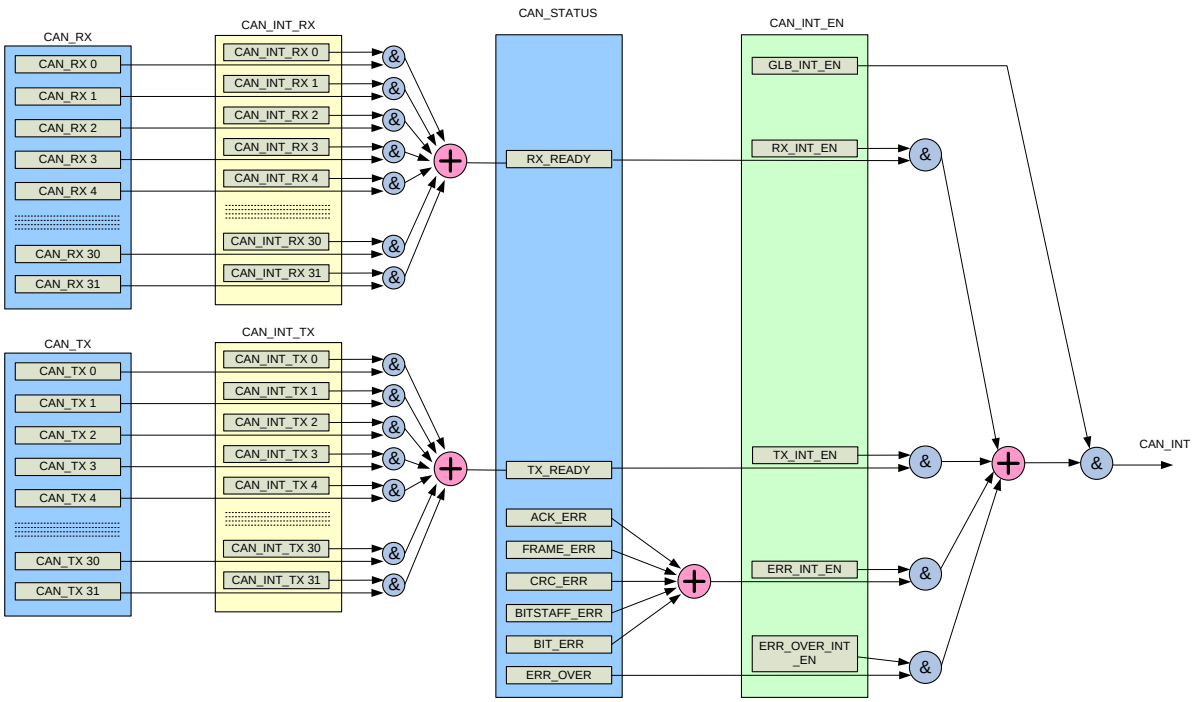



Рисунок 156 – Схема формирования прерывания блока CAN

## 15.5 Контроллер CAN FD

Ниже приведен пример использования контроллера.

Исходные данные:

- В исходном состоянии контроллер выключен.
- Контроллер настраивается на работу с внешними абонентами.
- Контроль устаревания сообщений определяется для каждого пакета.
- Фильтрация отключена.
- Размер очередей на приём и передачу устанавливается максимальным.
- Размер сообщений в очередях устанавливается максимальным.
- Прерывания запрещены.
- Статусный регистр активен при включении CAN. При выключенном CAN статусный регистр принимает нулевое состояние.

### 15.5.1 *Настройка*

1. Произвести настройку параметров работы контроллера, регистры CFG0, CFG1, CFG2, CFG3.
2. Установить маску прерываний, регистр MSK: биты TXQ0E, RXQF, RXQOVL, RXFRMCMPLT, TXFRMCMPLT устанавливаются в единицу.
3. Установить требуемый коэффициент деления системной частоты, регистр PRSC, чтобы на PHY был подан сигнал тактирования необходимой частоты.
4. Установить требуемый коэффициент деления системной частоты, регистр FPRSC, чтобы на PHY был подан сигнал тактирования необходимой частоты при повышенной скорости передачи в режиме CAN FD.
5. Установить значение 2-х фильтров принимаемых сообщений, регистры RXF0 = 0xcafebabe, RXF1 = 0xdeadbeef.
6. Установить маски для 2-х фильтров принимаемых сообщений, регистры RXFMSK0 = 0x1fffffff, RXFMSK1 = 0x1fffffff.
7. Установить разрешение работы фильтров принимаемых сообщений, регистр RXFCFG = 0x00000003.
8. Установить базовые адреса очереди передачи, регистр TXQBADDR0.
9. Установить базовые адреса очереди приема, регистр RXQBADDR.
10. Разрешить работу контроллера, регистр CNTR: EN = 1, ENTX = 1, ENACK = 1, остальные поля устанавливаются в значение 0.

### 15.5.2 *Передача данных*

1. До настройки или после того, как возникнет прерывание, вызванное ST → TXQ0E, пользователь может заполнить в памяти "ячейки" очереди передачи, начиная с адреса TXQBADDR0.
2. В регистре TXQCNTR00 записать биты, соответствующие ячейкам, готовым к передаче. Например, TXQBADDR0 = 0x00000003 (было записано 2 «ячейки» очереди передачи и обе разрешено передать).
3. Ожидаем прерывание, вызванное ST → TXFRMCMPLT, считываем регистр ST, убеждаемся, что во время передачи не возникло ошибок.
4. Возможно ожидание прерывания, вызванное ST → TXQ0E.
5. Контролируем отсутствие ошибок передачи по состоянию ST.
6. Дополнительно можно проанализировать регистры TXQCNTR00, TXQCNTR01, TXQCNTR02, TXQCNTR03 и убедиться, что очереди передачи пусты.
7. Если в процессе передачи возникла ошибка передачи, то, начиная с первой ячейки 0-ой очереди передачи, контролируем в поле статусов ошибки транзакции.

#### 15.5.2.1 Отмена передачи пакета по устареванию

В текущей реализации контроль времени устаревания происходит в двух точках. Первый раз в момент, когда DMA передаёт пакет в буфер. Второй раз перед началом передачи пакета (перед тем как LLC должен дать команду на передачу для уровня MAC).

Такая система позволяет ускорить процесс работы. Пока передается один пакет, мы можем отбросить несколько устаревших из очереди. Разрешение проверки устаревания формируется для каждого пакета независимо.

#### 15.5.2.2 Отмена передачи пакета CPU

CPU имеет возможность отменить или заменить любой пакет из очереди на передачу. Процесс отмены должен проходить следующим образом. Пользователь должен снять бит, отвечающий за передаваемый пакет (далее данный бит будем называть QES-бит), из регистра очереди передачи (TXQCNTRkn). Возможно два варианта работы:

1. Пакет будет отменен. В этом случае в регистре TXQCNTRkn QES-бит будет сброшен.
2. Пакет не будет отменен. В регистре TXQCNTRkn QES-бит останется в единице, до момента успешного окончания отправки пакета. Такая ситуация возможна в случае попытки отмены пользователем текущего передаваемого пакета. Пакет принимает статус «текущий передаваемый» с момента, когда LLC выставил бит о начале передачи данного сообщения на модуль mac. Отмена передачи пользователем текущего передаваемого пакета никак не будет влиять на работу CAN. При возникновении описанной ситуации пользователь должен дождаться обнуления QES-бита регистра очереди передачи TXQCNTRkn, прежде чем проводить какие-либо операции с его ячейкой памяти.

Рекомендация к работе пользователя: после операции отмены сообщения нужно вычитать QES-бит из регистра TXQCNTRkn. И уже на основании полученных данных продолжить работу.

#### 15.5.2.3 Изменение очереди передачи пакетов CPU

Изменение очереди передачи пакетов допустимо только через операцию отмены пакета. Пользователь сначала должен отменить передаваемый пакет (процесс отмены передачи пакета описан в пункте «Отмена передачи пакета CPU»), потом переписать ячейку памяти и поставить пакет в очередь.

#### 15.5.2.4 Нагрузка на каналы данных

Примем, что используется шина AMBA шириной 32 бита.

Контроллер CAN и процессорное ядро являются ведущими на AHB части шины.

Для записи в память одной ячейки очереди требуется 80/4 = 20 обращений в память, что займет минимум 20*2 = 40 тактов частоты AHB, если нет поддержки блочного считывания. Для передачи по интерфейсу CAN FD на максимальной скорости максимального пакета данных требуется ориентировочно 1 100 тактов частоты передачи одного бита. Для передачи по интерфейсу CAN FD на максимальной скорости пакета данных в 1 байт требуется оценочно 600 тактов частоты передачи одного бита. Если частоты работы AHB части AMBA и передачи по интерфейсу CAN равны, то за время передачи такого пакета контроллер должен успеть обратиться к памяти за новым пакетом.

При описанном выше процессе передачи данных происходят следующие транзакции с памятью (после настройки и разрешения работы, если передача прошла без ошибок):

- Запись ячеек очереди – 2*20 обращений;
- Запись меток времени (8 байт каждая) для поставленных в очередь ячеек – 2*2 обращений;
- Загрузка ячейки очереди во внутренний буфер контроллера CAN – 2*20 обращений;
- Запись меток времени (8 байт каждая) для переданных ячеек очереди – 2*2 обращений;
- Запись статуса переданных ячеек очереди – 2*1 обращений.

При максимальном размере передаваемого сообщения и одинаковой частоте передачи бита информации и тактирования AHB это занимает 8% от времени передачи сообщения (180/(2*1100)). При минимальном размере передаваемого сообщения и

одинаковой частоте передачи бита информации и тактирования AHB это занимает 2,3% от времени передачи сообщения (28/(2*600)).

### 15.5.3 *Прием данных*

1. В регистрах RXQCNTRn записать биты, соответствующие ячейкам, готовым к приему. Например, RXQCNTR0 = 0x00000003 (3 «ячейки» очереди приема содержат сообщение, остальные готовы к приему).
2. Ожидаем прерывание, вызванное ST.RXFRMCMPLT, считываем регистр ST, убеждаемся, что во время приема не возникло ошибок.
3. Возможно ожидание прерывания, вызванное ST.RXQF.
4. Контролируем отсутствие ошибок приема по состоянию ST.
5. Дополнительно можно проанализировать регистры RXQCNTRn и убедиться, что очередь приема пуста.
6. Если в процессе приема возникла ошибка приема, то принимаемое сообщение не сохраняется.

#### 15.5.3.1 Фильтрация принимаемых сообщений

Фильтрация принимаемых сообщений реализуется на основе регистров RXF0 - RXF15, RXFMSK0 - RXFMSK15 и RXFCFG.

Регистр RXFCFG включает или отключает выбранные фильтры в соответствии с записанным битом. В регистрах RXFn записываются значения ID пакетов, которые планируется принимать.

Регистры RXFMSKn определяют, какие биты из RXFn будут участвовать в сравнении с ID принимаемого по линии пакета. Регистры RXFMSKn и RXFn ставятся в соответствии друг другу, то есть RXFMSK0 определяет биты, участвующие в сравнении, из регистра RXF0 и т.д.

При приёме пакета с базовым ID в фильтрации участвуют только биты с 10 по 0 регистров RXFMSKn и RXFn. При приёме пакета с расширенным ID проверяется весь вектор.

Если ID принимаемого пакета проходит фильтрацию хотя бы по одному из включенных фильтров, пакет будет определен как принятый (если не возникло каких-либо ошибок по приему, не зависящих от фильтрации).

### 15.5.4 *Управление частотой MAC*

Управление частотой MAC-модуля происходит на основе регистров PRSC, FPRSC, CNTR бит EN, и на основе выходного сигнала модуля MAC → FAST_CLK_EN. При включении модуля (CNTR бит EN) частота начинает поступать на модуль MAC, в соответствии с настройкой одного из предделителей. Какой предделитель является управляющим в данный момент определяется на основе сигнала MAC → FAST_CLK_EN. Если FAST_CLK_EN имеет высокий логический уровень, то управление происходит предделителем FPRSC. Если FAST_CLK_EN имеет низкий логический уровень, то управление происходит предделителем PRSC. При записи в регистры предделителей, они блокируются для дальнейшего изменения на время синхронизации в домен CANCLK. Синхронизация регистров предделителей и сигнала EN происходит на входной частоте MAC_CLK, не зависящей от предделителей.

Формулы расчета частоты приведены в описании регистров PRSC, FPRSC.

### 15.5.5 *Синхронизация данных*

Входная частота для модуля MAC является MAC_CLK. MAC_CLK попадает на предделитель и на его выходе формируется частота MAC_CLK_PRSC. На частоте MAC_CLK_PRSC работает модуль MAC и буфер-приёмник. Остальные модули системы работают на частоте HCLK. При обмене данными между модулями задействованы синхронизаторы.

Сигналы CNTR бит EN, CNTR бит RST_MAC, PRSC, FPRSC проходят синхронизацию на частотах HCLK → MAC_CLK.

Сигналы CNTR (кроме бита EN и RST_MAC), CFG0 биты SPW и STM, CFG1 проходят синхронизацию на частотах HCLK → MAC_CLK_PRSC.

Данные от модуля MAC проходят синхронизацию на частотах MAC_CLK_PRSC → HCLK.

Для синхронизации применяется 4-х фазный протокол.

### 15.5.6 *Описание регистров контроллера CAN FD CANFD_CNTR*

| Базовый адрес | Смещение | Название | Состояние после сброса | Описание |
|---|---|---|---|---|
| | | | | **Основные регистры** |
| 0x4008_6000 | 0x000 | CNTR | 32'h0 | Регистр управления |
| | 0x004 | CFG0 | 32'h0 | Регистр конфигурации №0 |
| | 0x008 | CFG1 | 32'h0 | Регистр конфигурации №1 |
| | 0x00c | CFG2 | 32'h0 | Регистр конфигурации №2 |
| | 0x010 | CFG3 | 32'h40604040 | Регистр конфигурации №3 |
| | 0x014 | RXFCFG | 32'h0 | Регистр конфигурации фильтров входящих сообщений |
| | 0x018 | PRSC | 32'h0 | Регистр предделителя |
| | 0x01c | FPRSC | 32'h0 | Регистр «быстрого» предделителя |
| | 0x020 | ST | 32'h0 | Регистр статуса |
| | 0x024 | MSK | 32'h0 | Регистр маски |
| | 0x028 | RXQBADDR | 32'h0 | Регистр базового адреса очереди приема |
| | 0x02c | TXQBADDR0 | 32'h0 | Регистр базового адреса очереди передачи №0 |
| | 0x030 | TXQBADDR1 | 32'h0 | Регистр базового адреса очереди передачи №1 |
| | 0x034 | TXQBADDR2 | 32'h0 | Регистр базового адреса очереди передачи №2 |
| | 0x038 | TXQBADDR3 | 32'h0 | Регистр базового адреса очереди передачи №3 |
| | 0x03c | CURTIMEHo | 32'h0 | Регистр текущего времени счетчика устаревания, старшая часть |
| | 0x040 | CURTIMELo | 32'h0 | Регистр текущего времени счетчика устаревания, младшая часть |
| – | ... | – | – | – |
| | | | | **Регистры управления очередями** |
| | 0x050 | RXQCNTR0 | 32'h0 | Регистр 0 управления очередью приемника |
| | 0x054 | RXQCNTR1 | 32'h0 | Регистр 1 управления очередью приемника |
| | 0x058 | TXQCNTR00 | 32'h0 | Регистр 0 управления очередью передатчика 0 |
| | 0x05c | TXQCNTR01 | 32'h0 | Регистр 1 управления очередью передатчика 0 |
| | 0x060 | TXQCNTR02 | 32'h0 | Регистр 2 управления очередью передатчика 0 |
| | 0x064 | TXQCNTR10 | 32'h0 | Регистр 0 управления очередью передатчика 1 |
| | 0x068 | TXQCNTR11 | 32'h0 | Регистр 1 управления очередью передатчика 1 |
| | 0x06c | TXQCNTR12 | 32'h0 | Регистр 2 управления очередью передатчика 1 |
| | 0x070 | TXQCNTR20 | 32'h0 | Регистр 0 управления очередью передатчика 2 |

| Базовый адрес | Смещение | Название | Состояние после сброса | Описание |
|---|---|---|---|---|
| | 0x074 | TXQCNTR21 | 32'h0 | Регистр 1 управления очередью передатчика 2 |
| | 0x078 | TXQCNTR22 | 32'h0 | Регистр 2 управления очередью передатчика 2 |
| | 0x07c | TXQCNTR30 | 32'h0 | Регистр 0 управления очередью передатчика 3 |
| | 0x080 | TXQCNTR31 | 32'h0 | Регистр 1 управления очередью передатчика 3 |
| | 0x084 | TXQCNTR32 | 32'h0 | Регистр 2 управления очередью передатчика 3 |
| | 0x088 | TXRCNTR00 | 32'h0 | Регистр 0 сброса очереди передатчика 0 |
| | 0x08c | TXRCNTR01 | 32'h0 | Регистр 1 сброса очереди передатчика 0 |
| | 0x090 | TXRCNTR02 | 32'h0 | Регистр 2 сброса очереди передатчика 0 |
| | 0x094 | TXRCNTR10 | 32'h0 | Регистр 0 сброса очереди передатчика 1 |
| | 0x098 | TXRCNTR11 | 32'h0 | Регистр 1 сброса очереди передатчика 1 |
| | 0x09c | TXRCNTR12 | 32'h0 | Регистр 2 сброса очереди передатчика 1 |
| | 0x0a0 | TXRCNTR20 | 32'h0 | Регистр 0 сброса очереди передатчика 2 |
| | 0x0a4 | TXRCNTR21 | 32'h0 | Регистр 1 сброса очереди передатчика 2 |
| | 0x0a8 | TXRCNTR22 | 32'h0 | Регистр 2 сброса очереди передатчика 2 |
| | 0x0ac | TXRCNTR30 | 32'h0 | Регистр 0 сброса очереди передатчика 3 |
| | 0x0b0 | TXRCNTR31 | 32'h0 | Регистр 1 сброса очереди передатчика 3 |
| | 0x0b4 | TXRCNTR32 | 32'h0 | Регистр 2 сброса очереди передатчика 3 |
| – | ... | – | – | – |
| | | | | **Регистры фильтров** |
| | 0x0c0 | RXF0 | 32'h0 | Регистр фильтра 0 входящих сообщений |
| | 0x0c4 | RXF1 | 32'h0 | Регистр фильтра 1 входящих сообщений |
| | 0x0c8 | RXF2 | 32'h0 | Регистр фильтра 2 входящих сообщений |
| | 0x0cc | RXF3 | 32'h0 | Регистр фильтра 3 входящих сообщений |
| | 0x0d0 | RXF4 | 32'h0 | Регистр фильтра 4 входящих сообщений |
| | 0x0d4 | RXF5 | 32'h0 | Регистр фильтра 5 входящих сообщений |
| | 0x0d8 | RXF6 | 32'h0 | Регистр фильтра 6 входящих сообщений |
| | 0x0dc | RXF7 | 32'h0 | Регистр фильтра 7 входящих сообщений |
| | 0x0e0 | RXF8 | 32'h0 | Регистр фильтра 8 входящих сообщений |
| | 0x0e4 | RXF9 | 32'h0 | Регистр фильтра 9 входящих сообщений |

| Базовый адрес | Смещение | Название | Состояние после сброса | Описание |
|---|---|---|---|---|
| | 0x0e8 | RXF10 | 32'h0 | Регистр фильтра 10 входящих сообщений |
| | 0x0ec | RXF11 | 32'h0 | Регистр фильтра 11 входящих сообщений |
| | 0x0f0 | RXF12 | 32'h0 | Регистр фильтра 12 входящих сообщений |
| | 0x0f4 | RXF13 | 32'h0 | Регистр фильтра 13 входящих сообщений |
| | 0x0f8 | RXF14 | 32'h0 | Регистр фильтра 14 входящих сообщений |
| | 0x0fc | RXF15 | 32'h0 | Регистр фильтра 15 входящих сообщений |
| | 0x100 | RXFMSK0 | 32'h0 | Регистр маски фильтра 0 входящих сообщений |
| | 0x104 | RXFMSK1 | 32'h0 | Регистр маски фильтра 1 входящих сообщений |
| | 0x108 | RXFMSK2 | 32'h0 | Регистр маски фильтра 2 входящих сообщений |
| | 0x10c | RXFMSK3 | 32'h0 | Регистр маски фильтра 3 входящих сообщений |
| | 0x110 | RXFMSK4 | 32'h0 | Регистр маски фильтра 4 входящих сообщений |
| | 0x114 | RXFMSK5 | 32'h0 | Регистр маски фильтра 5 входящих сообщений |
| | 0x118 | RXFMSK6 | 32'h0 | Регистр маски фильтра 6 входящих сообщений |
| | 0x11c | RXFMSK7 | 32'h0 | Регистр маски фильтра 7 входящих сообщений |
| | 0x120 | RXFMSK8 | 32'h0 | Регистр маски фильтра 8 входящих сообщений |
| | 0x124 | RXFMSK9 | 32'h0 | Регистр маски фильтра 9 входящих сообщений |
| | 0x128 | RXFMSK10 | 32'h0 | Регистр маски фильтра 10 входящих сообщений |
| | 0x12c | RXFMSK11 | 32'h0 | Регистр маски фильтра 11 входящих сообщений |
| | 0x130 | RXFMSK12 | 32'h0 | Регистр маски фильтра 12 входящих сообщений |
| | 0x134 | RXFMSK13 | 32'h0 | Регистр маски фильтра 13 входящих сообщений |
| | 0x138 | RXFMSK14 | 32'h0 | Регистр маски фильтра 14 входящих сообщений |
| | 0x13c | RXFMSK15 | 32'h0 | Регистр маски фильтра 15 входящих сообщений |
| – | ... | – | – | – |
| | | | | **Регистры счетчиков ошибок** |
| | 0x140 | TECE | 32'h0 | Счетчик канальных ошибок при передаче сообщения (кроме ошибок арбитража) |
| | 0x144 | RECE | 32'h0 | Счетчик канальных ошибок при приеме сообщения (кроме ошибок арбитража) |
| | 0x148 | TEC | 32'h0 | Текущее количество канальных ошибок при передаче сообщения |
| | 0x14c | REC | 32'h0 | Текущее количество канальных ошибок при приеме сообщения |
| | 0x150 | EOVLC | 32'h0 | Счетчик превышения TEC и REC «уровня предупреждения» |

| Базовый адрес | Смещение | Название | Состояние после сброса | Описание |
|---|---|---|---|---|
| | 0x154 | EPSC | 32'h0 | Счетчик переходов канальной части CAN-контроллера в состояние «error passive» |
| | 0x158 | BOSC | 32'h0 | Счетчик переходов канальной части CAN-контроллера в состояние «bus off» |
| | 0x15c | LMSGLC | 32'h0 | Счетчик отброшенных принятых сообщений по причине переполнения очереди приема |

#### 15.5.6.1 Основные регистры

***CNTR***

Таблица 66 – Регистр CNTR

| Base ADDR= | | 0x4008_6000 | | | Offset= | | 0x000 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | CNTR | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | nISO | RST_MAC | UPTIME | SOVLFRM | ENACK | ENTX | CANFD_EN | EN |
| | | | | | | | | | | | | | | | |

Таблица 67 – Назначение разрядов регистра CNTR

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…8 | – | | Зарезервировано |
| 7 | nISO | | Режим расчета CRC для пакетов формата CAN FD<br>1 – расчет CRC по стандарту Bosch;<br>0 – расчет CRC по стандарту ISO |
| 6 | RST_MAC | | Сброс MAC модуля CAN<br>1 – модуль находится под сбросом;<br>0 – модуль в рабочем состоянии |
| 5 | UPTIME | | Обновить регистр времени CURTIMEHo, CURTIMELo.<br>Устанавливается в неактивное состояние аппаратно |
| 4 | SOVLFRM | | Передать Overload Frame.<br>Устанавливается в неактивное состояние аппаратно |
| 3 | ENACK | | Разрешить передачу ACK в ответ на собственные пакеты |
| 2 | ENTX | | Разрешить передачу пакетов |
| 1 | CANFD_EN | | Разрешение работы в формате CAN FD<br>1 – модуль работает в формате CAN FD;<br>0 – модуль работает в формате CAN |
| 0 | EN | | Разрешение работы |

Примечание – При записи в регистр запускается процесс синхронизации управляющих бит на частоту MAC_CLK_PRSC (чатота сформированная на основе предделителя), кроме EN и RST_MAC. Сигналы EN и RST_MAC синхронизируются на частоте MAC_CLK(входная частота модуля MAC, не зависящая от предделителя). Биты CANFD_EN, ENTX, ENACK, SOVLFRM, nISO объединены в одну группу. При изменении любого бита данной группы запись в группу блокируется на время пересинхронизации.

### *CFG0*

Таблица 68 – Регистр конфигурации CFG0

| Base ADDR= | | 0x4008_6000 | | | Offset= | | 0x004 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | CFG0 | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | TSI | | | | STM | SPW | INTGRE | TSR | TECE | RECE | EOVLC | EPSC | BOSC | LMSGLC |
| | | | | | | | | | | | | | | | |

Таблица 69 – Назначение разрядов регистра CFG0

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…14 | – | | Зарезервировано |
| 13…10 | TSI | | 0001 – 1;<br>0010 – 10;<br>0100 – 100;<br>1000 – 1000 |
| 9 | STM | | Режим передачи самому себе:<br>0 – выключен;<br>1 – включен |
| 8 | SPW | | Бит задает режим семплирования бита:<br>0 – одна точка;<br>1 – три точки |
| 7 | INTGRE | | Дожидаться или нет 11-ти рецессивных бит после включения (CNTR.EN → 1)<br>0 – нет;<br>1 – да |
| 6 | TSR | | Сброс счетчика.<br>0 – нет;<br>1 – да |
| 5 | TECE | | Разрешение работы счетчика TECE.<br>0 – нет;<br>1 – да |
| 4 | RECE | | Разрешение работы счетчика RECE.<br>0 – нет;<br>1 – да |
| 3 | EOVLC | | Разрешение работы счетчика EOVLC.<br>0 – нет;<br>1 – да |
| 2 | EPSC | | Разрешение работы счетчика EPSC.<br>0 – нет;<br>1 – да |
| 1 | BOSC | | Разрешение работы счетчика BOSC.<br>0 – нет;<br>1 – да |
| 0 | LMSGLC | | Разрешение работы счетчика LMSGLC.<br>0 – нет;<br>1 – да |

Примечание – При записи в регистр запускается процесс синхронизации управляющих бит SPW и STM на частоту MAC_CLK_PRSC. До момента окончания синхронизации запись в данные биты регистра невозможна.

### *CFG1*

Таблица 70 – Регистр CFG1

| Base ADDR= | | 0x4008_6000 | | | Offset= | | 0x008 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | CFG1 | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| – | PHYLDCOR | | | | | PHERRCOR2 | | | | | PHERRCOR1 | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RJW | | – | FPHERRCOR2 | | | | | FRJW | | – | FPHERRCOR1 | | | | |
| | | | | | | | | | | | | | | | |

Таблица 71 – Назначение разрядов регистра CFG1

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31 | – | | Зарезервировано |
| 30…26 | PHYLDCOR | | Компенсации физической задержки линии |
| 25…21 | PHERRCOR2 | | 2-ой регистр компенсации фазовой ошибки |
| 20…16 | PHERRCOR1 | | 1-ый регистр компенсации фазовой ошибки |
| 15,14 | RJW | | Ширина скачка синхронизации |
| 13 | – | | Зарезервировано |
| 12…8 | FPHERRCOR2 | | 2-ой регистр компенсации фазовой ошибки |
| 7,6 | FRJW | | Ширина скачка быстрой синхронизации |
| 5 | – | | Зарезервировано |
| 4…0 | FPHERRCOR1 | | 1-ый регистр компенсации фазовой ошибки |

Примечание – При записи в регистр запускается процесс синхронизации управляющих бит на частоту MAC_CLK_PRSC. До момента окончания синхронизации запись в регистр невозможна.

### *CFG2*

Таблица 72 – Регистр CFG2

| Base ADDR= | | 0x4008_6000 | | | Offset= | | 0x00c | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | CFG2 | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TECWL | | | | | | | | RECWL | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 73 – Назначение разрядов регистра CFG2

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…15 | – | | Зарезервировано |
| 14…8 | TECWL | | Уровень срабатывания сигнала предупреждения о достижении или превышении значения счетчиком TEC. Когда значение этого счетчика достигает или превышает уровень предупреждения, выставляется соответствующий бит в регистре ST и инкрементируется счетчик EOVLC |
| 7 | – | | Зарезервировано |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 6…0 | RECWL | | Уровень срабатывания сигнала предупреждения о достижении или превышении значения счетчиком REC. Когда значение этого счетчика достигает или превышает уровень предупреждения, выставляется соответствующий бит в регистре ST и инкрементируется счетчик EOVLC |

### *CFG3*

Таблица 74 – Регистр CFG3

| Base ADDR= | | 0x4008_6000 | | | Offset= | | 0x010 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | CFG2 | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | RXQD | | | | | | | | TXQDn | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RXQMMSIZE | | | | | | | | TXQMMSIZE | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 75 – Назначение разрядов регистра CFG3

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31 | – | | Зарезервировано |
| 30…24 | RXQD | | Размер очереди на прием (по умолчанию = 64 сообщения) |
| 23 | – | | Зарезервировано |
| 22…16 | TXQDn | | Размер для всех очередей на передачу (по умолчанию = 96 сообщений) |
| 15 | – | | Зарезервировано |
| 14…8 | RXQMMSIZE | | Максимальный размер сообщений очереди приема [0:64], по умолчанию установлен максимальный (размер только для поля данных) |
| 7 | – | | Зарезервировано |
| 6…0 | TXQMMSIZE | | Максимальный размер сообщений очереди передачи [0:64], по умолчанию установлен максимальный (размер только для поля данных) |

### *RXFCFG*

Таблица 76 – Регистр RXFCFG

| Base ADDR= | | 0x4008_6000 | | | Offset= | | 0x014 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | RXFCFG | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FM | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 77 – Назначение разрядов регистра RXFCFG

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…16 | – | | Зарезервировано |
| 15…0 | FM | | 0 – фильтр выключен;<br>1 – фильтр включен |

***PRSC***

Таблица 78 – Регистр PRSC

| Base ADDR= | | 0x4008_6000 | | | Offset= | | 0x018 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | PRSC | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | BRP | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 79 – Назначение разрядов регистра PRSC

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…10 | – | | Зарезервировано |
| 9…0 | BRP | | Значение предделителя |

Примечание – При записи в регистр запускается процесс синхронизации управляющих бит на частоту MAC_CLK. До момента окончания синхронизации запись в регистр невозможна. Частота работы модуля MAC вычисляется по формуле CLK8M = MAC_CLK/(BRP + 1).

***FPRSC***

Таблица 80 – Регистр FPRSC

| Base ADDR= | | 0x4008_6000 | | | Offset= | | 0x01c | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | FPRSC | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | BRP | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 81 – Назначение разрядов регистра FPRSC

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…10 | – | | Зарезервировано |
| 9…0 | FBRP | | Значение быстрого предделителя |

Примечание – При записи в регистр запускается процесс синхронизации управляющих бит на частоту Техническое описание MAC_CLK. До момента окончания синхронизации запись в регистр невозможна. Частота работы модуля MAC при повышенной скорости передачи вычисляется по формуле CLK8M = MAC_CLK/(FBRP + 1).

### *ST*

Таблица 82 – Регистр ST

| Base ADDR= | | 0x4008_6000 | | | Offset= | | 0x020 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | ST | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | RXQF | TXQ0F | TXQ1F | TXQ2F | TXQ3F | TXQ0E | TXQ1E | TXQ2E | TXQ3E | EAS | EPS | BOS | RXACT |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TXACT | DLC_ERR | RXML | RXQOVL | TECOVL | RECOVL | RXFRMCMPLT | TXFRMCMPLT | ARBL | TXERR | RXERR | BITERR | BITSTAFFERR | ACKERR | CRCERR | FRMERR |
| | | | | | | | | | | | | | | | |

Таблица 83 – Назначение разрядов регистра ST

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…29 | – | | Зарезервировано |
| 28 | RXQF | | (f) Очередь приема полная, места для еще одного пакета нет |
| 27 | TXQ0F | | (f) Очередь 0 на передачи полная |
| 26 | TXQ1F | | (f) Очередь 1 на передачи полная |
| 25 | TXQ2F | | (f) Очередь 2 на передачи полная |
| 24 | TXQ3F | | (f) Очередь 3 на передачи полная |
| 23 | TXQ0E | | (f) Очередь 0 на передачу пустая |
| 22 | TXQ1E | | (f) Очередь 1 на передачу пустая |
| 21 | TXQ2E | | (f) Очередь 2 на передачу пустая |
| 20 | TXQ3E | | (f) Очередь 3 на передачу пустая |
| 19 | EAS | | (f) Контроллер CAN находится в состоянии error active state |
| 18 | EPS | | (f) Контроллер CAN находится в состоянии error passive state |
| 17 | BOS | | (f) Контроллер CAN находится в состоянии bus off state |
| 16 | RXACT | | (f) Идет прием сообщения |
| 15 | TXACT | | (f) Идет передача сообщения |
| 14 | DLC_ERR | | (e) Ошибка DLC.<br>Возникает при приеме, если DLC принятого пакета больше заданного пользователем в RXQMMSIZE. Возникает при передаче, если DLC передаваемого пакета больше, чем размер, определенный в TXQMMSIZE |
| 13 | RXML | | (e) Последнее сообщение в буфере переписано, так как внутренние буферы не свободны, не удалось переместить их содержимое в память системы из-за ее занятости |
| 12 | RXQOVL | | (e) Очередь приема переполнена (одно или два сообщения находятся в буферах) |
| 11 | TECOVL | | (e) Счетчик TEC достиг заданного максимума или превысил его |
| 10 | RECOVL | | (e) Счетчик REC достиг заданного максимума или превысил его |
| 9 | RXFRMCMPLT | | (e) Сообщение полностью принято |
| 8 | TXFRMCMPLT | | (e) Передача сообщения завершена |
| 7 | ARBL | | (e) Потеря арбитража |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 6 | TXERR | | (e) Ошибка передачи |
| 5 | RXERR | | (e) Ошибка приема |
| 4 | BITERR | | (e) Ошибка передачи бита |
| 3 | BITSTAFFERR | | (e) Ошибка бит-стаффинга |
| 2 | ACKERR | | (e) Нет сигнала acknowledge |
| 1 | CRCERR | | (e) Ошибка подсчета CRC |
| 0 | FRMERR | | (e) Ошибка формата сообщения на линии |

Примечание – Устройство начинает выдавать корректный статус только после включения (бит EN регистра CNTR в 1). До этого момента значение статуса – 0x00000000.

Флаг – эти биты характеризуют текущее состояние контроллера CAN. Они устанавливаются в 1, когда указанное условие выполняется, и переходят в 0 сразу, как только оно перестает выполняться. Они НЕ сбрасываются по чтению ST. Прерывание по биту-флагу формируется в момент перехода бита ST из 0 в 1 (но не наоборот). Если бит ST уже находился в 1, и в этот момент разрешается прерывание по этому биту, то прерывание тут же срабатывает.

Событие – эти биты сигнализируют о возникновении определенного события. Они устанавливаются в 1 в момент, когда событие происходит, и сбрасываются в 0 по чтению регистра ST. Прерывание по биту-событию срабатывает в момент возникновения данного события. Следует иметь ввиду, что бит в регистре ST установится в 1 от первого из событий, и, если его не сбросить (прочитав ST), останется в 1. Прерывания же формируются и от последующих событий тоже, неважно, в 0 или в 1 бит в регистре ST.

***MSK***

Таблица 84 – Регистр MSK

| Base ADDR= | | 0x4008_6000 | | | Offset= | | 0x024 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | MSK | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | RXQF | TXQ0F | TXQ1F | TXQ2F | TXQ3F | TXQ0E | TXQ1E | TXQ2E | TXQ3E | EAS | EPS | BOS | RXACT |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TXACT | DLC_ERR | RXML | RXQOVL | TECOVL | RECOVL | RXFRMCMPLT | TXFRMCMPLT | ARBL | TXERR | RXERR | BITERR | BITSTAFFERR | ACKERR | CRCERR | FRMERR |
| | | | | | | | | | | | | | | | |

Таблица 85 – Назначение разрядов регистра MSK

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…29 | – | | Зарезервировано |
| 28 | RXQF | | Очередь приема полная, места для еще одного пакета нет |
| 27 | TXQ0F | | Очередь 0 на передачи полная |
| 26 | TXQ1F | | Очередь 1 на передачи полная |
| 25 | TXQ2F | | Очередь 2 на передачи полная |
| 24 | TXQ3F | | Очередь 3 на передачи полная |
| 23 | TXQ0E | | Очередь 0 на передачу пустая |
| 22 | TXQ1E | | Очередь 1 на передачу пустая |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 21 | TXQ2E | | Очередь 2 на передачу пустая |
| 20 | TXQ3E | | Очередь 3 на передачу пустая |
| 19 | EAS | | Контроллер CAN находится в состоянии error active state |
| 18 | EPS | | Контроллер CAN находится в состоянии error passive state |
| 17 | BOS | | Контроллер CAN находится в состоянии bus off state |
| 16 | RXACT | | Идет прием сообщения |
| 15 | TXACT | | Идет передача сообщения |
| 14 | DLC_ERR | | Маска ошибки DLC.<br>Возникает при приеме, если DLC принятого пакета больше заданного пользователем в RXQMMSIZE.<br>Возникает при передаче, если DLC передаваемого пакета больше, чем размер, определенный в TXQMMSIZE |
| 13 | RXML | | Сообщение не принято (отклонено), так как внутренние буферы не свободны, не удалось переместить их содержимое в память системы из-за ее занятости |
| 12 | RXQOVL | | Очередь приема переполнена, последний принятый пакет удален |
| 11 | TECOVL | | Счетчик TEC достиг заданного максимума или превысил его |
| 10 | RECOVL | | Счетчик REC достиг заданного максимума или превысил его |
| 9 | RXFRMCMPLT | | Сообщение полностью принято |
| 8 | TXFRMCMPLT | | Передача сообщения завершена |
| 7 | ARBL | | Потеря арбитража |
| 6 | TXERR | | Ошибка передачи |
| 5 | RXERR | | Ошибка приема |
| 4 | BITERR | | Ошибка передачи бита |
| 3 | BITSTAFFERR | | Ошибка бит-стаффинга |
| 2 | ACKERR | | Нет сигнала acknowledge |
| 1 | CRCERR | | Ошибка подсчета CRC |
| 0 | FRMERR | | Ошибка формата сообщения на линии |

### *RXQBADDR*

Таблица 86 – Регистры RXQBADDR

| Base ADDR= | | 0x4008_6000 | | | Offset= | | 0x028 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| ADDR | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDR | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 87 – Назначение разрядов регистров RXQBADDR

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | ADDR | | Базовый адрес очереди в памяти |

### *TXQBADDRn*

Таблица 88 – Регистры TXQBADDRn

| Base ADDR= | | 0x4008_6000 | | | Offset= | | 0x02c<br>0x030<br>0x034<br>0x038 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| ADDR | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDR | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 89 – Назначение разрядов регистров TXQBADDRn

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | ADDR | | Базовый адрес очереди в памяти |

### *CURTIMEHo*

Таблица 90 – Регистр CURTIMEHo

| Base ADDR= | | 0x4008_6000 | | | Offset= | | 0x03c | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | CURTIMEHo | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CURENT TIME[63:48] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CURENT TIME[47:32] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 91 – Назначение разрядов регистра CURTIMEHo

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | CURENT TIME[63:32] | | Регистр содержит старшую часть счетчика текущего времени. Обновляется по выставлению бита UPTIME в регистре CNTR |

### *CURTIMELo*

Таблица 92 – Регистр CURTIMELo

| Base ADDR= | | 0x4008_6000 | | | Offset= | | 0x040 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | CURTIMELo | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CURENT TIME[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CURENT TIME[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 93 – Назначение разрядов регистра CURTIMELo

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | CURENT TIME[31:0] | | Регистр содержит младшую часть счетчика текущего времени. Обновляется по выставлению бита UPTIME в регистре CNTR |

#### 15.5.6.2 Регистры управления очередями

***RXQCNTRn***

Таблица 94 – Регистр RXQCNTRn

| Base ADDR= | | 0x4008_6000 | | | Offset= | | 0x050<br>0x054 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| QES | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QES | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 95 – Назначение разрядов регистра RXQCNTRn

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | QES | | Статус элемента очереди – каждый бит отвечает за свой элемент. Полный номер элемента = номер бита *n.<br>1 – ячейка очереди заполнена (прием сообщения полностью завершен);<br>0 – ячейка очереди свободна |

***TXQCNTRkn***

Таблица 96 – Регистр TXQCNTRkn

| Base ADDR= | | 0x4008_6000 | | | Offset= | | 0x058<br>0x05c<br>0x060<br>0x064<br>0x068<br>0x06c<br>0x070<br>0x074<br>0x078<br>0x07c<br>0x080<br>0x084 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| QES | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QES | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 97 – Назначение разрядов регистра TXQCNTRkn

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | QES | | Статус элемента очереди – каждый бит отвечает за свой элемент. Полный номер элемента = номер бита *n.<br>1 – ячейка очереди готова к передаче или передается;<br>0 – ячейка очереди свободна (передача завершена, еще не разрешалась или отменена) |

#### 15.5.6.3 Регистры сброса очередей

***TXRCNTRkn***

Таблица 98 – Регистр TXRCNTRkn

| Base ADDR= | 0x4008_6000 | Offset= | 0x088<br>0x08c<br>0x090<br>0x094<br>0x098<br>0x09c<br>0x0a0<br>0x0a4<br>0x0a8<br>0x0ac<br>0x0b0<br>0x0b4 |
|---|---|---|---|

| REG Name: | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| QER | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| QER | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 99 – Назначение разрядов регистра TXRCNTRkn

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | QER | | Отменить элемент очереди – каждый бит отвечает за свой элемент. Полный номер элемента = номер бита *n.<br>1 – отменить передачу ячейки очереди |

#### 15.5.6.4 Регистры фильтров

***RXFn***

Таблица 100 – Регистр RXFn

| Base ADDR= | 0x4008_6000 | Offset= | 0x0c0<br>0x0c4<br>0x0c8<br>0x0cc<br>0x0d0<br>0x0d4<br>0x0d8<br>0x0dc<br>0x0e0<br>0x0e4<br>0x0e8<br>0x0ec<br>0x0f0<br>0x0f4<br>0x0f8<br>0x0fc |
|---|---|---|---|

| REG Name: | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | ID_E | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ID_E | | | | | BASE_ID | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 101 – Назначение разрядов регистра RXFn

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…29 | – | | Зарезервировано |
| 28…11 | ID_E | | Фильтр RXFn для расширенного идентификатора |
| 10…0 | BASE_ID | | Фильтр RXFn для базового идентификатора |

***RXFMSKn***

Таблица 102 – Регистр RXFMSKn

| Base ADDR= | 0x4008_6000 | Offset= | 0x100<br>0x104<br>0x108<br>0x10c<br>0x110<br>0x114<br>0x118<br>0x11c<br>0x120<br>0x124<br>0x128<br>0x12c<br>0x130<br>0x134<br>0x138<br>0x13c |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | MSK_ID_E | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MSK_ID_E | | | | | MSK_BASE_ID | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 103 – Назначение разрядов регистра RXFMSKn

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…29 | – | | Зарезервировано |
| 28…11 | MSK_ID_E | | Маска фильтра RXFn для расширенного идентификатора.<br>1 – соответствующий бит RXFn участвует в сравнении;<br>0 – соответствующий бит RXFn не участвует в сравнении |
| 10…0 | MSK_BASE_ID | | Маска фильтра RXFn для базового идентификатора.<br>1 – соответствующий бит RXFn участвует в сравнении;<br>0 – соответствующий бит RXFn не участвует в сравнении |

### 15.5.6.5 Регистры счетчиков ошибок

***TECE***

Таблица 104 – Регистр TECE

| Base ADDR= | 0x4008_6000 | Offset= | 0x140 |
|---|---|---|---|
| REG Name: | TECE | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| VAL | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| VAL | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 105 – Назначение разрядов регистра TECE

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | VAL | | Счетчик.<br>Примечание – В отличие от стандартного счетчика TEC этот счетчик инкрементируется всегда на 1 и никогда не декрементируется |

***RECE***

Таблица 106 – Регистр RECE

| Base ADDR= | | 0x4008_6000 | | | Offset= | | 0x144 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | RECE | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| VAL | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| VAL | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 107 – Назначение разрядов регистра RECE

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | VAL | | Счетчик.<br>Примечание – В отличие от стандартного счетчика REC этот счетчик инкрементируется всегда на 1 и никогда не декрементируется |

***TEC***

Таблица 108 – Регистр TEC

| Base ADDR= | | 0x4008_6000 | | | Offset= | | 0x148 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | TEC | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| VAL | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| VAL | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 109 – Назначение разрядов регистра TEC

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | VAL | | |

***REC***

Таблица 110 – Регистр REC

| Base ADDR= | | 0x4008_6000 | | | Offset= | | 0x14c | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | REC | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| VAL | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| VAL | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 111 – Назначение разрядов регистра REC

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | VAL | | |

***EOVLC***

***EPSC***

***BOSC***

***LMSGLS***

Таблица 112 – Регистры EOVLC, EPSC, BOSC, LMSGLS

| Base ADDR= | 0x4008_6000 | Offset= | 0x150<br>0x154<br>0x158<br>0x15c | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| VAL | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| VAL | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 113 – Назначение разрядов регистров EOVLC, EPSC, BOSC, LMSGLS

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | VAL | | |

## 15.6 Контроллер МКПД (MIL_CNTR)

В микроконтроллере имеется два независимых контроллера интерфейса мультиплексного канала передачи данных по ГОСТ Р 52070-2003 (далее МКПД). Каждый из которых, содержит необходимую логику и память для обработки и хранения командных слов и слов данных одного полного сообщения МКПД. Каждый контроллер содержит два канала для приёма/передачи сообщений МКПД: основной и резервный. В один момент времени может работать только один из каналов основной или резервный. Одновременная работа двух каналов не предусмотрена. Контроллер может работать как в режиме контролера шины, так и в режиме оконечного устройства. Для хранения входящих и исходящих командных и статусных слов, а также команд управления используются 16-разрядные регистры. Для хранения данных используется шестнадцатиразрядная двухпортовая память, в которой данные хранятся в области памяти соответствующей подадресу командного слова. В каждом подадресе можно хранить только одно полное сообщение МКПД. При передаче сообщения данные в память можно заносить как на «лету», так и до начала передачи. При приёме сообщения, данные можно считывать из памяти, как на «лету», так и после установки флага VALMESS.

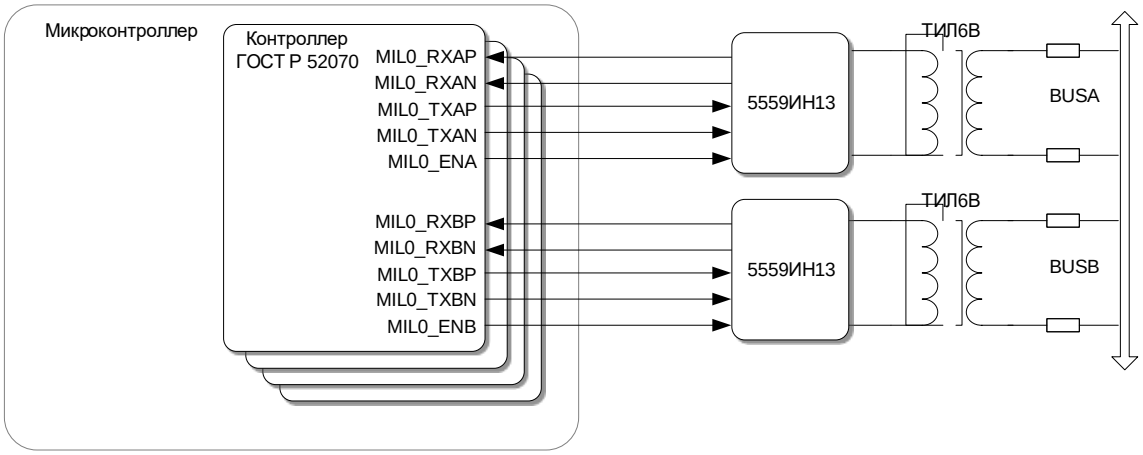



Рисунок 157 – Структурная схема контроллера интерфейса МКПД

Особенности:

- поддержка основных (формат 1 – формат 6) и групповых (формат 7 – формат 10) форматов сообщений;
- поддержка режимов работы: контроллер шины, оконечное устройство, монитор;
- скорость передачи данных 1 Мбит/с в полудуплексном режиме;
- поддержка двух каналов связи: основного и резервного;
- двухпортовая память принимаемых данных 1Кх16;
- двухпортовая память передаваемых данных 1Кх16;
- возможность формирования прерываний при успешном приёме и при возникновении ошибок на шине;
- маскирование прерываний.

### 15.6.1 *Режимы работы*

Контроллер поддерживает три режима работы: контроллера шина (КШ), оконечного устройства (ОУ) и неадресуемого монитора (М).

Контроллер шины

В этом режиме контроллер передаёт команды в магистраль, участвует в пересылке слов данных, принимает и контролирует ответную информацию о состоянии ОУ. Помимо этого, КШ реализует все команды управления. Для того чтобы реализовать передачу командного слова в магистраль используется регистр CommandWord1. А для сообщений формата 3 и 8 помимо этого применяется регистр CommandWord2. Ответная информация о состоянии ОУ после приёма из магистрали хранится в регистре StatusWord1. Для сообщений формата 3 и 8 помимо этого применяется регистр StatusWord2. Для передачи и приёма слов данных команд управления (КУ), форматы сообщений 5, 6 и 10, применяется регистр ModeData. Выбор этого режима работы осуществляется в регистре CONTROL установкой бита BCMODE и сбросом RTMODE.

Оконечное устройство

В этом режиме контроллер осуществляет проверку достоверности командных слов, поступающих к нему от КШ. Командное слово считается достоверным, если не возникло ошибок в магистрали при его приёме, или если поле «Адрес ОУ» соответствует коду собственного адреса ОУ или коду 11111 (групповая команда). Если командное слово определено как достоверное, то ОУ посылает в линию ответное слово (ОС) и в зависимости от поля «Приём/Передача» принимает или передаёт число данных, соответствующее полю «Число СД/Код КУ». Если же происходит приём от КШ команды управления, то ОУ реагирует в соответствии с форматами сообщений команд управления.

Принятое из магистрали командное слово помещается в регистр CommandWord1, а для сообщений формата 3 и 8 принятое второе командное слово помещается в регистр CommandWord2. Ответное слово ОУ для передачи в магистраль помещается в регистр StatusWord1. Помимо этого, для сообщения формата 3, этот регистр содержит принятое ответное слово от другого ОУ. Для передачи и приёма слов данных команд управления, форматы сообщений 5, 6 и 10, применяется регистр ModeData. Выбор этого режима работы осуществляется в регистре CONTROL установкой бита RTMODE и сбросом BCMODE.

Монитор

В этом режиме осуществляется прослушивание магистрали и отбор необходимой информации для проведения: технического обслуживания, регистрации эксплуатационных параметров, анализа решаемых задач или обеспечения информацией резервного КШ. Монитор пассивно прослушивает выбранную шину и захватывает весь трафик на шине, но никогда не передаёт информацию на шину. Принятое из магистрали командное слово помещается в регистр CommandWord1, а для сообщений формата 3 и 8 принятое второе командное слово помещается в регистр CommandWord2. Ответное слово ОУ принятое из магистрали помещается в регистр StatusWord1. А для сообщений формата 3 и 8 помимо этого применяется регистр StatusWord2. Для приёма слов данных команд управления, форматы сообщений 5, 6 и 10, применяется регистр ModeData. Выбор этого режима работы осуществляется в регистре CONTROL установкой битов RTMODE и BCMODE. Для быстрой расшифровки сообщений можно применить регистр MSG. Каждому формату сообщения на магистрали соответствует определённый код в этом регистре.

### 15.6.2 *Форматы сообщений*

Сообщения, передаваемые по информационной магистрали, имеют формат, соответствующий форматам основных или групповых сообщений. Любые другие типы сообщений, не соответствующие ГОСТ Р 52070-2003 не поддерживаются.

Форматы основных сообщений, приведённые ниже, используются для передачи информации предназначенной одному ОУ и предусматривают выдачу ОС. В данном случае КС – командное слово, СД – слово данных, ОС – ответное слово. Времена t1 и t2 формируются аппаратно и не могут быть изменены программно. Пауза t2 между сообщениями, формируемая КШ, не менее 4 мкс. А пауза перед передачей ОС, формируемая ОУ, в пределах от 4 до 12 мкс. Если после ожидания 14 мкс так и не поступило ОС от ОУ, то фиксируется отсутствие ОС от ОУ и формируется соответствующий признак ошибки.

Время непрерывной передачи данных в линию не превышает 660 мкс, что соответствует командному слову и 32-м словам данных.

Формат 1 – передача данных от КШ к ОУ
Формат 2 – передача данных от ОУ к КШ
Формат 3 – передача данных от ОУ к ОУ
Формат 4 – передача КУ
Формат 5 – передача КУ и приём СД от ОУ
Формат 6 – передача КУ и СД оконечному устройству

Групповые сообщения, приведённые ниже, начинающиеся с передачи КШ групповой команды с кодом адреса 11111, используются для передачи информации одновременно нескольким ОУ без выдачи ими ОС.

Формат 7 – передача данных (в групповом сообщении) от КШ к оконечным устройствам

Формат 8 – передача данных (в групповом сообщении) от оконечного устройства к оконечным устройствам

Формат 9 – передача групповой команды управления

Формат 10 – передача групповой команды управления со словом данных

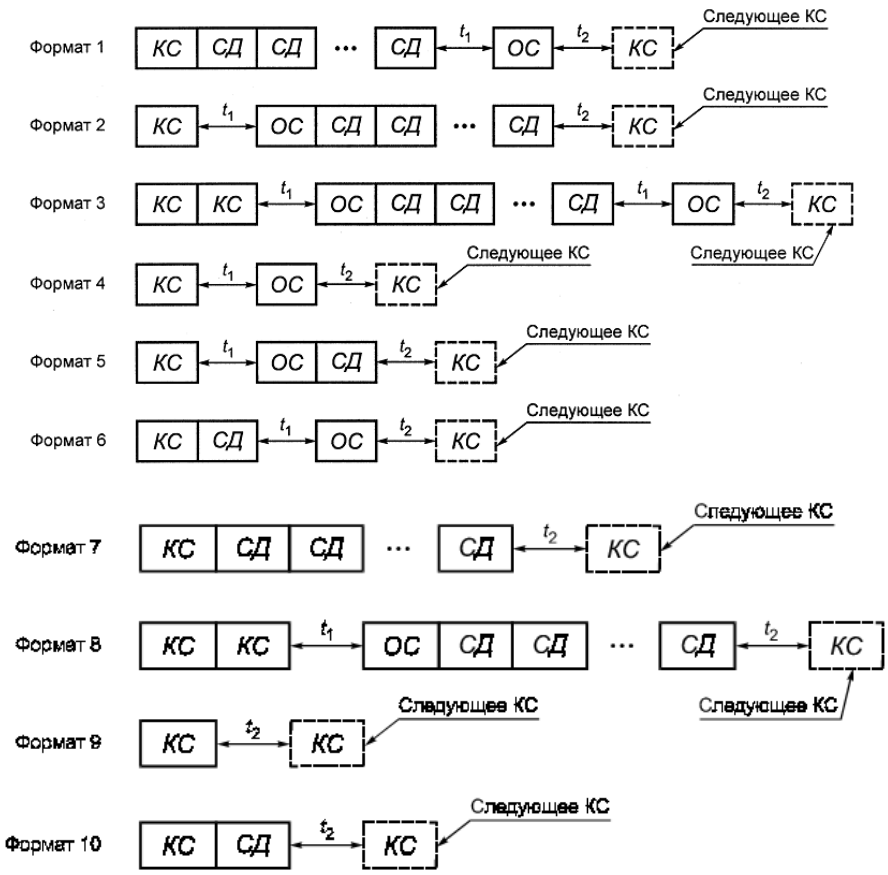



Рисунок 158 – Форматы сообщений МКПД

Если ОУ в формате сообщения ОУ-ОУ получило достоверное командное слово на приём информации, то первое СД должно быть им принято через паузу не более (57±3) мкс, в противном случае формируется соответствующий признак ошибки.

### 15.6.3 *Формат слов*

Каждое слово начинается с сигнала пословной синхронизации (с синхросигнала) и имеет 17 информационных разрядов, включая разряд контроля по чётности.

Командное слово содержит:

- синхросигнал;
- поле «Адрес ОУ»;
- разряд «Приём/Передача»;
- поле «Подадрес/Режим управления»;
- поле «Число СД/Код КУ»;
- разряд контроля по чётности.

Синхросигнал имеет длительность, составляющую три интервала времени передачи одного двоичного разряда. Полярность первой половины сигнала положительная, а вторая – отрицательная.

Адрес ОУ содержит код адреса из диапазона кодов 00000 – 11110, которому предназначено КС. КС с кодом адреса 11111 называется групповой командой, а сообщение, содержащее групповую команду – групповым.

Разряд «Приём/Передача» указывает на действие, которое должно выполнить ОУ (принимать или передавать СД). Логический нуль означает, что ОУ должно принимать СД, а логическая «1» – передавать СД.

Поле «Подадрес/Режим управления» содержит код подадреса ОУ или код признака режима управления 00000 или 11111.

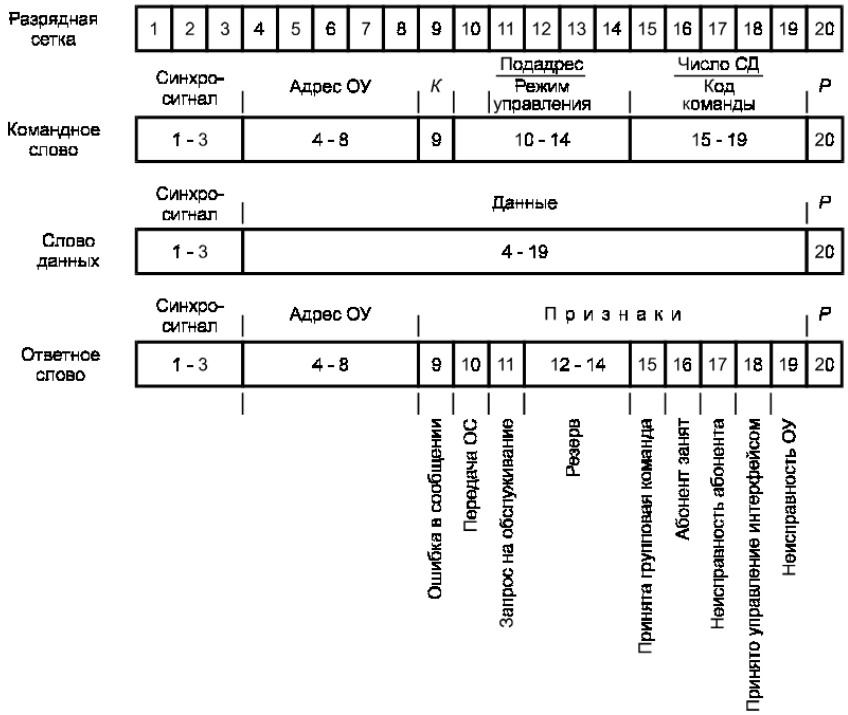



Рисунок 159 – Форматы слов

Поле «Число СД/Код КУ» содержит код числа слов данных, которые должны быть переданы или приняты ОУ в связи с приёмом адресованного ему КС, или код КУ. В одном сообщении может быть передано не более 32 СД. Числовое значение двоичных кодов, обозначающих число СД, соответствует их десятичным эквивалентам, за исключением кода 00000, который соответствует числу 32.

Разряд контроля по чётности используется для контроля по чётности предшествующих ему 16 разрядов КС. Разряд принимает такое значение, чтобы сумма значений всех 17 информационных разрядов слова (включая контрольный разряд) была нечётной.

Слово данных содержит:

- синхросигнал;
- данные;
- разряд контроля по чётности.

Синхросигнал имеет длительность, составляющую три интервала времени передачи одного двоичного разряда. Полярность первой половины сигнала отрицательная, а вторая положительная.

Поле данных содержит передаваемые данные, а разряд контроля по чётности формируется так же, как в командном слове.

Ответное слово содержит:

- синхросигнал;
- поле «Адрес ОУ»;
- поле разрядов признаков состояния: ошибка в сообщении, передача ОС, запрос на обслуживание, принята групповая команда, абонент занят, неисправность абонента, принято управление интерфейсом, неисправность ОУ;
- разряд контроля по четности.

Синхросигнал аналогичен синхросигналу КС. Поле «Адрес ОУ» содержит собственный адрес ОУ. Поле разрядов признаков состояния ОУ отображает текущее состояние ОУ. Разряд контроля по чётности формируется так же, как в командном слове.

### 15.6.4 *Структурная схема в режиме КШ*

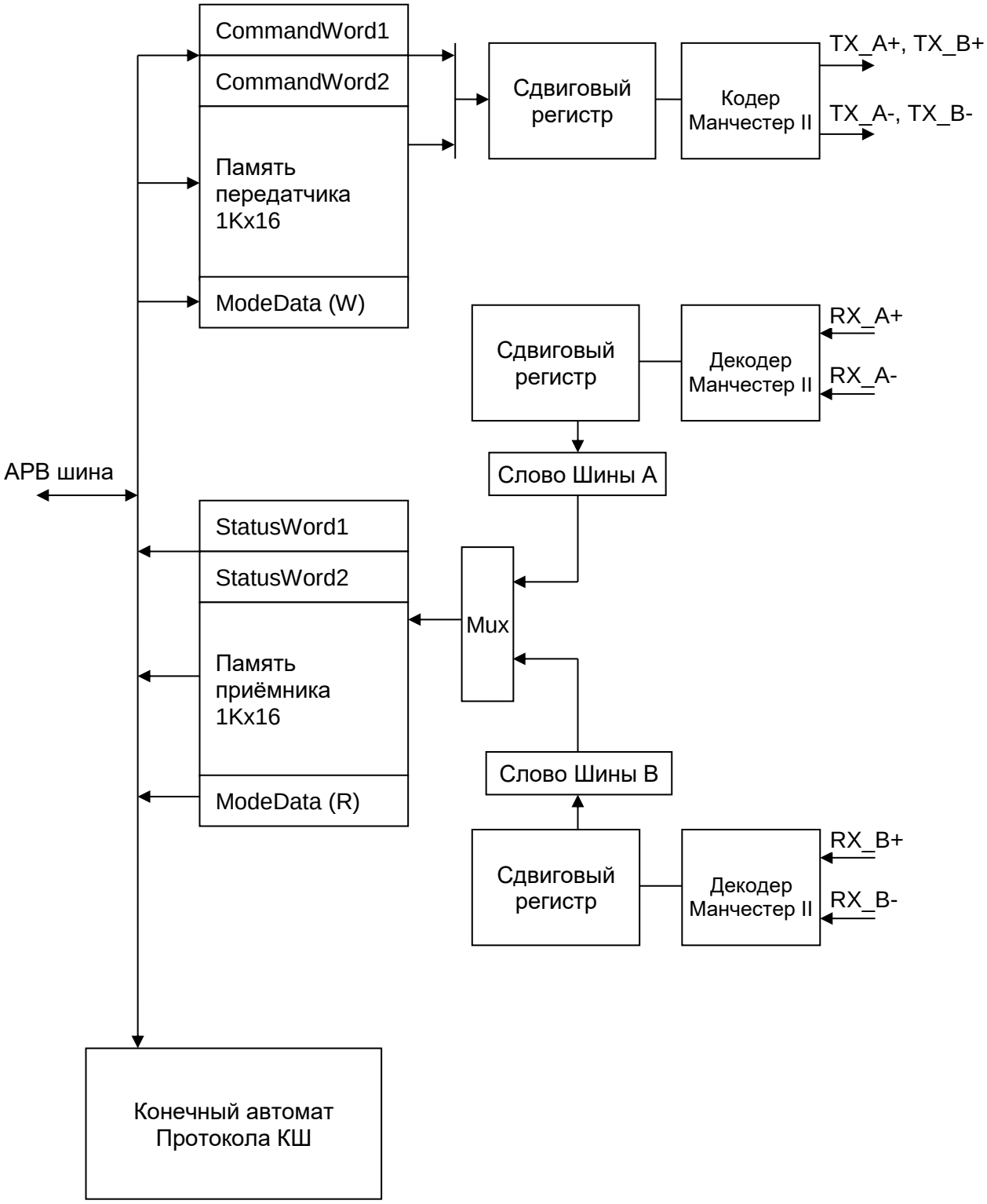



Рисунок 160 – Структурная схема в режиме КШ

### 15.6.5 *Структурная схема в режиме ОУ*

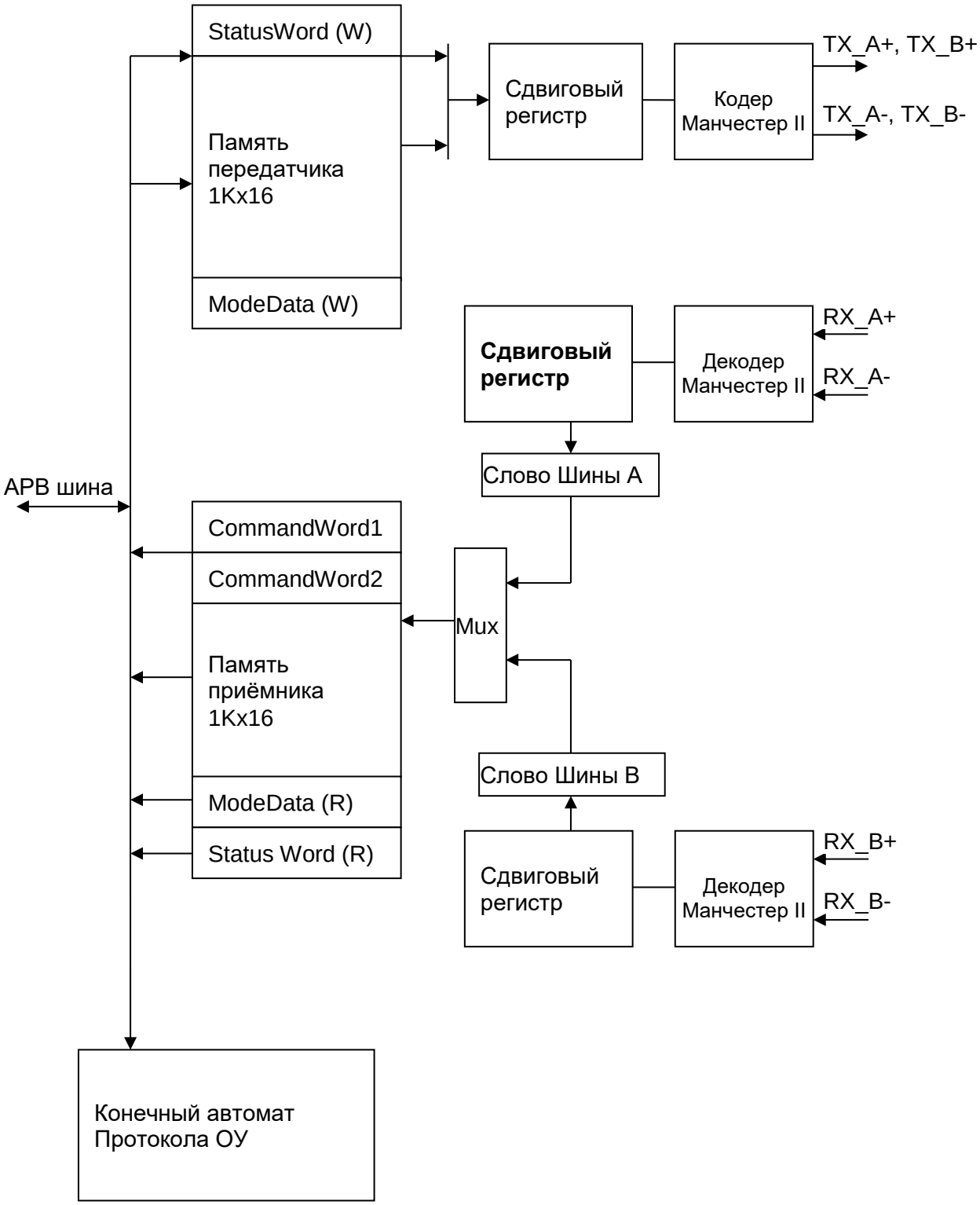



Рисунок 161 – Структурная схема в режиме ОУ

### 15.6.6 *Структурная схема в режиме М*

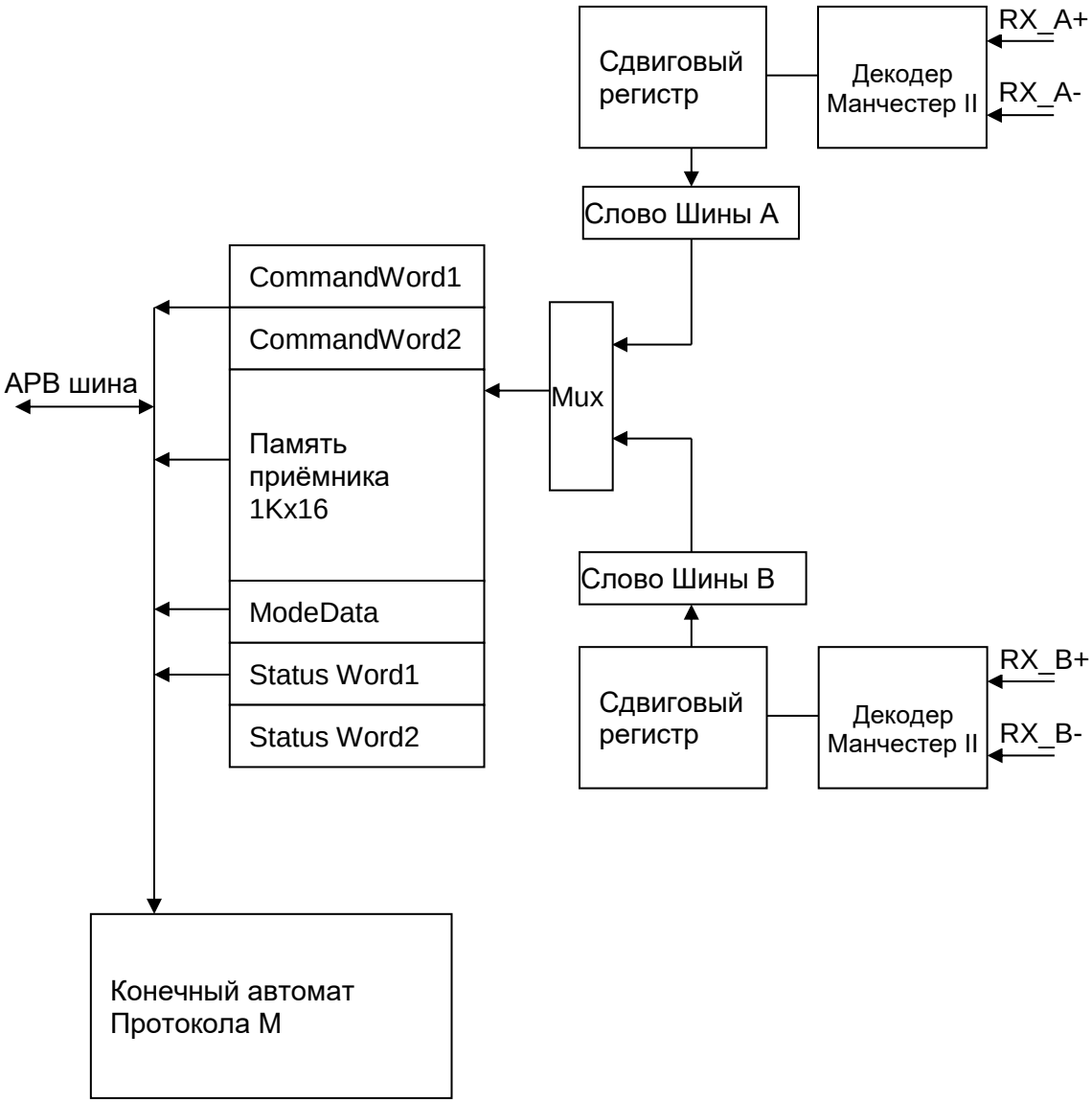



Рисунок 162 – Структурная схема в режиме М

### 15.6.7 *Инициализация*

Перед началом работы с контроллером в первую очередь необходимо инициализировать память передатчика и приемника во избежание ошибок по ECC. Далее необходимо сбросить контроллер, чтобы очистить все регистры сообщений. Это достигается установкой бита MR регистра CONTROL в логическую «1». Затем бит необходимо сбросить в «0». Далее нужно задать в регистре CONTROL значение делителя частоты DIV таким образом, чтобы при делении частоты ядра HCLK на это значение получить опорную частоту блока контроллера 1 МГц. После этого с помощью бит RTMODE и BCMODE выбирается соответствующий режим работы ОУ или КШ. Одновременная установка двух этих бит запрещена.

Для того чтобы выбрать какой канал будет использован для передачи данных основной или резервный, устанавливается соответствующий бит (TRA – основной канал, TRB – резервный канал). В режиме КШ командные слова будут передаваться только по тому каналу, который выбран. В режиме ОУ необходимо установить оба бита, так как ОУ

не может выбирать, по какому каналу ему передавать СД и ОС, и поэтому их передача происходит по тому каналу, по которому было принято КС.

Для режима ОУ в битах RTA4 – RTA0 регистра CONTROL задаётся адрес ОУ, который должен соответствовать адресу в поле «Адрес ОУ» командного слова, если идёт обращение к этому ОУ.

```
Пример инициализации ОУ
MILCNR1->CONTROL=0x00000001; //установка бита MR=1
MILCNR1->CONTROL=0x00014078;
//RTMODE=1, TRB=TRA=1, RTA=1, DIV=40
Пример инициализации КШ
MILCNR1->CONTROL = 0x00000001; //установка бита MR=1
MILCNR1->CONTROL = 0x00014014;
//BCMODE=1, TRA=1, TRB=0, RTA=0, DIV=40
```

В обоих случаях значения делителя частоты DIV = 40, что соответствует частоте ядра 40 МГц, и для получения опорной частоты контроллера необходимо 40 МГц/DIV = 1 МГц.

### 15.6.8 *Приём и передача в режиме ОУ*

Для того, чтобы настроить контроллер в режиме ОУ, необходимо выполнить все пункты, описанные в подразделе 15.6.7 «Инициализация». После этого необходимо задать ответное слово для КШ с помощью регистра StatusWord1. В режиме ОУ регистр по записи содержит предназначенное для передачи КШ ответное слово, а по чтению содержит ответное слово, полученное от передающего ОУ в транзакции ОУ-ОУ.

```
Пример записи ответного слова ОУ
MILCNR1->StatusWord1=0x00000800;
```

В данном случае в регистр заносятся только старшие 5 разрядов, соответствующие адресу ОУ. Остальные разряды признаки состояния ОУ можно оставить в нуле. Но в процессе работы может возникать необходимость изменять эти биты. Для этого необходимо программно устанавливать и сбрасывать эти биты, так как аппаратно они не изменяются.

Для того чтобы обеспечить формат сообщения 5, необходимо задавать слово данных, передаваемое КШ в команде управления. Для этих целей используется регистр ModeData.

```
Пример записи слова данных команды управления
MILCNR1->ModeData=0x000055AA;
```

После того как проведена инициализация, заданы ответное слово и слово данных команды управления, ОУ сразу готово к работе и может отвечать на все возможные форматы сообщений.

Так как в процессе работы ОУ в каждый момент времени требуется передача определённых СД и СД команды управления, то программно необходимо обновлять те области памяти, которые содержат эти данные. Без обновления при запросе данных КШ будут переданы последние записанные в эти области памяти данные. Поэтому при написании программы следует помнить и обновлять данные и слова данных команды управления.

Для хранения СД применяется адресное пространство 0x000-0xFFC (относительно базового адреса периферийного блока). Данные 16-ти разрядные, но обращение к ним должно быть выровнено по границе 32-х разрядного слова. То есть 2 младших разряда не участвуют в формировании адреса.

```
Пример инициализации данных для подадреса 1
addon=0x80;
addr_DATA=(unsigned int)&MILCNR1->DATA[0];
for(i=1;i<=32;i++)
{
   *((volatile unsigned int *)(addr_DATA+addon)) = i;
   addon=addon+4;
}
```

Из примера видно, что стартовый адрес памяти СД для подадреса 1 – 0x80, для последующих подадресов: n*0x80, где n-номер подадреса (n=1-31).

При приёме СД или слова данных команды управления, признаком обновления их значений является флаг VALMESS. После того как флаг установлен можно считать новые данные или слово данных команды управления. Но следует учитывать то, что этот флаг автоматически сбрасывается через 4 мкс, после его установки, поэтому желательно применять прерывания по установке сигнала VALMESS.

```
Пример чтения слова данных команды управления
i= MILCNR1->ModeData;
```

В переменную *i* будет прочитано значение слова данных команды управления.

```
Пример чтения данных для подадреса 1
addon=0x80;
addr_DATA=(unsigned int)&MILCNR1->DATA[0];
for(i=0;i<32;i++)
{
   mas[i]=*((volatile unsigned int *)(addr_DATA+addon));
   addon=addon+4;
}
```

Для упрощения декодирования команд управления КШ в режиме ОУ доступен регистр кодов MSG полученных сообщений. Каждому формату сообщения на магистрали соответствует определённый код в этом регистре. Чтение и последующая дешифрация этого кода упрощает процедуру декодирования сообщения и уменьшает время обработки сообщений. При использовании регистра экономится время на чтение двух командных регистров и разбор значений бит этих регистров.

### 15.6.9 *Приём и передача в режиме КШ*

В отличие от режима ОУ, в режиме КШ необходимо задавать не ответное слово, а командное слово или два командных слова в режиме работы ОУ-ОУ. Помимо этого, нужно инициировать процедуру приёма или передачи данных установкой бита BCSTART. После завершения транзакции на шине этот бит автоматически сбрасывается в ноль. Поэтому для инициирования новой транзакции нужно повторно устанавливать этот бит.

```
Пример записи командных слов и бита BCSTART
MILCNR1->CommandWord1=0x00000820;       //Командное слово 1
MILCNR1->CommandWord2=0x00000000;       //Командное слово 2
MILCNR1->CONTROL=0x00014016; //Регистр управления
```

Как видно из примера, в командном слове 1 задаётся код слов данных 00000, что соответствует 32 СД. Данные будут передаваться от контроллера шины оконечному устройству с адресом 1 из подадреса 1. Второе командное слово задаётся равным нулю и никак не влияет на транзакцию. В регистре управления устанавливается бит BCMODE, что соответствует режиму работы КШ, а также устанавливается бит BCSTART, что инициирует

начало транзакции, выбирается канал А для передачи (TRA=1), а также устанавливается делить частоты 40, что соответствует частоте работы ядра 40 МГц.

Для того чтобы инициировать приём в этом примере, необходимо только установить бит 10 равным единице в командном слове 1.

Если транзакция завершена успешно (признак VALMESS установился в единицу), то полученные СД или слово данных команды управления могут быть прочитаны. В противном случае устанавливается один из флагов ошибки. Сброс этих флагов осуществляется либо установкой битом MR, либо инициированием новой транзакции битом BCSTART.

В режиме работы ОУ-ОУ, форматы сообщений 3 и 8, КШ принимает из магистрали СД в подадрес, указанный во втором командном слове, регистр CommandWord2.

После приема ОС необходимо выполнить программный анализ бит признаков состояния ОУ. Аппаратно контроллер выполняет анализ бит "Абонент занят" и "Ошибка в сообщении". Если в принятом ОС установлен бит "Абонент занят" или бит "Ошибка в сообщении", то в регистре STATUS устанавливается бит ERR и выполняется переход в состояние обработки паузы t2. В формате 5 при приеме ОС с битом "Абонент занят", равным нулю, бит "Ошибка в сообщении" аппаратно не анализируется.

### 15.6.10 *Прерывания*

Для уменьшения потерь времени программы на опрос флагов контроллера, введено одно прерывание, генерируемое при установке любого из флагов контроллера. Прерывание может генерировать установка одного из четырёх флагов:

- флаг ошибки;
- флаг успешного завершения транзакции в канале;
- флаг приёма достоверного КС, ОС или слова данных команды управления;
- флаг неактивности контроллера.

Каждый из флагов может быть маскирован битами разрешения прерывания по какому-либо флагу.

Разрешенные в регистре INTEN сигналы флагов контроллера по фронту формируют независимые внутренние импульсные сигналы INT[3:0]. Сигналы INT[3:0] объединяются по схеме ИЛИ и формируют общий импульсный запрос прерывания, который реализует прерывание по фронту. Сброс прерывания выполняется автоматическии ввиду импульсного запроса прерывания.

## 15.7 Контроллер I2C (I2C_CNTR)

Интерфейс 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.

Контроллер интерфейса I2C работает только в режиме ведущего устройства («мастера»). Реализована совместимость с многомастерными системами: предусмотрены механизмы детектирования коллизий и арбитража шины, предотвращающие потерю данных при обмене, когда два или более мастера пытаются осуществить передачу одновременно. Поддерживается работа с ведомыми устройствами, применяющими растяжение тактовой частоты SCL (clock stretching) для замедления скорости обмена данными.

Контроллер интерфейса I2C позволяет работать на типовых скоростях:

- нормальная: 100 кбит/с;
- быстрая: 400 кбит/с;
- очень быстрая: 1 Мбит/с.

### 15.7.1 *Конфигурация контроллера I2C*

Контроллер I2C формирует тактовый сигнал обмена данными SCL с использованием внутреннего предделителя частоты DIV, коэффициент деления которого задается в регистрах PRL (младшая часть) и PRH (старшая часть). Минимальное значение DIV равно 1. Если рассчитанное значение DIV не использует старшую часть, то регистр PRH должен быть установлен в 0. Настройка DIV должна выполняться только при выключенном контроллере I2C (EN_I2C=0).

Рассчетная скорость обмена данными по интерфейсу I2C определяется по формуле:

$$F_{SCL} = \frac{HCLK}{5 \cdot (DIV + 1)}. \quad (14)$$

Необходимо учитывать, что контроллер I2C отслеживает фактическое состояние линии SCL. Если после завершения формирования низкого уровня контроллер отпускает линию SCL, но она остаётся в состоянии логического нуля (например, из-за растяжения SCL ведомым устройством или медленного фронта нарастания сигнала SCL), то начало следующего тактового цикла откладывается до перехода SCL в высокий уровень. Таким образом фактическая скорость обмена может оказаться ниже расчётной, определяемой формулой (14).

Разрешение работы контроллера I2C осуществляется установкой бита EN_I2C в регистре CMD. Выключение контроллера I2C (сброс бита EN_I2C) допустимо только после завершения обмена на шине, то есть после окончания передачи сигнала STOP. Нарушение данного требования может привести к блокировке шины I2C.

Запись в регистр управления CMD возможна только при разрешенной работе контроллера I2C (EN_I2C=1). Попытка записи в регистр CMD при неактивном состоянии контроллера I2C (EN_I2C=0) игнорируется.

### 15.7.2 *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»).

### 15.7.3 *Протокол I2C*

Типовая передача по интерфейсу I2C содержит 4 фазы:

- сигнал START;
- передача адреса;
- передача данных;
- сигнал STOP.

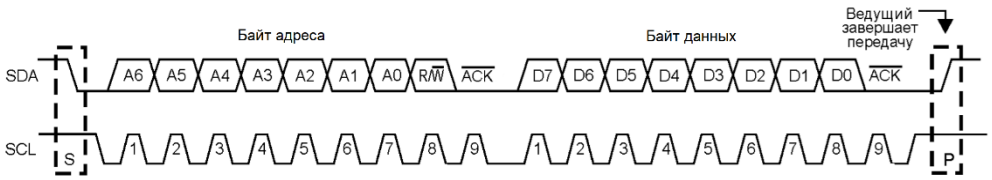



Рисунок 163 – Прием/передача по I2C в 7-битном режиме

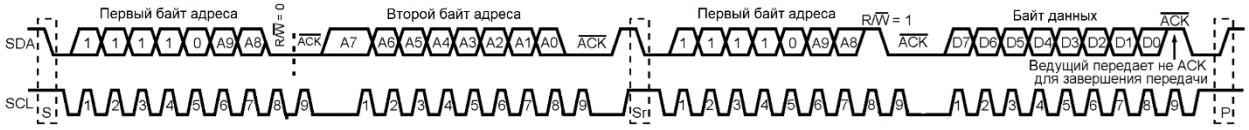



Рисунок 164 – Прием по I2C в 10-битном режиме

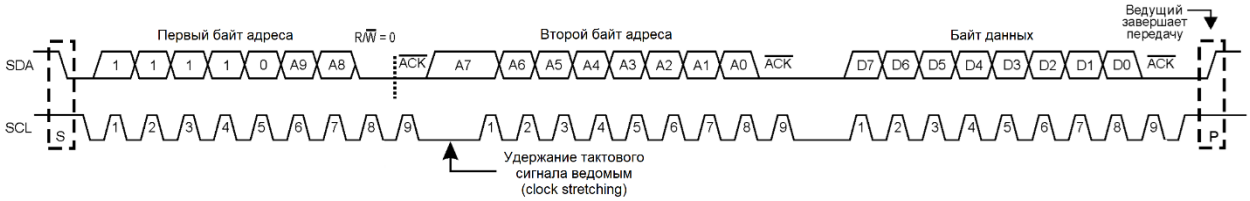



Рисунок 165 – Передача по I2C в 10-битном режиме

### 15.7.4 *Сигнал START*

Когда шина находится в свободном состоянии, то есть ни 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 CMD при установленных битах WR или RD. Для начала стандартного обмена необходимо установить биты START и WR в регистре CMD, что приведет к последовательной выдаче на линию сигнала START, а также адреса ведомого устройства и бита RW, которые записаны в регистр TXD. В зависимости от состояния линии SCL генерируется либо сигнал START, либо повторный сигнал START.

### 15.7.5 *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: 0 – запись данных в ведомое устройство, 1 – чтение данных из ведомого устройства. 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 тактовый импульс SCL.

Процесс выдачи 7-битного адреса выполняется совместно с выдачей сигнала START. Необходимо записать адрес ведомого и бит RW в регистр TXD, после чего установить биты START и WR в регистре CMD. Контроллер осуществит передачу сигнала START и адреса в линию.

Контроллер также поддерживает передачу 10-битного адреса ведомого путем генерации нескольких циклов передачи адреса (см. рисунки 164 и 165). Сначала необходимо выполнить передачу сигнала START и первого байта адреса ведомого с битом RW равным 0 (запись в ведомое устройство), как указано выше. После завершения передачи необходимо записать второй байт адреса ведомого в регистр TXD, после чего установить бит WR в регистре CMD. Если требуется смена направления передачи на RW = 1 (чтение из ведомого устройства), то после завершения передачи второго байта адреса необходимо выполнить повторную передачу сигнала START и первого байта адреса ведомого с битом RW равным 1.

### 15.7.6 *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 тактовый импульс SCL. 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 нужно записать данные в регистр TXD и установить бит WR в регистре CMD. Для чтения данных из ведомого устройства нужно в регистре CMD установить бит RD, при этом одновременно записав нужное значение в бит ACK. На время выполнения передачи контроллер интерфейса выставляет флаг TR_PROG в регистре STA. Когда передача завершена, этот флаг снимается и устанавливается флаг INT. Если при этом установлен бит разрешения EN_INT в регистре CTR, то генерируется прерывание контроллеру прерываний. Регистр RXD содержит корректные принятые данные после сброса флага TR_PROG или установки флага INT в регистре STA. Новый цикл чтения или записи можно начать только тогда, когда флаг TR_PROG сброшен.

### 15.7.7 *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.

Сигнал STOP генерируется при записи единицы в бит STOP регистра CMD.

## 15.8 Контроллер USB (USB_CNTR)

USB-контроллер канального уровня соответствует стандарту USB 2.0 и поддерживает следующий функционал:

- Контроллер периферийного устройства с поддержкой режимов LS (Low Speed), FS (Full Speed), HS (High Speed);
- Host-контролер (хост) с поддержкой работы с несколькими устройствами (при использовании хаба). Режим работы HOST HS (High Speed) не поддерживается блоком USB;
- Поддержка протокола согласования Host Negotiation Protocol стандарта OTG.

В USB-контроллере реализованы три конечных точки в дополнение к точке 0, которые можно настроить независимо на прием и передачу. Для конечной точки 0 используется FIFO размером 64 байт для буферизации одного пакета. Для конечных точек 1, 2 и 3 используются независимые FIFO размером 1024 байт. При приеме и передаче используется одно и тоже FIFO.

### 15.8.1 *Поддержка нескольких устройств*

#### 15.8.1.1 Распределение устройств между конечными точками

Отдельные функции подключенных устройств назначаются конечным точкам в ядре USB-контроллера через группу из трех регистров, которые связаны с каждой конечной точкой Rx или Tx, реализованной в ядре (включая конечную точку 0).

Это регистры Tx/RxFuncAddr, Tx/RxHubAddr и Tx/RxHubPort. (Расположение этих регистров зависит от того, какая из конечных точек контроллера настраивается).

Информацией, которая должна быть записана в регистр Tx/RxFuncAddr, является адрес целевой функции, к которой необходимо получить доступ через выбранную конечную точку.

Регистры Tx/RxHubAddr и Tx/RxHubPort предусмотрены для случая, когда full-speed или low-speed устройство подключено к контроллеру через high-speed USB 2.0 хаб, который выполняет требуемую трансляцию транзакций между high-speed передачей и low-/full-speed передачей. В этом случае регистры Tx/RxHubAddr и Tx/RxHubPort должны хранить адрес хаба, который выполняет трансляцию транзакции, и порт хаба, через который конечная точка должна получить доступ к устройству.

Регистр Tx/RxHubAddr также используется для записи того, предлагает ли хаб несколько транзакционных трансляторов или только один. Это существенно влияет на общую достижимую пропускную способность.

В дополнение к записи адреса целевой функции через эти три регистра, необходимо записать номер конечной точки, скорость работы целевого устройства и тип трансляции, которая будет выполнена.

Для конечной точки Tx эту информацию необходимо установить в регистре TxType, когда регистр Index установлен на нужную конечную точку. Для конечной точки Rx эту информацию необходимо установить в регистре RxType, когда регистр Index установлен на нужную конечную точку. В обоих случаях номер конечной точки записывается в битах D3 - D0, тип транзакции выбирается через биты D5 - D4, а рабочая скорость выбирается через биты D7 - D6.

В случае конечной точки 0 необходимо установить только скорость (эта конечная точка имеет средства для обработки только транзакций управления и, следовательно, всегда связана с конечной точкой 0 этого устройства). Эта установка скорости выполняется через биты D7 - D6 регистра Type 0, когда в регистре Index установлен 0.

#### 15.8.1.2 Описание работы

После того как распределение функций по конечным точкам было сделано и скорость работы целевого устройства установлена, большинство операций в многоточечной настройке ничем не отличаются от операций для эквивалентных действий, в которых ядро подключено только к одному устройство.

Однако необходимы дополнительные действия в следующих ситуациях:

- используется опция динамического распределения функций по конечным точкам (например, чтобы поддерживать более широкий диапазон устройств)
- управляющие пакеты, обычно ассоциируемые с конечной точкой 0, обрабатываются через другую конечную точку.

*Если используется динамическое распределение*, пользователю необходимо отслеживать текущее состояние Data Toggle конечной точкой и каждого из устройств, связанного с этой конечной точкой. Это необходимо, чтобы пользователь смог выбрать правильное состояние Data Toggle при переключении между устройствами.

Состояние Data Toggle может быть изменено при помощи изменения соответствующего регистра TxCSR/RxCSR и установкой туда бит Data Toggle Write Enable и Data Toggle, которые доступны, пока ядро находится в режиме хоста.

*Если управляющие пакеты обрабатываются через конечную точку, отличную от конечной точки 0,* пользователь должен позаботиться о том, чтобы каждый Setup токен был отправлен. Это включает в себя настройку бита SetupPkt из регистра TxCSR и бита TkPktRdy, когда ядро работает в режиме хоста. Если бит SetupPkt не установлен, будет отправлен токен OUT.

Использование другой конечной точки для этой функции возможно, как описано выше, но есть еще одно замечание:

(I) управляющая функция должна быть назначена только на пару конечных точек Rx/Tx (то есть с тем же номером конечной точки).

(II) выбранные конечные точки должны быть связаны с FIFO, который сможет вместить размер пакета транзакции EP0, на выбранной рабочей скорости (то есть минимум 8 байт для транзакций low-/full-speed, или 64 байта для транзакций high-speed).

### 15.8.2 *Схема программирования*

В этом и следующих разделах рассмотрены действия, которые необходимо выполнить устройству, управляющему ядром USB-контроллера, и некоторые аспекты работы ядра, которые влияют на это.

#### 15.8.2.1 Обработка USB-прерываний

Когда USB прерывает ЦП, ему необходимо прочитать регистр статуса прерывания, чтобы определить, какие конечные точки вызвали прерывание и перейти к соответствующей процедуре обработки. Если несколько конечных точек вызвали прерывание, сначала необходимо обработать конечную точку 0, а затем остальные конечные точки. Прерывание приостановки должно обрабатываться последним.

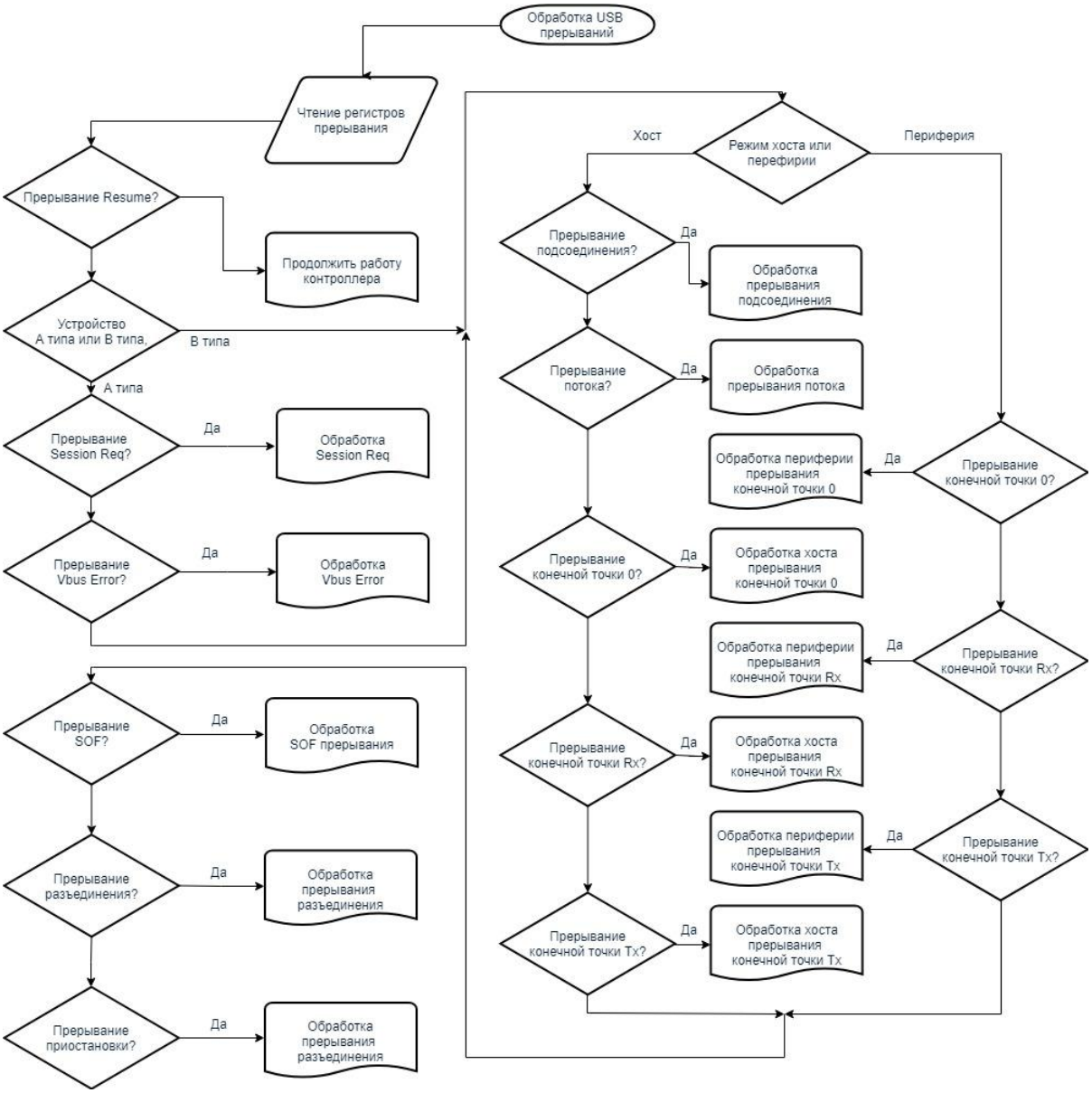



Рисунок 166 – Блок-схема процедуры обслуживания прерываний USB

#### 15.8.2.2 Приостановка/возобновление

Как USB-контроллер входит и выходит из режима Suspend (приостановки), зависит от того, работает ли он в настоящее время как хост или как периферия.

***Когда контроллер работает как хост***

*(I) Вход в режим Suspend*. При работе в качестве хоста контроллер может перейти в режим Suspend, установив бит SuspendMode в регистр Power. После установки этого бита, контроллер завершит текущую транзакцию, а затем остановит планировщик транзакций и счетчик кадров. Дальнейшие транзакции не будут происходить, и SOF-пакеты не будут создаваться.

Если бит Enable SuspendM (Power.D0) установлен, UTMI + PHY перейдет в режим с низким энергопотреблением, если контроллер перейдет в режим Suspend и остановит XCLK.

*(II) Передача сигналов Resume.* Когда программа требует, чтобы контроллер покинул режим Suspend, ему необходимо очистить бит Suspend в регистре Power, установить бит возобновления и оставить его на 20 мс. Пока этот бит установлен, контроллер будет генерировать сигналы Resume на шине. Через 20 мс ЦП должен

сбросить бит Resume, после чего будет запущен счетчик кадров и планировщик транзакций.

*(III) Ответ на удаленную активацию.* Если контроллер в режиме Suspend получил сигнал Resume, UTMI + PHY будет выведен из режима низкого энергопотребления и перезапустит XCLK. Контроллер выйдет из режима Suspend и автоматически установит бит Resume в регистре питания (D2), чтобы взять на себя генерацию сигнала Resume. Если прерывание Resume разрешено, то оно будет сгенерировано.

***Когда контроллер работает как периферийное устройство***

(I*) Вход в режим Suspend*. При работе в качестве периферийного устройства контроллер анализирует активность на USB-устройстве, и, если в течение 3 мс не было активности, он переходит в режим Suspend. Если прерывание Suspend разрешено, то в этот момент оно будет оправлено. Уровень выхода SUSPENDM также будет низким (если включен).

На этом этапе контроллер остается активным (и, следовательно, способен обнаружить, когда на USB-накопителе появляется сигнал Resume), или программа может отключить контроллер, остановив тактировщик. Однако во втором случае контроллер не сможет обнаружить сигнал Resume на USB, а значит и перезапустить тактировщик.

*(II) Когда на шине возникает сигнал Resume*, сначала необходимо перезапустить тактировщик. Затем контроллер автоматически выйдет из режима Suspend. Если прерывание Resume доступно, то оно будет создано.

*(III) Инициирование удаленной активации*. Если программа хочет инициировать удаленную активацию, в то время как контроллер находится в режиме Suspend, он должен в регистре Power установить бит возобновления (D2). (*Примечание – Если источник тактирования контроллера был остановлен, его необходимо перезапустить до того, как запись бита произойдет*). Программа должна оставить этот бит установленным в течение приблизительно 10 мс (минимум 2 мс, максимум 15 мс) перед тем, как сбросить. К этому времени хаб должен будет получить сигнал Resume на USB.

*Примечание* – когда программа инициирует удаленную активацию, прерывание Resume не генерируется.

#### 15.8.2.3 Перезагрузка USB

Когда USB-контроллер работает как периферийное устройство, и на USB-устройстве обнаружено состояние сброса, устройство выполнит следующие действия:

- Сбросит FAddr.
- Сбросит Index.
- Очистит все FIFO конечных точек.
- Сбросит все регистры Control/Status.
- Включит все прерывания конечных точек.
- Сгенерирует прерывание Reset.

Если бит HS Enab в регистре Power (D5) был установлен, контроллер также попытается согласоваться для работы в режиме high-speed. Получится ли согласоваться для работы в high-speed режиме отобразится в бите HS Mode (Power.D4).

Когда ПО ***в режиме периферийного устройства*** ПО, управляющее контроллером, при получении прерывание Reset, должно закрыть все открытые каналы и ждать начала перечисления шины.

***В хост режиме*** если бит Reset установлен в регистре Power, когда контроллер находится в режиме хоста, контроллер сгенерирует сигнал Reset на шине.

ЦП должен поддерживать бит сброса установленным не менее 20 мс, чтобы обеспечить правильный сброс целевого устройства.

После того как ЦП сбросит бит, контроллер запустит счетчик кадров и планировщик транзакций.

### 15.8.3 *Контроль транзакций (при помощи конечной токи 0)*

#### 15.8.3.1 Контроль транзакций в качестве периферийного устройства

***Запрос нулевых данных***

Запросы нулевых данных содержат всю свою информацию в 8-байтной команде и не требуют передачи дополнительных данных. Примеры стандартных запросов «Нулевые данные»: SET_FEATURE, CLEAR_FEATURE, SET_ADDRESS, SET_CONFIGURATION, SET_INTERFACE.

Последовательность событий начнется, как и при всех запросах, когда программное обеспечение получит прерывание конечной точки 0. Также будет установлен бит RxPktRdy (CSR0.D0). Затем команда из 8-ми байт должна прочитаться из FIFO конечной точки 0, декодироваться и далее, соответствующие ей действия должны быть приняты.

Затем биты ServicedRxPktRdy (D6) (указывающий, что команда была прочитана из FIFO) и DataEnd (D3) (указывающий, что для этого запроса не ожидается никаких дополнительных данных) установятся в регистр CSR0.

Когда хост переходит на этап запроса, будет отправлено второе прерывание конечной точки 0, указывающее, что запрос завершился. Никаких дополнительных действий от программного обеспечения не требуется: второе прерывание – это просто подтверждение успешного завершения запроса.

Если команда не опознана или по какой-либо другой причине не может быть выполнена, тогда, после декодировки, установятся биты ServicedRxPktRdy (D6) и SendStall (D5) в регистр CSR0. Когда хост переходит на этап запроса, контроллер отправит STALL, чтобы сообщить хосту, что запрос не был выполнен. Будет создано второе прерывание конечной точки 0 и установится бит SentStall (CSR0.D2).

Если хост продолжает отправляет данные после того, как бит DataEnd был установлен, контроллер отправит STALL. Прерывание конечной точки 0 будет сгенерировано и бит SentStall (CSR0.D2) будет установлен.

***Запрос на запись***

Запросы на запись включают дополнительный пакет (или пакеты) данных, отправляемых с хоста после 8-байтовой команды. Примером стандартного запроса на запись является: SET_DESCRIPTOR.

Последовательность событий начнется, как и при всех запросах, когда программное обеспечение получит прерывание конечной точки 0. Также будет установлен бит RxPktRdy (CSR0.D0). Затем 8-байтная команда должна прочитаться из FIFO конечной точки 0 и декодироваться.

Как и при нулевом запросе данных, бит ServicedRxPktRdy (D6) (указывающий, что команда была прочитана из FIFO) должен быть установлен в регистре CSR0, но бит DataEnd (D3) не должен быть установлен (указывая что ожидается больше данных).

Когда получено второе прерывание конечной точки 0, регистр CSR0 должен быть прочитан для проверки состояния конечной точки. Бит RxPktRdy (CSR0.D0) должен быть установлен, чтобы указать, что пакет данных получен. Затем регистр COUNT0 должен быть прочитан для определения размера этого пакета данных. Затем пакет данных можно читать из FIFO конечной точки 0.

Если длина данных, связанных с запросом (определяемая полем wLength в команде), больше, чем максимальный размер пакета для конечной точки 0, будут отправлены еще пакеты данных. В этом случае бит ServicedRxPktRdy должен быть установлен в регистр CSR0, но бит DataEnd должен быть сброшен.

Когда все ожидаемые пакеты данных получены, биты ServicedRxPktRdy и DataEnd (указывающий, что больше данных не ожидается) должны быть установлены в регистре CSR0.

Когда хост переходит на этап запроса, будет создано другое прерывание конечной точки 0, указывающее, что запрос завершен. Никаких дальнейших действий не требуется от программного обеспечения, прерывание является просто подтверждением успешного завершения запроса.

Если команда не распознана или по какой-либо другой причине не может быть выполнена, то после декодирования, биты ServicedRxPktRdy (D6) и SendStall (D5) должны будут установиться в регистр CSR0. Если хост продолжит отправлять данные, то контроллер отправит STALL, чтобы сообщить хосту, что запрос не был выполнен. Будет отправлено прерывание конечной точки 0 и установится бит SentStall (CSR0.D2).

***Запрос на чтение***

Запросы на чтение имеют пакет (или пакеты) данных, отправленный из функции на хост после 8-байтной команды. Примеры стандартных запросов чтение: GET_CONFIGURATION, GET_INTERFACE, GET_DESCRIPTOR, GET_STATUS, SYNCH_FRAME.

Последовательность событий начнется, как и во всех запросах, когда программное обеспечение получит прерывание конечной точки 0. Также будет установлен бит RxPktRdy (CSR0.D0). Затем 8-байтная команда должна прочитаться из FIFO конечной точки 0 и декодироваться. Затем бит ServicedRxPktRdy (указывающий, что команда прочитала FIFO) должен быть установлен в регистре CSR0.

Затем данные, которые должны быть отправлены хосту, должны быть записаны в FIFO конечной точки 0. Если отправляемые данные больше максимально допустимого размера пакета для конечной точки 0, в FIFO должна быть записана только часть пакета максимально допустимого размера. Затем бит TxPktRdy (D1) (указывающий, что в FIFO будет отправлен пакет) должен быть установлен в регистр CSR0. Когда пакет будет отправлен на хост, сгенерируется еще одно прерывание конечной точки 0, и следующий пакет данных может быть загружен в FIFO.

Когда последний пакет данных будет записан в FIFO, биты TxPktRdy и DataEnd (D3) (указывающий, что после этого пакета больше нет данных) должны быть установлены в регистре CSR0.

Когда хост переходит на этап запроса, будет создано еще одно прерывание конечной точки 0, чтобы указать, что запрос завершился. Никаких дальнейших действий от программного обеспечения не требуется: прерывание – это просто подтверждение того, что запрос успешно завершен.

Если команда не распознана или по какой-либо другой причине не может быть выполнена, после декодировки, должны быть установлены биты ServicedRxPktRdy (D6) и SendStall (D5) в регистре CSR0. Если хост запросит данные, контроллер отправит STALL, чтобы сообщить хосту, что запрос не был обработан. Будет сгенерировано прерывание конечной точки 0 и будет установлен бит SentStall (CSR0.D2).

Если хост запрашивает данные после установки DataEnd (D3), контроллер отправит STALL. Будет сгенерировано прерывание конечной точки 0 и будет установлен бит SentStall (CSR0.D2).

***Состояния конечной точки 0***

Когда USB-контроллер работает как периферийное устройство, контроль конечной точки 0 может происходить в трех состояниях: IDLE, TX и RX

Режим IDLE устанавливается по умолчанию при включении или сбросе.

RxPktRdy (CSR0.D0) установливается, когда конечная точка 0 находится в состоянии IDLE, указывая на новый запрос устройства. После того, как запрос устройства выгружается из FIFO, контроллер декодирует дескриптор, чтобы проверить наступление фазы передачи данных и, если она наступила, направление передачи данных передачи (чтобы установить направление FIFO).

В зависимости от направления передачи данных конечная точка 0 переходит в состояние TX или RX. Если фаза данных отсутствует, конечная точка 0 остается в состоянии IDLE, чтобы принять следующий запрос устройства.

Действия, которые процессор должен выполнять на разных этапах возможных передач (например, загрузка FIFO, настройка TxPktRdy), указаны на диаграмме.

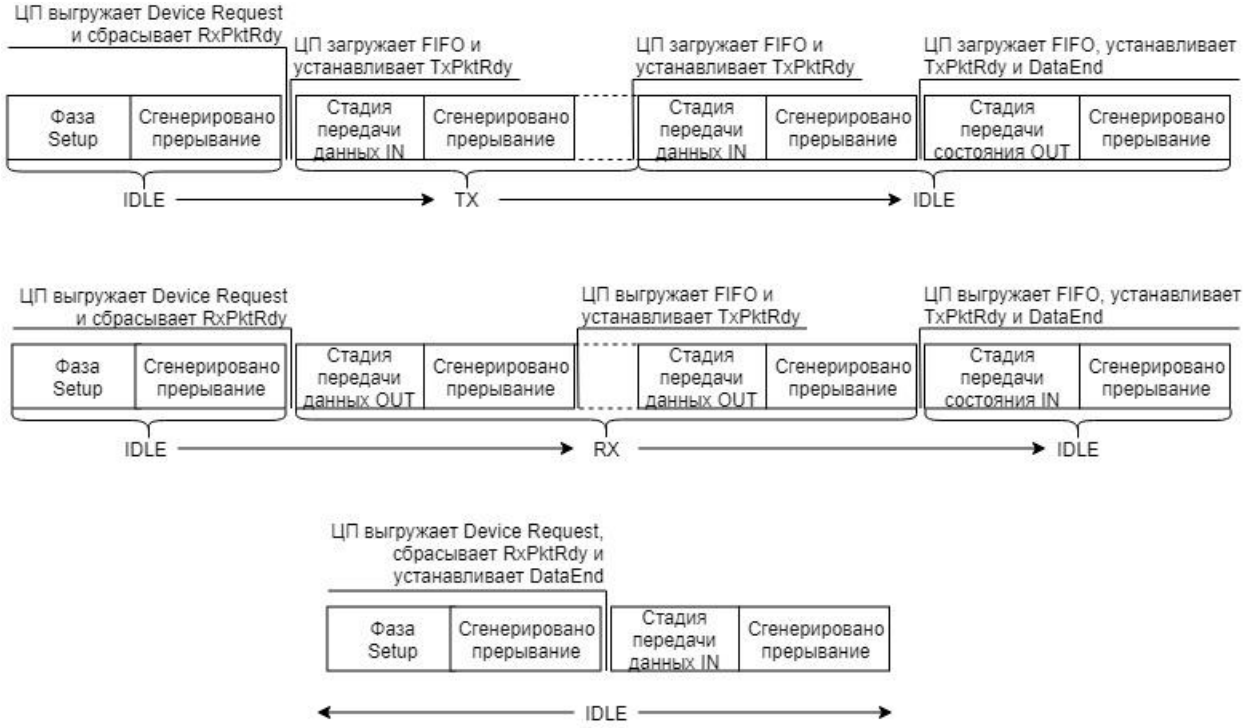



Рисунок 167 – Диаграмма передачи данных

***Процедура обслуживания конечной точки 0 как периферийного устройства***

Прерывание конечной точки 0 генерируется в следующих случаях:

- ядро устанавливает бит RxPktRdy (CSR0.D0) после получения действительного токена и записи данных в FIFO;
- ядро очищает бит TxPktRdy (CSR0.D1) после того, как пакет данных в FIFO был успешно передан хосту;
- ядро устанавливает бит SentStall (CSR0.D2) после завершения транзакции управления из-за нарушения протокола;
- ядро устанавливает бит SetupEnd (CSR0.D4), потому что передача управления завершилась до того, как установлен DataEnd (CSR0.D3).

Каждый раз, когда инициализируется сервис контрольной точки 0, прошивка должна проверить, была ли текущая передача управления завершена из-за состояния STALL или из-за преждевременного завершения передачи управления. Если передача управления заканчивается из-за условия STALL, бит SentStall должен быть установлен. Если передача управления заканчивается из-за преждевременного окончания передачи управления, бит SetupEnd должен быть установлен. В любом случае прошивка должна прервать обработку текущей передачи управления и установить состояние IDLE.

После того, как прошивка определила, что прерывание не было создано неразрешенным состоянием шины, следующее действие будет зависеть от состояния конечной точки.

*Если конечная точка 0 находится в состоянии «IDLE»*, единственной допустимой причиной, из-за которой может быть отправлено прерывание, является результат получения данных ядра с шины USB. Сервис должна проверить это, проверив бит RxPktRdy (CSR0.D0). Если бит установлен, то ядро получило пакет данных SETUP. Он должен быть выгружен из FIFO и декодирован для определения дальнейшего действия. В зависимости от этого действия, конечная точка 0 примет одно из трех следующих состояний:

- если команда представляет собой однопакетную транзакцию (SET_ADDRESS, SET_INTERFACE и т.д.) без фазы данных, конечная точка останется в состоянии IDLE;

- если команда имеет фазу передачи данных OUT (SET_DESCRIPTOR и т.д.), конечная точка перейдет в состояние Rx;
- если команда имеет фазу передачи данных IN (GET_DESCRIPTOR и т.д.), конечная точка перейдет в состояние TX.

*Если конечная точка находится в состоянии Tx*, прерывание укажет на то, что ядро получило токен IN, и данные из FIFO были отправлены. Прошивка должна ответить на это либо путем размещения большего количества данных в FIFO, если хост все еще ожидает данные, либо путем установки бита DataEnd, чтобы указать, что фаза передачи данных завершена. Как только фаза передачи данных транзакции будет завершена, конечная точка 0 будет возвращена в состояние IDLE для ожидания следующей транзакции управления.

*Если конечная точка находится в состоянии Rx*, прерывание укажет на то, что пакет данных получен. Прошивка должна ответить выгрузкой полученных данные из FIFO. Затем прошивка должна будет определить, получил ли он все ожидаемые данные. Если это так, прошивка должна установить бит DataEnd и вернуть конечную точку 0 в состояние IDLE. Если ожидается получение большего количества данных, прошивка должна установить бит ServicedRxPktRdy (CSR0.D6), чтобы указать, что она прочитал данные из FIFO и оставить конечную точку в состоянии RX.

**Режим IDLE**

Режим IDLE – это режим, который должен выбрать контроллер конечной точкой 0 при включении или перезагрузке и является режимом, в который контроллер конечной точки 0 должен возвращаться при завершении режимов RX и TX.

Это также режим, в котором обрабатывается фаза передачи SETUP (как показано на рисунке 168).

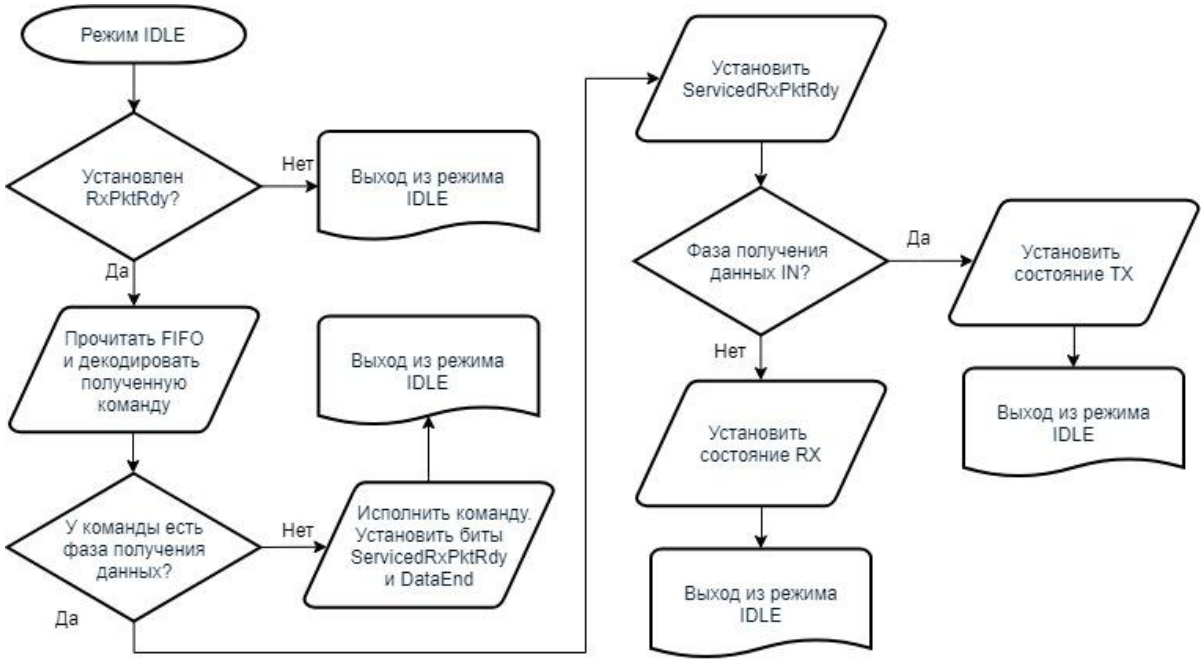



Рисунок 168 – Алгоритм режима IDLE

**Режим TX**

Когда конечная точка находится в состоянии TX, все входящие токены должны обрабатываются как часть фазы передачи данных до тех пор, пока требуемый объем данных не будет отправлен на хост. Если принимается либо токен SETUP, либо токен OUT

и конечная точка находится в состоянии TX, то это вызовет условие SetupEnd, поскольку ядро ожидает только токены IN.

Три события могут привести к переключению режима TX до того, как ожидаемый объем данных будет отправлен:

- хост отправляет недействительный токен, вызывающий условие SetupEnd (установит CSR0.D4);
- прошивка отправляет пакет, содержащий данных меньше максимального размера (MaxP);
- прошивка отправляет пустой пакет данных.

Пока транзакция не будет завершена, прошивка должна загружать FIFO при получении прерывания, которое указывает на то, что пакет был отправлен из FIFO (Прерывание генерируется, когда TxPktRdy очищается).

Когда прошивка принудительно завершает передачу (путем отправки короткого или пустого пакета данных), должен быть установлен бит DataEnd (CSR0.D3), чтобы указать ядру, что фаза передачи данных завершена и, что затем ядро должно получить подтверждающий пакет.

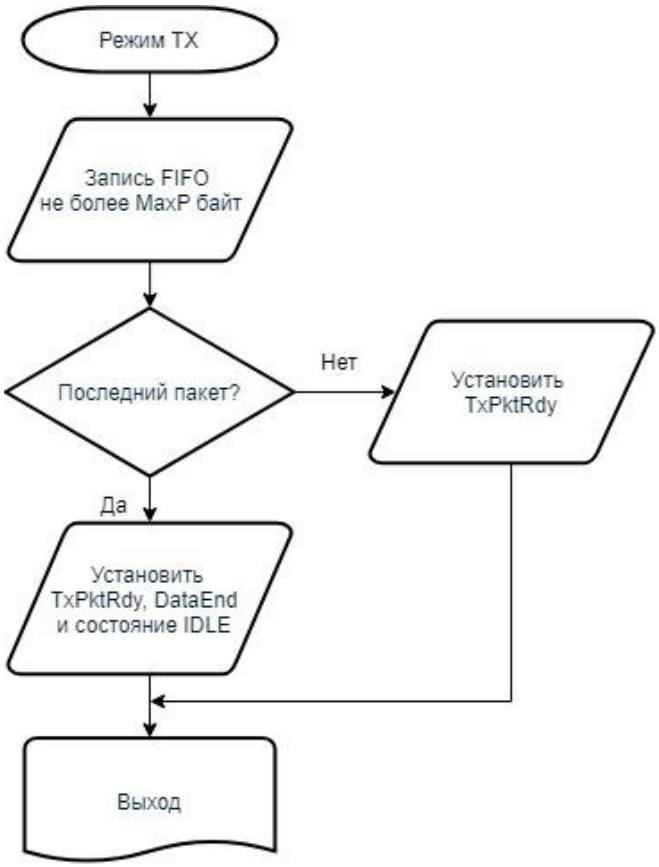



Рисунок 169 – Алгоритм режима TX

**Режим RX**

В режиме RX все поступающие данные должны рассматриваться как часть фазы передачи данных до тех пор, пока ожидаемый объем данных не будет получен. Если принимается либо SETUP, либо IN токен, и конечная точка находится в состоянии RX, то это приводит к возникновению условия SetupEnd, поскольку ядро ожидает только токены OUT.

Три события могут привести к прекращению режима RX до того, как ожидаемый объем данных будет получен:

- хост отправляет недействительный токен, вызывающий условие SetupEnd (установит CSR0.D4);
- хост отправляет пакет, содержащий данных меньше максимального размера пакета для конечной точки 0;
- хост отправляет пустой пакет данных.

Пока транзакция не будет завершена, прошивка должна выгружать FIFO при получении прерывания, которое указывает на то, что пришли новые данные (установлен RxPktRdy (CSR0.D0)) и сбрасывать RxPktRdy, устанавливая бит ServicedRxPktRdy (CSR0.D6).

Когда прошивка обнаруживает завершение передачи (путем получения либо ожидаемого количества данных, либо пустого пакета данных), он должен установить бит DataEnd (CSR0.D3), чтобы указать ядру, что фаза передачи данных завершена, и что далее ядро должно получить пакет подтверждения.

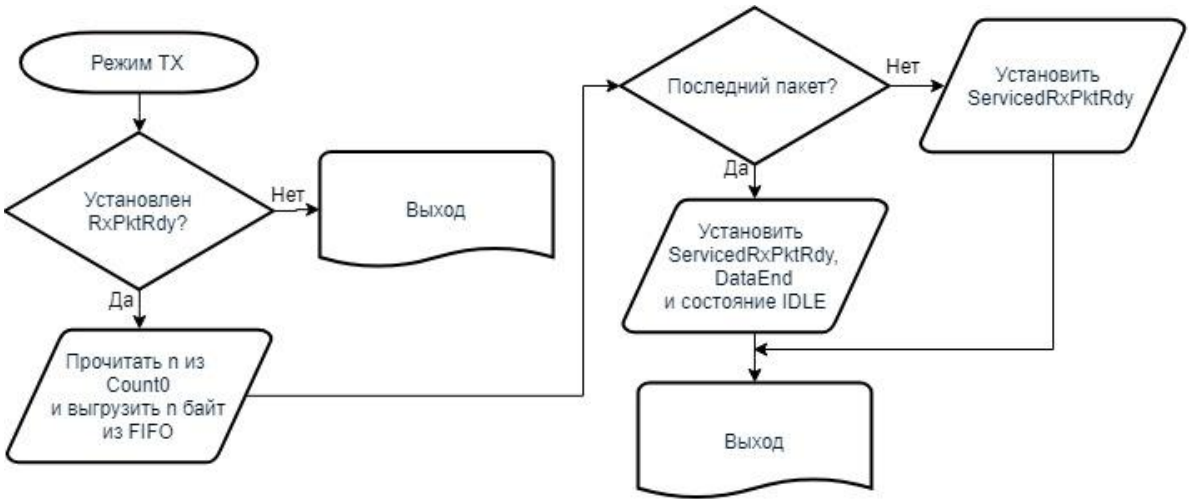



Рисунок 170 – Алгоритм режима RX

***Работа с ошибками как периферийного устройства***

Передача управления может быть прервана из-за ошибки протокола на USB, если хост преждевременно прекращает передачу, или если программное обеспечение функций контроллера хочет прервать передачу (например, потому что оно не может обработать команду).

USB-контроллер автоматически обнаружит ошибки протокола и отправит пакет STALL на хост при следующих условиях:

- Хост отправляет больше данных во время фазы передачи данных OUT запроса на запись, чем указано в команде. Это условие обнаруживается, если хост отправляет токен OUT после того, как бит DataEnd (CSR0.D3) был установлен.
- Хост запрашивает больше данных во время фазы передачи данных IN запроса на чтение, чем указано в команде. Это условие обнаруживается, когда хост отправляет токен IN после того, как бит DataEnd (CSR0) был установлен.
- Хост отправляет больше чем MaxP байт данных в пакете данных OUT.
- Хост отправляет пакет DATA1 с ненулевой длиной во время фазы STATUS запроса на чтение.

Когда USB-контроллер отправит пакет STALL, он установит бит SentStall (CSR0.D2) и сгенерирует прерывание. Когда программное обеспечение получит прерывание конечной точки 0 с установленным битом SentStall, оно должно прервать текущую передачу, сбросить бит SentStall и вернуться в состояние IDLE.

Если хост преждевременно завершит передачу, введя фазу STATUS до того, как все данные запроса будут переданы, или отправив новый пакет SETUP до завершения

текущей передачи, то установится бит SetupEnd (CSR0.D4) и сгенерируется прерывание конечной точки 0. Когда программное обеспечение получит прерывание конечной точки 0 с установленным битом SetupEnd, оно должно прервать текущую передачу, установить бит ServicedSetupEnd (CSR0.D7) и вернуться в состояние IDLE. Если бит RxPktRdy (CSR0.D0) установлен, это указывает на то, что хост отправил еще один пакет SETUP, и программное обеспечение должно обработать эту команду.

Если программное обеспечение хочет прервать текущую передачу, поскольку оно не может обработать команду или имеет некоторую другую внутреннюю ошибку, то оно должно установить бит SendStall (CSR0.D5). Затем USB-контроллер отправит пакет STALL на хост, установит бит SentStall (CSR0.D2) и сгенерирует прерывание конечной точки.

***Дополнительные действия***

При работе в качестве периферии ядро USB-контроллера автоматически реагирует на определенные условия на шине USB или действиях хоста. Подробности приведены ниже:

**STALL на управляющих передачах**

Ядро USB-контроллера автоматически выдает STALL на управляющие передачи при следующих условиях:

- Хост отправляет больше данных во время фазы передачи данных OUT управляющей передачи, чем указано в запросе устройства во время фазы SETUP. Это условие обнаруживается контроллером, когда хост отправляет токен OUT (вместо токена IN) после того, как ЦП выгрузил последний пакет OUT и установил DataEnd.
- Хост запрашивает больше данных во время фазы передачи данных IN управляющей передачи, чем указано в запросе устройства во время фазы SETUP. Это условие обнаруживается контроллером, когда хост отправляет токен IN (вместо токена OUT) после того, как ЦП сбросил TxPktRdy и установил DataEnd в ответ на ACK, выданный хостом тому пакету, который должен был быть последним.
- Хост отправляет больше чем MaxP данных с токеном данных OUT.
- Хост отправляет неверный идентификатор PID для фазы состояния OUT для управляющей передачи.
- Хост отправляет пакет данных нулевой длины для фазы состояния OUT.

***Пакет данных OUT с нулевой длинной в управляющих передачах***

Пакет данных OUT с нулевой длинной используется для указания конца передачи управления. При нормальной работе такие пакеты должны приниматься только после того, как будет передан весь запроса устройства (то есть после того, как ЦП установит DataEnd). Если, однако, хост отправляет пакет данных OUT с нулевой длиной до передачи всего запроса устройства, то это говорит о преждевременном завершении передачи. В этом случае контроллер автоматически очистит все токены IN, загруженные ЦП, готовые для фазы передачи данных из FIFO, и установит SetupEnd (CSR0.D4).

#### 15.8.3.2 Управляющие передачи в режиме хоста

Управляющие передачи в режиме хоста проводятся через конечную точку 0, а программное обеспечение необходимо для обработки всех стандартных запросов к устройству, которые могут быть отправлены или получены через конечную точку 0 (как описано в спецификации универсальной последовательной шины, редакция 2.0, глава 9).

Что касается USB-периферии, необходимо обработать три категории стандартных запросов к устройствам: *запросы нулевой информации* (в которых вся информация включена в команду); *запросы на запись* (в котором за командой последуют дополнительные данные); *запросы на чтение* (в которых устройство должно отправить данные обратно на хост).

- Запросы нулевых данных содержат команду SETUP, за которой следует фаза состояния IN

- Запросы на запись включают команду SETUP, за которой следует фаза передачи данных OUT, которая, в свою очередь, сопровождается фазой состояния IN.
- Запросы на чтение включают команду SETUP, а затем фазу передачи данных IN, которая, в свою очередь, сопровождается фазой состояния OUT.

Таймаут может быть установлен так, чтобы ограничить время, в течение которого USB-контроллер будет повторять транзакцию, которая не может быть получена целью (NAK). Этот предел может составлять от 2 до $2^{15}$ кадров/микрокадров и устанавливается через регистр NAKLimit0.

*Примечание* – Перед началом любых транзакций в качестве хоста, необходимо установить регистр FAddr для адресации периферийного устройства. Если устройство подключается первым, в FAddr должен быть установлен ноль. После того, как появляется команда SET_ADDRESS, в FAddr должен быть установлен новый адрес цели.

***Фаза установки в качестве хоста***

Для фазы настройки управляющей транзакции ЦП, управляющий хост-устройством, должен:

1 Загрузить 8 байт требуемой команды запроса устройства в FIFO конечной точки 0.
2 Затем, он должен установить SetupPkt и TxPtRdy (бит CSR0.D3 и CSR0.D1, соответственно). *Примечание – Эти биты необходимо установить вместе.*
3 Затем USB-контроллер отправит токен SETUP, за которым последует 8-байтная команда для конечной точки 0 адресного устройства, и при необходимости повторит попытку.
4 В конце попытки отправки данных, USB-контроллер должен сгенерировать прерывание конечной точки 0 (то есть установить IntrTx.D0). Затем ЦП должен прочитать CSR0, чтобы узнать, был ли установлен бит RxStall (D2), бит Error (D4) или NAK Timeout (D7).

Если RxStall установлен, то цель не приняла команду (например, потому что она не поддерживается целевым устройством) и поэтому выдала ответ STALL.

Если установлена ошибка, это означает, что USB-контроллер попытался отправить пакет SETUP и следующий пакет данных три раза, не получив ответа.

Если установлен NAK Timeout, то USB-контроллер получил ответ NAK, для каждой попытки отправить пакет SETUP за время большее, чем время, установленное в регистре NAKLimit0. Затем USB-контроллер может либо продолжить попытки передачи этой транзакции (пока время передачи снова не истечет), очистив бит времени ожидания NAK, либо прервать транзакцию, очистив FIFO, прежде чем сбросить бит времени ожидания NAK.

Если ни один из параметров RxStall, Error или NAK Timeout не установлен, то фаза настройки прошла, и ЦП должен перейти к следующей фазе передачи данных IN, OUT или фазе состояния IN, указанной для конкретного стандартного запроса к устройству.

***Фаза передачи данных IN в качестве хоста***

Для фазы передачи данных IN управляющей транзакции ЦП, управляющий хост-устройством, должен:

1 Установить ReqPkt (CSR0.D5).
2 Подождать, пока USB-контроллер отправит токен IN и получит требуемые данные.
3 Когда USB-контроллер сгенерирует прерывание конечной точки 0 (то есть установит IntrTx.D0), ЦП должен прочитать CSR0, чтобы установить, какой из бит RxStall (D2), Error (D4), NAK Timeout(D7) или RxPktRdy (D0) был установлен:
   - Если RxStall установлен, то цель выдала ответ STALL.

- Если установлен Error, то USB-контроллер попытался отправить требуемый токен три раза и не получил ответ.
- Если установлен NAK Timeout, то USB-контроллер получил NAK, для каждой попытки отправить пакет SETUP за время превышающее установленное в регистре NAKLimit0. Затем USB-контроллер может либо продолжить попытки передать эту транзакцию, сбросив бит NAK Timeout, либо прервать транзакцию, сбросив ReqPkt перед сбросом бита NAK Timeout.

4 Если RxPktRdy был установлен, ЦП должен прочитать данные из FIFO конечной точки 0, а затем сбросить RxPktRdy.
5 Если ожидаются дополнительные данные, ЦП должен повторить шаги 1 – 4.
6 Когда все данные были успешно получены, ЦП должен перейти на фазу состояния OUT контрольной транзакции.

***Фаза передачи данных OUT в качестве хоста***

Для фазы передачи данных OUT управляющей транзакции процессор, управляющий хост-устройством, должен:

1 Загрузить данные, которые нужно отправить в FIFO конечной точки 0
2 Установить бит TxPtRdy (CSR0.D1).
3 Затем USB-контроллер должен отправить токен OUT, за которым последуют данные из FIFO в конечную точку 0 адресного устройства. Этот этап повторится при необходимости.
В конце попытки отправить данные USB-контроллер должен сгенерировать прерывание конечной точки 0 (то есть установить IntrTx.D0). Затем ЦП должен прочитать CSR0, чтобы узнать какой из бит RxStall (D2), Error (D4) или NAK Timeout (D7) был установлен.
Если RxStall установлен, то цель выдала ответ STALL.
Если установлен Error, то USB-контроллер попытался отправить токен OUT и следующий пакет данных три раза, не получив ответа.
Если установлен NAK Timeout, то USB-контроллер получил ответ NAK, для каждой попытки отправить токен OUT, за время, превышающее установленное в регистре NAKLimit0. Затем USB-контроллер может либо продолжить попытки отправки этой транзакции, сбросив бит NAK Timeout, либо прервать транзакцию, очистив FIFO, прежде чем сбросить бит NAK Timeout.
Если ни один из бит RxStall, Error или NAK Limit не установлен, данные OUT были корректно получены.
4 Если необходимо отправить дополнительные данные, ЦП должен повторить шаги 1 – 3.

Когда все данные были успешно отправлены, ЦП должен перейти в фазу состояния IN контрольной транзакции.

***Фаза состояния IN в качестве хоста (следует за фазой SETUP или за фазой передачи данных OUT)***

Для фазы состояния IN контрольной транзакции, ЦП, управляющий хост-устройством, должен:

1. Установить StatusPkt и ReqPkt (биты CSR0.D6 и CSR0.D5). *Примечание – эти биты необходимо установить вместе.*
2. Подождать, пока USB-контроллер отправит токен IN и получит ответ от USB-периферии.
3. Когда USB-контроллер сгенерирует прерывание конечной точки 0 (то есть установит IntrTx.D0), ЦП должен прочитать CSR0, чтобы установить, какой из бит RxStall (D2), Error (D4), NAK Timeout (D7) или RxPktRdy (D0) был установлен.
Если RxStall установлен, то цель не смогла выполнить команду и поэтому выдала ответ STALL.

Если Error установлен, то USB-контроллер попытался отправить требуемый токен три раза и не получил ответ.

Если NAK Timeout установлен, то USB-контроллер получил ответ NAK для каждой попытки отправить токен IN за время, превышающее установленное в регистре NAKLimit0. Затем контроллер может либо продолжить отправку этой транзакции, сбросив бит NAK Timeout, либо прервать транзакцию, сбросив ReqPkt и StatusPkt перед сбросом NAK Timeout.

4. Если RxPktRdy установлен, ЦП должен просто сбросить RxPktRdy.

***Фаза состояния OUT в качестве хоста (следует за фазой передачи данных IN)***

Для фазы состояния OUT управляющей транзакции, ЦП, управляющий хост-устройством, должен:

1 Установить StatusPkt и TxPktRdy (бит CSR0.D6 и CSR0.D1). *Примечание – Эти биты необходимо установить вместе.*

2 Дождаться, пока USB-контроллер отправит токен OUT и пакет DATA1 с нулевой длиной.

3 В конце попытки отправить данные USB-контроллер сгенерирует прерывание конечной точки 0 (то есть установит IntrTx.D0). Затем ЦП прочитает CSR0, чтобы узнать, какой из бит RxStall (D2), Error (D4) или NAK Timeout (D7) был установлен:
   - Если RxStall установлен, то цель не смогла выполнить команду и поэтому выдала ответ STALL.
   - Если Error установлен, то USB-контроллер попытался отправить пакет STATUS и следующий пакет данных три раза, но не получил ответ.
   - Если NAK Timeout установлен, это означает, что USB-контроллер получил ответ NAK для каждой попытки отправить токен IN за время большее, чем установлено в регистре NAKLimit0. Затем контроллер может либо продолжить попытки этой транзакции, сбросив бит NAK Timeout, либо прервать транзакцию, сбросив FIFO, прежде чем сбросить NAK Timeout.

4 Если ни один из RxStall, Error или NAK Timeout не установлен, фаза STATUS была проведена правильно.

### 15.8.4 *Поточные (Bulk) передачи транзакций*

#### 15.8.4.1 Обработка поточных транзакций как периферийное устройство

***Поточные транзакции IN***

Четыре дополнительных параметра доступны для использования с конечной точкой Tx, используемой в периферийном режиме для поточных передач транзакций IN:

- Двухпакетная буферизация

  Если значение, записанное в регистр TxMaxP, меньше или равно половине размера FIFO, выделенного конечной точке, двойная буферизация пакетов будет активирована автоматически.

- AutoSet

  Когда функция AutoSet включена, бит TxPktRdy (TxCSR.D0) будет установлен, после загрузки пакета размера TxMaxP байт в FIFO.

- Автоматическое разделение пакетов

  Этот параметр конфигурации позволяет активировать запись больших пакетов данных в поточные конечные точки, которые затем разделяются на пакеты соответствующего (заданного) размера для передачи по шине USB.

**Настройка**

При настройке конечной точки Tx для поточных транзакций в регистр TxMaxP должен быть записан размер (в байтах) максимального размера пакета для конечной точки. Это значение должно совпадать с полем wMaxPacketSize стандартного дескриптора конечной точки. Кроме того, соответствующий бит разрешения прерывания в регистре IntrTxE должен быть установлен (если для этой конечной точки требуется прерывание), а старший байт регистра TxCSR должен быть установлен, как показано ниже (биты D9 - D8 не используются):

| | | | |
|---|---|---|---|
| D15 | **AutoSet** | 0/1 | Установите значение 1, если требуется функция AutoSet |
| D14 | **ISO** | 0 | Установите значение 0, чтобы включить поточный протокол |
| D13 | **Mode** | 1 | Установите значение 1, чтобы гарантировать, что FIFO доступен (необходимо только, если FIFO используется совместно с конечной точкой Rx) |
| D11 | **FrcDataTog** | 0 | Установите значение 0, чтобы установить нормальное Data Toggle |

Если конечная точка сконфигурирована впервые (при помощи команды SET_CONFIGURATION или SET_INTERFACE в конечной точке 0), бит ClrDataTog (D6) должен быть установлен в младший байт. Это гарантирует, что Data Toggle (которое автоматически обрабатывается контроллером) запустится в правильном состоянии. Также, если в FIFO есть какие-либо пакеты данных (установлен бит FIFONotEmpty (TxCSR.D1)), их следует удалить, установив бит FlushFIFO (TxCSR.D3).

Примечание – Возможно, потребуется установить этот бит дважды, если включена двойная буферизация.

**Процесс работы**

Когда нужно передать данные по поточному каналу IN, пакет данных нужно загрузить в FIFO, и бит TxPktRdy (D0) нужно установить в регистр TxCSR. Когда пакет будет отправлен, контроллер сбросит бит TxPktRdy и сгенерирует прерывание, по которому можно загружать следующий пакет в FIFO. Если включена двойная буферизация пакетов, то после того, как первый пакет был загружен и установлен бит TxPktRdy, контроллер немедленно сбросит бит TxPktRdy и сгенерирует прерывание, по которому можно загружать второй пакет в FIFO. Программное обеспечение должно работать одинаково, загружая пакет, когда оно получает прерывание, независимо от того, включена ли двойная буферизация пакетов или нет.

В общем случае размер пакета не должен превышать размер, указанный нижними 11-тью битами регистра TxMaxP. Эта часть регистра определяет полезную нагрузку (размер пакета) для передачи по USB и требуется по спецификации USB для 8-ми, 16-ти, 32-х, 64-х (full-/high-speed) или 512-ти байт (high-speed). Если необходимо передать данные большего размера, то нужно отправить их как несколько USB-пакетов, которые должны быть размера TxMaxP [D10: D0], за исключением последнего пакета, размер которого неважен.

Если была выбрана опция автоматического разделения пакетов при настройке ядра (это можно определить по биту MPTxE (D6) регистра ConfigData), то в FIFO могут быть записаны пакеты в 32 раза превышающие размер, указанный в TxMaxP [D10:D0] (предполагается, что FIFO достаточно велик, чтобы принимать эти пакеты), которые затем разделятся ядром на пакеты нужного размера для передачи по USB. Размер пакетов, записанных в FIFO, задается как m × полезная нагрузка, где TxMaxP [D15: D11] = m - 1. ПО должно установить соответствующие значения в регистр TxMaxP и в биты 10:0 поля wMaxPacketSize стандартного дескриптора конечной точки. Сам же процесс передачи больших пакетов ничем не отличается от процесса поточной передачи пакета стандартного размера.

**Обработка ошибок**

Если программное обеспечение хочет остановить поточный канал IN, оно должно установить бит SendStall (TxCSR.D4). Когда контроллер получит следующий токен IN, он отправит STALL на хост, установит бит SentStall (TxCSR.D5) и сгенерирует прерывание.

Если программное обеспечение получит прерывание с битом SentStall (TxCSR.D5), оно должно будет сбросить бит SentStall. Тем не менее оно должно оставить бит SendStall (TxCSR.D4) до тех пор, пока оно не будет готово повторно активировать поточный канал IN.

Примечание – Если хост не смог получить STALL по какой-либо причине, он отправит еще один токен IN, поэтому рекомендуется оставить бит SendStall установленным до тех пор, пока программное обеспечение не будет готово повторно активировать поточный канал IN. После включения канала, необходимо сбросить Data Toggle, установив бит ClrDataTog в регистре TxCSR (D6).

***Поточная транзакции OUT в качестве периферии***

Четыре дополнительных функции доступны для использования с конечной точкой Rx, используемой в периферийном режиме для поточных транзакций OUT.

- Двухпакетная буферизация

  Если значение, записанное в регистр RTxMaxP, меньше или равно половине размера FIFO, выделенного конечной точке, то двойная буферизация пакетов будет активирована автоматически.
- AutoClear

  Когда функция AutoClear включена, бит RxPktRdy (RxCSR.D0) будет автоматически сброшен, когда пакет размера RxMaxP байт будет выгружен из FIFO.
- Автоматическое объединение пакетов

  Этот параметр конфигурации позволяет активировать чтение пакетов данных, которые затем объединяется в пакеты большего размера для последующего их чтения программным обеспечением.

**Настройка**

При настройке конечной точки Rx для поточных транзакций OUT, необходимо записать максимальный размер пакета (в байтах) для конечной точки в регистр RxMaxP. Это значение должно совпадать с полем wMaxPacketSize стандартного дескриптора конечной точки. Кроме того, соответствующий бит разрешения прерывания должен быть установлен в регистре IntrRxE (если для этой конечной точки требуется прерывание), а старший байт регистра RxCSR должен быть установлен, как показано ниже (биты D10 - D8 не используются/только для чтения):

| | | | |
|---|---|---|---|
| D15 | **AutoClear** | 0/1 | Установите значение 1, если требуется функция AutoClear |
| D14 | **ISO** | 0 | Установите значение 0, чтобы включить поточный протокол передачи |
| D12 | **DisNyet** | 0 | Установите значение 0, чтобы использовать нормальный контроль потока PING |

Если конечная точка конфигурируется впервые (после команды SET_CONFIGURATION или SET_INTERFACE конечной точке 0), бит ClrDataTog (D7) должен быть установлен в младший байт RxCSR. Это гарантирует, что Data Toggle (которое автоматически обрабатывается контроллером) запустится в правильном состоянии. Также, если в FIFO уже есть какие-либо пакеты данных (указано в бите RxPktRdy (RxCSR.D0)), их следует удалить, установив бит FlushFIFO (RxCSR.D4).

Примечание – Возможно, потребуется установить этот бит дважды, если включена двойная буферизация.

**Процесс работы**

После получения пакета данных поточной конечной точкой Rx, установится бит RxPktRdy (RxCSR.D0) и сгенерируется прерывание. Программное обеспечение должно

прочитать регистр RxCount для конечной точки, чтобы определить размер пакета данных. Чтобы бит RxPktRdy сбросился, пакет данных следует прочитать из FIFO.

Полученные пакеты данных не должны превышать размер, указанный в регистре RxMaxP (так как это должно быть значение, указанное в поле wMaxPacketSize дескриптора конечной точки, отправленное на хост). Когда блок данных, размер которого превышает wMaxPacketSize, должен быть отправлен в функцию, он будет отправлен как несколько пакетов. Все пакеты будут размером wMaxPacketSize, за исключением последнего.

Если включена опция автоматического объединения пакетов (это можно определить по биту MPRxE (D7) в регистре ConfigData), то ядро может принимать до 32 пакетов одновременно и объединять их в один пакет в FIFO (предполагается, что FIFO достаточно велик, чтобы хранить более крупные пакеты). Размер пакетов, записанных в FIFO, определяется как m × wMaxPacketSize, где RxMaxP [D15: D11] = m - 1. Программное обеспечение, должно установить соответствующие значения в регистре RxMaxP и в биты 10:0 поля wMaxPacketSize дескриптора конечной точки.

**Обработка ошибок**

Если программное обеспечение хочет отключить поточный канал OUT, оно должно установить бит SendStall (RxCSR.D5). Когда USB-контроллер получит следующий пакет, он отправит STALL на хост, установит бит SentStall (RxCSR.D6) и сгенерирует прерывание.

Когда программное обеспечение получит прерывание с битом SentStall (RxCSR.D6), оно должно сбросить его. Однако оно должно держать бит SendStall (RxCSR.D5) установленным до тех пор, пока оно не будет готово повторно активировать поточный канал OUT.

*Примечание* – Если хост не смог получить STALL по какой-либо причине, он отправит еще один пакет, поэтому рекомендуется оставить бит SendStall установленным до тех пор, пока программное обеспечение не будет готово повторно активировать поточный канал OUT. Когда поточный канал OUT снова включится, Data Toggle следует перезапустить, установив бит ClrDataTog в регистр RxCSR (D7).

#### 15.8.4.2 Обработка поточных передач транзакций как хост

***Поточные передачи транзакции IN как хост***

Пять дополнительных функций доступны для использования с конечной точкой Rx, используемой в режиме хоста, для получения данных:

- Двухпакетная буферизация

  Когда включена двойная буферизация пакетов, один пакет может быть принят во время чтения другого. Двойная буферизация пакетов будет автоматически включена, если значение, записанное в регистр RxMaxP, меньше или равно половине размера FIFO, выделенного конечной точке.
- AutoReq(uest)

  Когда включена функция AutoReq(uest), бит ReqPkt (RxCSR.D5) будет автоматически устанавливаться при сбросе бита RxPktRdy.
- AutoClear

  Если функция AutoClear включена, бит RxPktRdy (RxCSR.D0) будет автоматически сброшен, когда пакет размера RxMaxP байт будет выгружен из FIFO.
- Автоматическое объединение пакетов

  Этот параметр конфигурации позволяет активировать чтение пакетов данных, которые затем объединяется в пакеты большего размера для последующего их чтения программным обеспечением.

**Настройка**

Перед инициализацией любой поточной передачи транзакции IN в режиме хоста:

- Должен быть установлен адрес целевой функции.

В регистре RxType используемой конечной точкой должны быть установлены биты D7, D6 для выбора рабочей скорости, биты D5, D4 = 10 (для выбора режима поточной передачи) и в битах D3 - D0 указан номер конечной точки, указанный в дескрипторе соответствующей конечной точки IN, присвоенной в процессе перечисления устройств.

- В регистр RxMaxP конечной точки должен быть записан максимальный размер пакета для передачи (в байтах). Это значение должно совпадать со значением в поле MaxPacketSize стандартного дескриптора конечной точки.
- В регистр RxInterval должно быть записано значение NAK Limit (2 – 215 кадров/микрокадров) или ноль, если функция NAK Timeout не требуется.
- Соответствующий бит разрешения прерывания должен быть установлен в регистре IntrRxE (если для этой конечной точки требуется прерывание)
- Биты регистра RxCSR должны быть установлены, как показано ниже:

| | | | |
|---|---|---|---|
| D15 | **AutoClear** | 0/1 | Установите значение 1, если требуется функция AutoClear |
| D14 | **AutoReq** | 0/1 | Установите значение 1, если требуется функция AutoReq |
| D12 | **DisNyet** | 0 | Установите значение 0, чтобы обеспечить нормальный контроль потока PING |

Если конечная точка конфигурируется впервые, то Data Taggle должен быть установлен на 0 либо с помощью битов Data Toggle Write Enable и Data Toggle (RxCSR.D10 и D9), либо путем установки бита ClrDataTog (D7) в младшем байте RxCSR. Также, если в FIFO уже есть какие-либо пакеты данных (определяется по биту RxPktRdy (RxCSR.D0)), их следует удалить, установив бит FlushFIFO (RxCSR.D4).

Примечание – Возможно, потребуется установить этот бит дважды, если включена двойная буферизация.

**Процесс работы**

Когда потребуется совершить поточную передачу данных с периферийного устройства USB, ЦП должен установить бит ReqPkt в соответствующий регистр RxCSR (D5). Затем контроллер отправит токен IN в выбранную конечную точку периферийного устройства и будет ожидать возвращения данных.

Если данные принимаются правильно, устанавливается RxPktRdy (RxCSR.D0). Если периферийное устройство отвечает STALL-ом, устанавливается RxStall (RxCSR.D6). Если получен NAK, контроллер будет повторять отправку до тех пор, пока транзакция не передастся успешно или не будет установлен бит NAKLimit в регистре RxInterval. Если никакого ответа не получено вообще, контроллер предпримет еще две дополнительные попытки, после чего он сообщит об ошибке (установит RxCSR.D2).

Затем контроллер сгенерирует соответствующее прерывание конечной точки, после чего ЦП должен прочитать соответствующий регистр RxCSR, чтобы определить, какой бит из RxPktRdy, RxStall, Error или NAK Timeout установлен. Если установлен бит NAK Timout, контроллер либо продолжит пытаться отправить эту транзакцию, сбросив бит NAK Timeout, либо прервет передачу транзакции, сбросив ReqPkt, прежде чем сбросить бит NAK Timeout.

Полученные пакеты не должны превышать размер, указанный в регистре RxMaxP. Если необходимо отправить блок данных больший, чем RxMaxP, то он будет отправлен как несколько пакетов.

Если опция автоматического объединения пакетов была выбрана при настройке ядра (это можно определить по биту MPRxE (D7) в регистре ConfigData), то ядро может принимать до 32 пакетов одновременно и объединять их в один пакет в FIFO (предполагается, что FIFO достаточно велик, чтобы хранить более крупные пакеты). Размер пакетов, записанных в FIFO, определяется как m × wMaxPacketSize, где RxMaxP [D15: D11] = m - 1. ПО должно задать соответствующие значения в регистре RxMaxP и те же значение в биты 10:0 в поле wMaxPacketSize дескриптора конечной точки.

Все пакеты будут размером wMaxPacketSize, за исключением последнего.

**Обработка ошибок**

Если цель захочет закрыть поточный канал IN, она отправит ответ STALL на токен IN. Это приведет к установке бита RxStall (RxCSR.D6).

### *Поточные транзакции out как хост*

Четыре дополнительных функции доступны для использования с конечной точкой Tx, используемой в режиме хоста, для поточной передачи данных:

- Двухпакетная буферизация

  Двойная буферизация пакетов будет активирована автоматически, если значение, записанное в регистр TxMaxP, меньше или равно половине размера FIFO, выделенного конечной точке. До двух пакетов может храниться в ожидании передачи на периферийное устройство.
- AutoSet

  Если функция AutoSet включена, бит TxPktRdy (TxCSR.D0) будет установлен, когда пакет размера TxMaxP байт будет загружен в FIFO.
- Автоматическое разделение пакетов

  Этот параметр конфигурации позволяет активировать запись больших пакетов данных в поточные конечные точки, которые затем разделяются на пакеты соответствующего (заданного) размера для передачи по шине USB.

**Настройка**

Перед началом любых поточных передач транзакций OUT необходимо:

- Установить адрес целевой функции.

  В регистр TxType соответствующей конечной точки, должны быть записаны биты D7, D6, для выбора скорости работы, бит D5, D4 = 10 (для выбора режима поточной передачи) и в битах D3 - D0, указан номер конечной точки, указанный в дескрипторе конечной точки OUT, присвоенный в процессе перечисления устройств.
- В регистр TxMaxP конечной точки должен быть записан максимальный размер пакета (в байтах) для передачи. Это значение должно быть таким же, как и в поле wMaxPacketSize стандартного дескриптора конечной точки.
- В регистр TxInterval должно быть записано значение NAK Limit ($2 - 2^{15}$ кадров/микрокадров) или ноль, если функция NAK Timeout не требуется.
- Бит разрешения прерывания должен быть установлен в регистре IntrTxE (если для этой конечной точки требуется прерывание).
- Следующие биты регистра TxCSR должны быть установлены, как показано ниже:

| | | | |
|---|---|---|---|
| D15 | **AutoSet** | 0/1 | Установите значение 1, если требуется функция AutoSet |
| D13 | **Mode** | 1 | Установите значение 1, чтобы гарантировать, что FIFO доступен (необходимо только в случае, если FIFO используется совместно с конечной точкой Rx) |
| D11 | **FrcDataTog** | 0 | Установите значение 0, чтобы разрешить нормальное Data Toggle |

Если конечная точка конфигурируется впервые, то Data Toggle конечных точек должен быть установлен на 0 либо с помощью битов Data Toggle Write Enable и Data Toggle (TxCSR.D9 и D8), либо путем установки бита ClrDataTog (D6) в младший байт TxCSR. Это гарантирует, что Data Toggle (которое автоматически обрабатывается контроллером) запустится в правильном состоянии. Кроме того, если в FIFO есть какие-либо пакеты данных (определяется по биту FIFONotEmpty (TxCSR.D1)), их следует удалить, установив бит FlushFIFO (TxCSR.D3).

*Примечание* – Возможно, потребуется установить этот бит дважды, если включена двойная буферизация.

**Процесс работы**

Когда требуется совершить поточную передачу данных, ЦП должен записать первый пакет данных в FIFO (или два пакета, если используется двойной буфер), и установить бит TxPktRdy в соответствующий регистр TxCSR (D0). Затем контроллер отправит токен OUT в выбранную конечную точку периферийного устройства, а затем первый пакет данных из FIFO.

Если данные правильно приняты периферийным устройством, придет ACK, после чего контроллер сбросит TxPktRdy (TxCSR.D0). Если периферийное устройство USB ответит STALL-ом, то установится RxStall (TxCSR.D5). Если получен NAK, будет повторять передачу до тех пор, пока транзакция не будет успешно завершена или не будет установлен NAKLimit в регистре TxInterval. Если никакого ответа вообще не получено, произойдет еще две дополнительные попытки отправить данные, а после контроллер сообщит об ошибке (TxCSR.D2).

Затем контроллер сгенерирует соответствующее прерывание конечной точки, после чего ЦП должен будет считать соответствующий регистр TxCSR, чтобы определить, установлены из бит RxStall (D5), Error (D2) или NAK Timeout (D7) установлен. Если установлен бит NAK Timeout, контроллер либо продолжит попытки отправки этой транзакции, сбросив бит NAK Timeout, либо прервет передачу, очистив FIFO, прежде чем сбросить бит NAK Timeout.

В общем случае размер пакета не должен превышать размер, указанный младшими 11-ю битами регистра TxMaxP (который должен был быть установлен в соответствии со значением, установленным в поле wMaxPacketSize соответствующего дескриптора конечной точки). Когда необходимо отправить блок данных с размером, превышающим TxMaxP, его нужно будет отправить как несколько пакетов. Эти пакеты должны быть размера TxMaxP [D10: D0], за исключением последнего пакета.

Если была выбрана опция автоматического разделения пакетов при конфигурации ядра (это можно определить по настройке бита MPTxE (D6) регистра ConfigData), то, пакеты с размером в 32 раза превышающим размер, указанный в TxMaxP [D10: D0], могут быть записаны в FIFO (предполагается, что FIFO достаточно велик, чтобы принимать большие пакеты), которые затем разобьются ядром на пакеты соответствующего размера для передачи по USB. Размер пакетов, записанных в FIFO, задается как m × USB-полезная нагрузка, где TxMaxP [D15: D11] = m - 1. ПО должно задать соответствующие значения в регистре TxMaxP.

Если общий размер блока данных кратен TxMaxP, хосту может потребоваться отправить нулевой пакет после отправки всех данных. Это можно сделать, установив TxPktRdy после получения последнего прерывания и не загружая каких-либо данных в FIFO.

**Обработка ошибок**

Если цель хочет выключить поточный канал OUT, она должна отправить ответ STALL. Это обозначается битом RxStall (TxCSR.D5).

### 15.8.5 *Full-speed/low-bandwidth (низкопропускные) передачи по прерываниям*

#### 15.8.5.1 Передачи по прерываниям как периферия

Передача по прерываниям IN использует тот же протокол, что и поточная передача IN, и может использоваться таким же образом. Точно так же передача по прерываниям OUT использует почти тот же протокол, что и поточная передача OUT, и может использоваться таким же образом.

Тем не менее, следует отметить, что конечные точки Tx USB-контроллера, который используется в качестве периферийного устройства, поддерживают одну функцию передачи по прерываниям IN, которую они не поддерживают в поточных передачах IN – они поддерживают непрерывное переключение бита Data Toggle. Эта функция активируется установкой бита FrcDataTog в регистре TxCSR (D11). Когда этот бит

установлен, контроллер посчитает пакет успешно отправленным и переключит data toggle, независимо от того, получен ли ACK от хоста или нет.

Другое отличие заключается в том, что конечные точки по прерываниям не поддерживают управление потоком PING. Это означает, что контроллер никогда не должен отвечать на пакет квитирования NYET, только на ACK / NAK / STALL. Для этого бит DisNyet в регистре RxCSR (D12) должен быть установлен, чтобы отключить передачу пакетов квитирования NYET в high-speed режиме.

#### 15.8.5.2 Передачи прерываний как хост

Когда USB-контроллер работает в качестве хоста, взаимодействие с конечной точкой по прерываниям на периферии USB обрабатываются почти так же, как поточные передачи, исключением того, что разрешены high-bandwidth передачи по прерываниям.

Основное различие заключается в том, что RxType [5: 4] и TxType [5: 4] необходимо установить равным 11 (для представления передачи Interrupt), а не 10.

Необходимый интервал опроса также необходимо установить в регистры RxInterval/TxInterval.

### 15.8.6 *Full-speed/low-bandwidth (низкопропускные) изохронные передачи*

#### 15.8.6.1 Работа с изохронными передачами как с периферией

***Изохронные передачи IN***

Три дополнительных функции доступны для использования с конечной точкой Tx в периферийном режиме для изохронной передачи транзакций IN:

- Двойная буферизация пакетов

  Двойная буферизация пакетов автоматически включится, если значение, записанное в регистр TxMaxP, меньше или равно половине размера FIFO, выделенного конечной точке. Если включено, в FIFO может находиться до двух пакетов в ожидании передачи на хост.
- AutoSet

  Когда функция AutoSet включена, бит TxPktRdy (TxCSR.D0) будет автоматически устанавливаться, когда пакет размера TxMaxP байт будет загружаться в FIFO.

**Настройка**

При конфигурировании конечной точки Tx для изохронных передач IN, максимальный размер пакета (в байтах) должен быть записан в регистр TxMaxP. Это значение должно совпадать с полем wMaxPacketSize стандартного дескриптора конечной точки. Кроме того, соответствующий бит разрешения прерывания должен быть установлен в регистре IntrTxE (если требуется прерывание для этой конечной точки), а старший байт регистра TxCSR должен быть установлен, как показано ниже (биты D9 - D8 не используются):

| | | | |
|---|---|---|---|
| D15 | **AutoSet** | 0/1 | Установите значение 1, если требуется функция AutoSet |
| D14 | **ISO** | 1 | Установите значение 1 для включения изохронного протокола передачи |
| D13 | **Mode** | 1 | Установите значение 1, чтобы удостовериться, что FIFO доступен (необходимо если только FIFO используется совместно с конечной точкой Rx) |
| D11 | **FrcDataTog** | 0 | Игнорируется в изохронном режиме |

**Процесс работы**

Изохронная конечная точка не поддерживает повторение попыток отправки данных, поэтому, если необходимо избежать переполнения данных, отправляемых на узел, они должны загружать данные в FIFO до получения токена IN. Хост будет отправлять один токен для каждого кадра, однако время в пределах кадра (или микрокадра) может меняться. Если токен IN принимается ближе к концу одного кадра, а затем в начале следующего кадра, то времени для перезагрузки FIFO будет мало. По этой причине обычно необходима двойная буферизация конечной точки.

Функция AutoSet может использоваться с изохронной конечной точкой Tx так же, как с поточной конечной точкой Tx. Однако, если данные не поступают из источника с постоянной скоростью, синхронизированной с часами кадра хоста, размер пакетов, отправленных хосту, должен увеличиваться или уменьшаться от кадра к кадру (или от микрокадра к микрокадру), чтобы соответствовать исходной скорости передачи данных. Это означает, что фактические размеры пакетов не всегда будут TxMaxP, что делает функцию AutoSet бесполезной.

Прерывание генерируется всякий раз, когда пакет отправляется на хост, и программное обеспечение может использовать это прерывание для загрузки следующего пакета в FIFO и установки бита TxPktRdy в регистр TxCSR (D0) так же, как для поточной конечной точки Tx. Поскольку прерывание может происходить почти в любое время внутри кадра (/микрокадра), в зависимости от того, когда хост запланировал передачу данных, это может привести к неправильным моментам времени подачи запросов на загрузку FIFO. Если источник данных для конечной точки поступает с какого-либо внешнего оборудования, может быть удобнее дождаться конца каждого кадра (микрокадра) до следующей загрузки FIFO, поскольку это минимизирует потребность в дополнительной буферизации. Это можно сделать, используя прерывание SOF (IntrUSB.D3) или внешний SOF_PULSE-сигнал от USB-контроллера, чтобы инициировать загрузку следующего пакета данных. SOF_PULSE генерируется один раз на кадр (микрокадр), когда принимается пакет SOF. (контроллер также поддерживает внешний счетчик кадров (микрокадров), чтобы он мог генерировать SOF_PULSE, когда пакет SOF был потерян.) Прерывания все еще могут использоваться для установки бита TxPktRdy в TxCSR (D0).

Источником проблем может стать запуск изохронного IN канала с двойной буферизацией. Двойная буферизация требует, чтобы пакет данных не передавался до тех пор, пока кадр (микрокадр) не будет загружен. Проблем нет, если функция загружает первый пакет данных, или по крайней мере, кадр (микрокадр), прежде чем хост настроит канал (и, следовательно, начнет посылать токены IN). Но если хост уже начал посылать токены IN к моменту загрузки первого пакета, то возможно пакет должен быть передан в том же кадре (микрокадре), в котором он был загружен, в зависимости от того, загружен ли он до или после получения IN токена. Эту потенциальную проблему можно избежать, установив бит ISO Update в регистр Power (D7). Когда этот бит установлен, любой пакет данных, загружаемый в FIFO изохронной конечной точки Tx, не будет передаваться до тех пор, пока не будет принят следующий пакет SOF, тем самым гарантируя, что пакет данных не будет отправлен слишком рано.

#### Обработка ошибок

Если конечная точка не имеет данных в своем FIFO, когда принимается токен IN, он отправит нулевой пакет данных на хост и установит бит UnderRun в регистре TxCSR (D2). Это свидетельствует о том, что программное обеспечение не обеспечивает доставку данных достаточно быстро для хоста.

Если программное обеспечение загружает один пакет на кадр (микрокадр), и обнаруживает, что бит TxPktRdy в регистре TxCSR (D0) установлен, когда он хочет загрузить следующий пакет, это означает, что пакет данных не был отправлен (возможно, потому что токен IN от хоста был поврежден). ПО может либо сбросить неотправленный пакет, установив бит FlushFIFO в регистр TxCSR (D3) либо может пропустить этот пакет.

### *Изохронные передачи OUT*

Три дополнительных функции доступны для конечной точки Rx, используемой в периферийном режиме для изохронных передач транзакций OUT:

- Двойная буферизация пакетов

  Двойная буферизация пакетов автоматически включается, когда значение, записанное в регистр RxMaxP, меньше или равно половине размера FIFO, выделенного конечной точке. При включении этой опции, в FIFO может быть сохранено до двух пакетов, ожидающих передачи на хост.

Примечание – Для предотвращения синхронных передач данных рекомендуется использовать двухбуквенную буферизацию, чтобы избежать ошибок при переполнении (см. «Процесс работы» ниже).

- AutoClear

  Когда функция AutoClear включена, бит RxPktRdy (RxCSR.D0) будет автоматически сбрасываться, когда пакет размера RxMaxP байт будет выгружен из FIFO.

**Настройка**

При конфигурировании конечной точки Rx для изохронной передачи данных OUT, максимальный размер пакета (в байтах) для конечной точки должен быть записан в регистр RxMaxP. Это значение должно совпадать со значение в поле wMaxPacketSize стандартного дескриптора конечной точки. Кроме того, должен быть установлен соответствующий бит разрешения прерывания в регистре IntrRxE (если для этой конечной точки требуется прерывание), а старший байт регистра RxCSR должен быть установлен, как показано ниже (биты D10 – D8 не используются / только для чтения):

| | | | |
|---|---|---|---|
| D15 | **AutoSet** | 0/1 | Установите значение 1, если требуется опция AutoSet |
| D14 | **ISO** | 1 | Установите значение 1 для включения изохронного протокола |
| D13 | **Mode** | 1 | Установите значение 1, чтобы удостовериться, что FIFO доступен (необходимо если только FIFO используется совместно с конечной точкой Rx) |
| D11 | **FrcDataTog** | 0 | Игнорируется в изохронном режиме |

**Процесс работы**

Изохронная конечная точка не поддерживает повторения попыток передачи данных, поэтому, если необходимо избежать переполнения данных, в FIFO должно быть место для приема пакета при его получении. Хост отправляет один пакет на кадр, однако время в пределах кадра может меняться. Если пакет принят ближе к концу одного кадра (микрокадра), а другой - к началу следующего кадра, времени для разгрузки FIFO будет мало. По этой причине обычно необходима двойная буферизация конечной точки.

Функция AutoClear может использоваться с изохронной конечной точкой Rx так же, как с поточной конечной точкой Rx. Однако, если приемник данных получает данные не с постоянной скоростью и синхронизируется с часами кадра хоста, размер пакетов, отправленных с хоста, должен увеличиваться или уменьшаться от кадра к кадру (или от микрокадра до микрокадра) до соответствия требуемой скорости передачи данных. Это означает, что фактические размеры пакетов не всегда будут равны RxMaxP, что делает функцию AutoClear бесполезной.

**Обработка ошибок**

Если в FIFO нет места для хранения пакета, полученного от хоста, будет установлен бит OverRun в регистре RxCSR (D2). Это свидетельствует о том, что программное обеспечение не выгружает данные достаточно быстро для хоста.

Если USB-контроллер обнаружит, что принятый пакет имеет ошибку CRC, он все равно разместит пакет в FIFO и установит биты RxPktRdy (RxCSR.D0) и DataError (RxCSR.D3).

#### 15.8.6.2 Работа с изохронными передачами как хоста

***Изохронные передачи IN***

Четыре дополнительных опции доступны для использования с конечной точкой Rx, используемой в режиме хоста, для получения этих данных:

- Двойная буферизация пакетов

  Когда включена двойная буферизация пакетов, один пакет может быть принят, пока другой считывается. Двойная буферизация пакетов будет активирована автоматически, если значение, записанное в регистр RxMaxP, будет меньше или равно половине размера FIFO, выделенного конечной точке.

  *Примечание – Двойная пакетная буферизация данных рекомендуется к использованию в синхронных передачах, чтобы избежать переполнения данных (см. «Процесс работы» ниже).*
- AutoClear

  Если опция AutoClear включена, бит RxPktRdy (RxCSR.D0) будет сброшен, когда пакет из RxMaxP байт будет выгружен из FIFO.
- AutoReq(uest)

  Если включена опция AutoReq (uest), бит ReqPkt (RxCSR.D5) будет автоматически устанавливаться, при сбросе бита RxPktRdy.

**Настройка**

Перед началом изохронных транзакций IN:

- Должен быть установлен адрес целевой функции.
- В регистре RxType используемой конечной точки контроллера должны быть установлены биты D7, D6 для выбора рабочей скорости, биты D5, D4 = 01 (для выбора режима изохронной передачи) и биты D3 - D0, хранящие значение номера конечной точки, содержащегося в дескрипторе конечной точки IN, полученном во время перечисления устройств.
- В регистр RxInterval конечной точки должен быть записан требуемый интервал передачи данных.
- В регистр RxMaxP конечной точки должен быть записан максимальный размер пакета (в байтах) для передачи. Это значение должно совпадать со значением в поле wMaxPacketSize стандартного дескриптора конечной точки.

  Соответствующий бит разрешения прерывания должен быть установлен в регистре IntrRxE (если для этой конечной точки требуется прерывание).
- Следующие биты регистра RxCSR должны быть установлены, как показано ниже:

| | | | |
|---|---|---|---|
| D15 | **AutoClear** | 0/1 | Установите значение 1, если требуется функция AutoClear |
| D14 | **AutoReq** | 0/1 | Установите значение 1, если требуется функция AutoRequest |
| D12 | **DisNyet** | 0 | Игнорируется в изохронном режиме |

**Процесс работы**

Работа начинается с того, что ЦП устанавливает ReqPkt (RxCSR.D5). Это приводит к тому, что контроллер отправляет IN токен в цель.

Когда принимается пакет, генерируется прерывание, по которому программное обеспечение может выгружать пакет из FIFO и сбрасывать бит RxPktRdy в регистре RxCSR (D0) так же, как и для поточной конечной точки Rx. Поскольку прерывание может происходить почти в любое время внутри кадра (/микрокадра), время запросов на выгрузку FIFO, вероятно, будет непостоянным. Если поток данных конечной точки уходит на внешнее оборудование, возможно лучше свести к минимуму потребность в дополнительной буферизации, дожидаясь конца каждого кадра перед выгрузкой FIFO. Это можно сделать, используя сигнал SOF_PULSE из контроллера, чтобы вызвать выгрузку

пакета данных. SOF_PULSE генерируется один раз на кадр (/микрокадр). Прерывания по-прежнему могут использоваться для сброса бита RxPktRdy в RxCSR.

**Обработка ошибок**

Если во время приема пакета возникает ошибка CRC или bit-stuff, пакет все равно будет размещен в FIFO, но будет установлен бит DataError (RxCSR.D3), чтобы указать, что данные могут быть повреждены.

### *Изохронная передача OUT*

Три дополнительных опции доступны для использования с конечной точкой Tx, в режиме хоста, для изохронной передачи данных:

- Двойная буферизация пакетов

  Двойная буферизация пакетов включается автоматически, если значение, записанное в регистре TxMaxP, меньше или равно половине размера FIFO, выделенного конечной точке. Когда опция включена, до двух пакетов можно сохранить в FIFO для ожидания передачи на периферию.
- AutoSet

  Если опция AutoSet включена, бит TxPktRdy (TxCSR.D0) будет автоматически устанавливаться, когда пакет размера TxMaxP байт будет загружен в FIFO.

**Настройка**

Перед началом изохронной передачи OUT:

- Должен быть установлен адрес целевой функции.
- В регистр TxType конечной точки контроллера, должны быть установлены биты D7, D6 для выбора рабочей скорости, биты D5, D4 = 01 (для выбора изохронной передачи), а в биты D3 - D0 установлено значение номера конечной точки, содержащегося в дескрипторе конечной точки OUT, полученном во время перечисления устройств.
- В регистр TxInterval конечной точки должен быть записан требуемый интервал передачи (обычно одна транзакция для каждого кадра/микрокадра).
- В регистр TxMaxP конечной точки контроллера должен быть записан максимальный размер пакета (в байтах) для передачи. Это значение должно совпадать со значением в поле wMaxPacketSize стандартного дескриптора конечной.
- Должен быть установлен соответствующий бит разрешения прерывания в регистре IntrTxE (если для этой конечной точки требуется прерывание)
- Следующие биты регистра TxCSR должны быть установлены, как показано ниже:

| | | | |
|---|---|---|---|
| D15 | **AutoSet** | 0/1 | Установите значение 1, если требуется функция AutoSet |
| D14 | **Mode** | 0/1 | Установите значение 1, чтобы гарантировать, что FIFO доступен (необходимо только в том случае, если FIFO используется совместно с конечной точкой Rx) |
| D12 | **FrcDataTog** | 0 | Игнорируется в изохронном режиме |

**Процесс работы**

Работа начинается с того, что ЦП записывает данные в FIFO, а затем устанавливает TxPktRdy (TxCSR.D0). Это вынуждает USB-контроллер отправить токен OUT, за которым последует первый пакет данных из FIFO.

Прерывание генерируется всякий раз, когда отправляется пакет, и ПО может по этому прерыванию загружать следующий пакет в FIFO и устанавливать бит TxPktRdy в регистре TxCSR (D0), аналогично поточной конечной точке Tx. Поскольку прерывание может происходить в любое время внутри кадра, то это может привести к неправильному времени запросов загрузки FIFO хоста при передаче. Если источник данных для конечной точки поступает с некоторого внешнего оборудования, может быть удобнее дождаться конца каждого кадра перед загрузкой FIFO, поскольку это минимизирует потребность в дополнительной буферизации. Это можно сделать, используя сигнал SOF_PULSE из контроллера, чтобы инициировать загрузку следующего пакета данных. SOF_PULSE генерируется один раз на кадр (/микрокадр). Прерывания могут использоваться для установки бита TxPktRdy в TxCSR.

Функция AutoSet может использоваться с изохронной конечной точкой Tx так же, как с поточной конечной точкой Tx. Однако, если данные поступают из источника не с постоянной скоростью, синхронизированной с часами фрейма, то размер пакетов будет увеличиваться или уменьшаться от кадра к кадру (или от микрокадра к микрокадру), чтобы соответствовать скорости источника данных. Это означает, что фактические размеры пакетов не всегда будут равны TxMaxP, что делает функцию AutoSet бесполезной.

### 15.8.7 *High-bandwidth (высокопропускные) изохронные передачи/ передачи по прерываниям*

Высокопропускные изохронные передачи и передачи по прерываниям используют те же самые протоколы, что и другие изохронные передачи, и передачи по прерываниям. Однако есть некоторые особенности проведения высокопропускных передач:

1 *Высопропускные изохронные передачи и передачи по прерываниям могут проводиться только если ядро контроллера было сконфигурировано с поддержкой таких передач.*

   Ядро также необходимо настроить таким образом, чтобы конечные точки, используемые для передач данных с высокой пропускной способностью, имели FIFO достаточного размера для хранения данных по меньшей мере одного пакета высокой пропускной способности (от 1 Кбайт и до 3 Кбайт).

2 *При настройке максимального размера пакета, обрабатываемого конечной точкой в регистре TxMaxP/RxMaxP, максимальное количество передач данных на микрокадр также должно быть установлено через биты D11 и D12 этого регистра.*

   Это максимальное количество передач (2 или 3) также представляет максимальное количество частей, на которые может быть разделен любой один «высокопропускной» пакет, что в свою очередь устанавливает максимальный размер пакета в 2 или 3 раза больше максимальной полезной нагрузки.

   *Примечание – Максимальная полезная нагрузка, которая может быть отправлена в любой транзакции, составляет 1 Кбайт.*

3 *При отправке пакетов бит TxPktRdy должен быть установлен программным обеспечением. Аналогично, при выгрузке пакетов из конечной точки RX, ПО должно очистить регистр RxPktRdy.*

   Функции AutoSet и AutoClear не могут использоваться для установки и очистки этих бит в транзакциях с высокой пропускной способностью.

4 *Передача пакетов в виде набора частей приводит к возникновению еще одного типа ошибок – передаче неполных пакетов.*

   Для конечных точек Tx эта проблема возможна только, если контроллер находится в режиме периферийного устройства и возникает, когда контроллер не может получить достаточное количество токенов IN от хоста для отправки всех частей пакета данных. Когда это происходит, контроллер устанавливает бит IncompTx в регистре TxCSR (D7).

   Для конечных точек Rx проблема возникает, когда PID из полученных частей пакета данных показывают, что одна или несколько частей пакета данных не

были получены. Когда это происходит, контроллер устанавливает бит IncompRx в регистре RxCSR (D8). В основном этот бит будет устанавливаться только тогда, когда ядро работает в периферийном режиме, но его также можно установить в режиме хоста, если (и только если) устройство, с которым контроллер обменивается данными, не отвечает в соответствии с протоколом USB.

#### 15.8.7.1 Подключение/Отключение

***В хост режиме***

Если контролер работает в хост режиме, ЦП начинает сессию установкой бита Session (DevCtl.D0). Затем питание подается на VBus, и ядро ожидает подключения устройства.

Когда устройство обнаружено, генерируется прерывание соединения (то есть IntrUSB.D4 устанавливается). Скорость подключенного устройства может быть определена путем чтения регистра DevCtl, где бит FSDev (D6) будет установлен для устройства с full-/high-speed, а бит LSDev (D5) будет установлен для low-speed устройства. Затем ЦП должен перезагрузить устройство.

ЦП должен поддерживать бит Reset, установленным 20 мс, чтобы гарантировать сброс цели. Затем он может начать перечисление устройств.

Если устройство отключено во время сеанса, будет создано прерывание Disconnect (то есть IntrUSB.D5 будет установлен).

***В режиме периферийного устройства***

Если USB-контроллер работает в периферийном режиме, прерывание не возникнет, когда устройство подключается к хосту.

Однако прерывание Disconnect (IntrUSB.D5) сгенерируется, когда хост завершит сеанс.

#### 15.8.7.2 Согласование роли хоста

Если USB-контроллер является устройством типа «A» (IDDIG низкий, B-Device (DevCtl.D7) = 0), он автоматически перейдет в хост режим при запуске сеанса.

Если USB-контроллер является устройством типа «B» (IDDIG высокий, B-Device (DevCtl.D7) = 1), он автоматически перейдет в режим периферийного устройства при запуске сеанса. Тем не менее ЦП может сделать запрос, чтобы контроллер стал хостом, установив бит Host Req в регистре DevCtl (D1). Этот бит можно установить одновременно с запросом на запуск сеанса, установив бит Session (DevCtl.D0) или в любой момент после начала сеанса.

Когда контроллер переходит в режим Suspend (нет активности на шине в течение 3 мс), то если бит Host Req установлен, он войдет в хост режим и начнет согласование хоста (как указано в дополнении USB On-The-Go) при помощи переключая TERMSEL, заставляя PHY отключить подтягивающий резистор на линии D. Это должно привести к тому, что устройство типа «A» переключится в режим периферийного устройства и подключит свой собственный подтягивающий резистор. Когда контроллер обнаружит это, он сгенерирует прерывание Connect (IntrUSB.D4), если оно включено. Он также установит бит Reset в регистре Power (D3), чтобы начать сброс устройства «A» (USB-контроллер автоматически запускает эту последовательность сброса, чтобы гарантировать, что перезапуск начнется в течение 1 мс после того, как устройство «A», подключит свой подтягивающий резистор). ЦП должен подождать не менее 20 мс, а затем сбросить бит Reset и пронумеровать устройство «A».

Когда устройство типа «B» на основе USB-контроллера закончило работу с шиной, ЦП должен перевести его в режим Suspend, установив бит Suspend Mode в регистр Power (D1). Устройство типа «A» должно обнаружить это и либо завершить сеанс, либо вернуться в режим хоста.

#### 15.8.7.3 Действия Vbus

Спецификация USB On-The-Go определяет ряд пороговых значений:

- VBus активен (VBus Valid) (напряжение от 4,4 до 4,75 В)
- Сеанс активен (Session Valid) для устройства типа «A» (напряжение от 0,8 до 2,1 В)
- Окончание сеанса (Session End) (напряжение от 0,2 до 0,8 В) которым должны удовлетворять устройства, участвующие в двухточечных связях.

***Действия контроллера в роли устройства типа «A»***

*Напряжение VBus больше напряжения активности VBus с сеансом, инициированным контроллером* (то есть Vbus [1: 0] (DevCtl. [D4: D3]) = 11b и установлен бит Session (DevCtl.D0)). Когда VBus становится выше, чем уровень активности, USB-контроллер входит в режим хоста и ожидает подключения устройства. После подключения он сгенерирует прерывание Connect (IntrUSB.D4). ЦП должен сбросить и перечислить подключенное устройство «B».

*Напряжение VBus больше напряжения активности VBus с сеансом, инициированным устройством типа «B»* (то есть Vbus [1: 0] (DevCtl. [D4: D3]) = 10b и сброшен бит сеанса (DevCtl.D0)). Когда VBus становится выше, чем уровень активности сеанса, контроллер сгенерирует прерывание Session Request (IntrUSB.D6). ЦП должен установить бит Session. Контроллер либо останется в режиме хоста, либо перейдет в режим периферии в зависимости от состояния подтягивающего резистора устройства «B». На выбранный режим будет указывать бит Host Mode (DevCtl.D2).

*Напряжение VBus меньше напряжения активности VBus, но бит Session все еще установлен* (то есть Vbus [1: 0] (DevCtl. [D4: D3]) ≠ 11b и установлен бит сеанса (DevCtl.D0)). Это указывает на проблему с уровнем мощности VBus. В этом случае контроллер автоматически прекратит сеанс и сгенерирует прерывание VBus Error (IntrUSB.D7).

***Действия контроллера в роли устройства типа «B»***

*Напряжение VBus больше напряжения активности сессии* (то есть Vbus [1: 0] (DevCtl. [D4: D3]) = 10b и сброшен бит Session (DevCtl.D0)). Это указывает на активность устройства типа «A». Контроллер установит бит Session и понизит уровень сигнал DPPULLDOWN, чтобы отключить подтягивающий резистор на линии D+.

*Напряжение VBus меньше напряжения активности сессии, но бит Session все еще установлен* (то есть Vbus [1: 0] (DevCtl. [D4: D3]) = 01b и установлен бит Session (DevCtl.D0)). Это означает, что устройство типа «A» потеряло питание (или отключилось). Контроллер сбросит бит Session (DevCtl.D0) и сгенерирует прерывание Disconnect (IntrUSB.D5). ЦП должен завершить сеанс.

*Напряжение VBus меньше напряжения активности сессии и установленным битом Host Request* (то есть Vbus [1: 0] (DevCtl. [D4: D3]) = 00b и установлен бит DevCtl.D1). Это условие смены Хоста. После 3 мс без активности на шине контроллер войдет в режим хоста и отключит подтягивающий резистор, чтобы начать согласование для роли хоста. Когда устройство «A» подключит собственный подтягивающий резистор, контроллер сгенерирует прерывание Connect (IntrUSB.D4). Он также установит бит Reset (Power.D3), чтобы начать сброс устройства «A». ЦП должен выждать не менее 20 мс, а затем сбросить бит Reset и перечислить устройство «A».

#### 15.8.7.4 Передача транзакций в качестве периферии

*Примечание* – Действия хоста показаны на белом фоне. Действия контроллера показаны на голубом.

### Транзакции управления

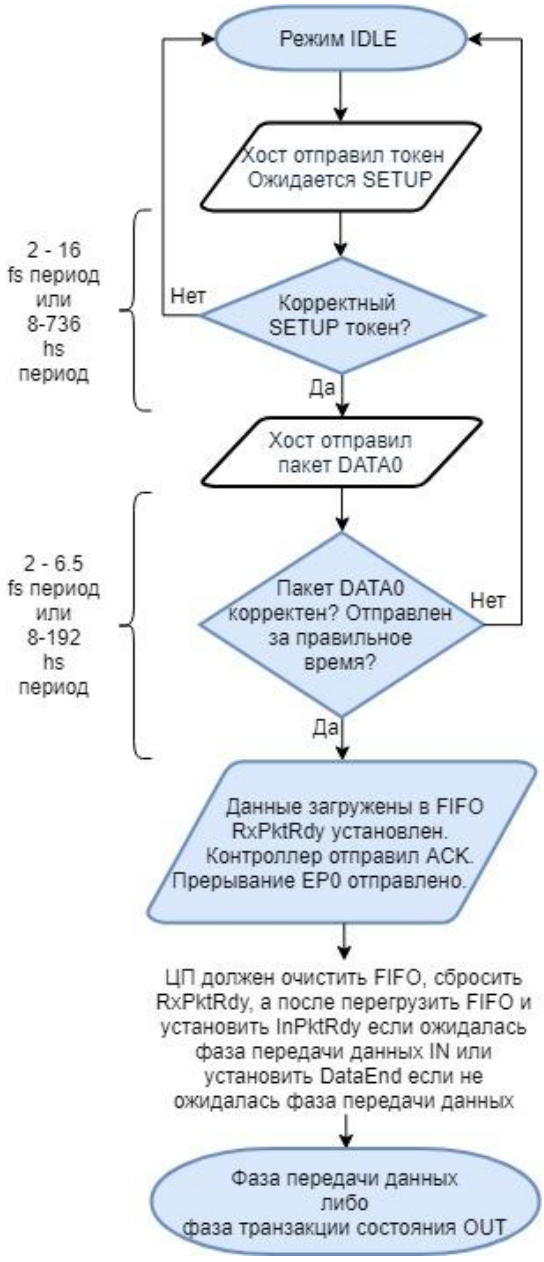



Рисунок 171 – Фаза установки

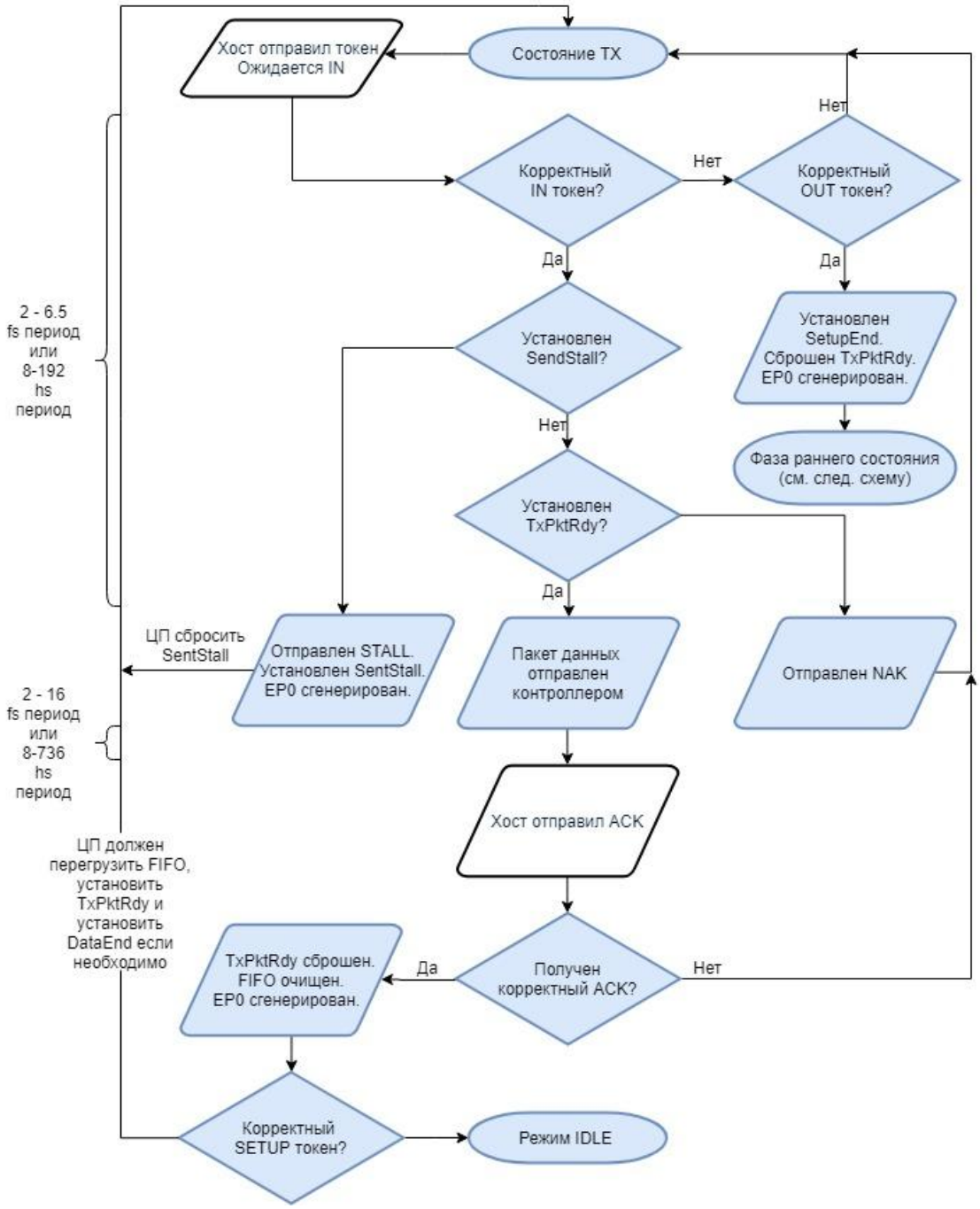



Рисунок 172 – Фаза передачи данных IN

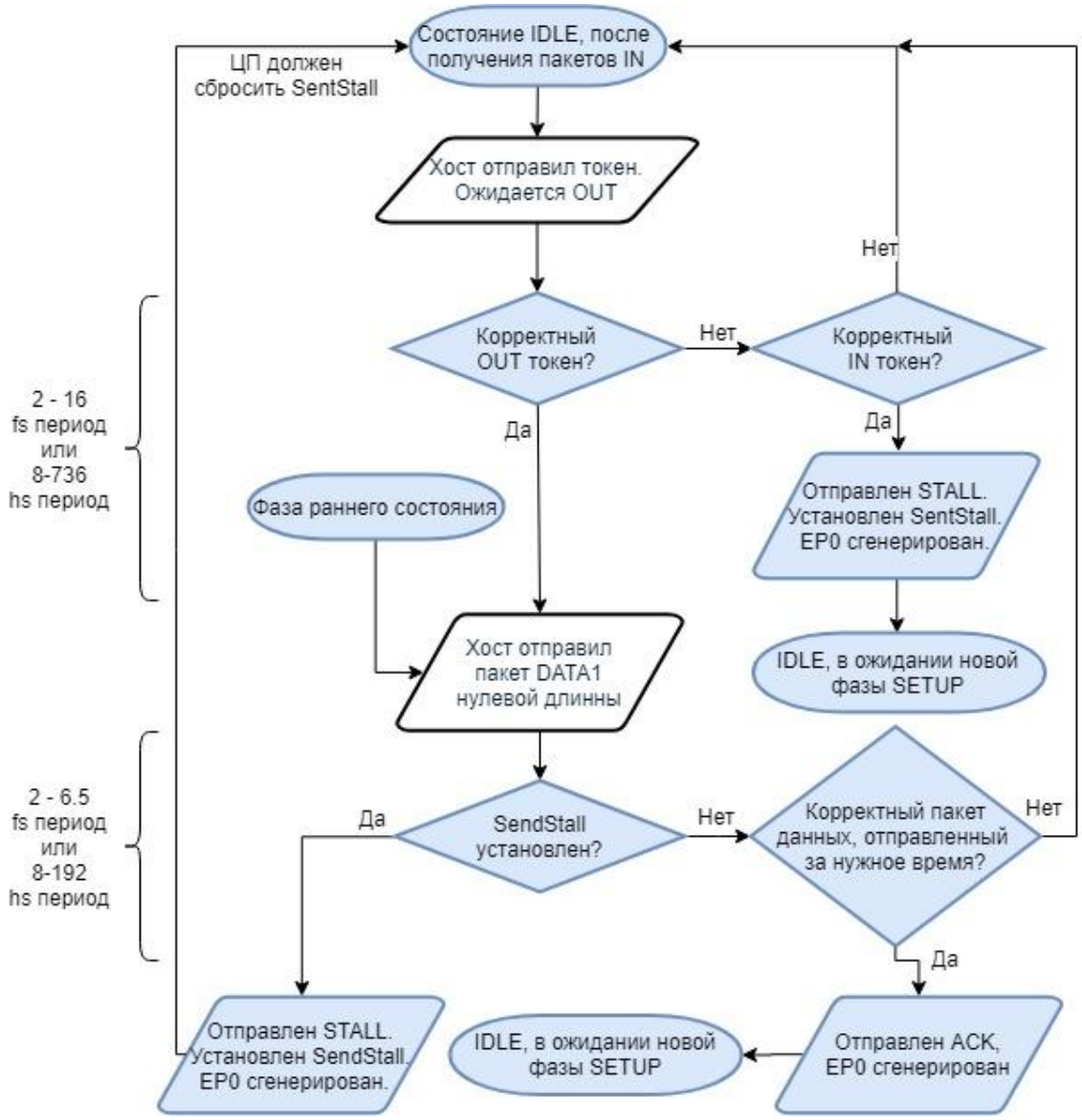



Рисунок 173 – Последующая фаза передачи состояния

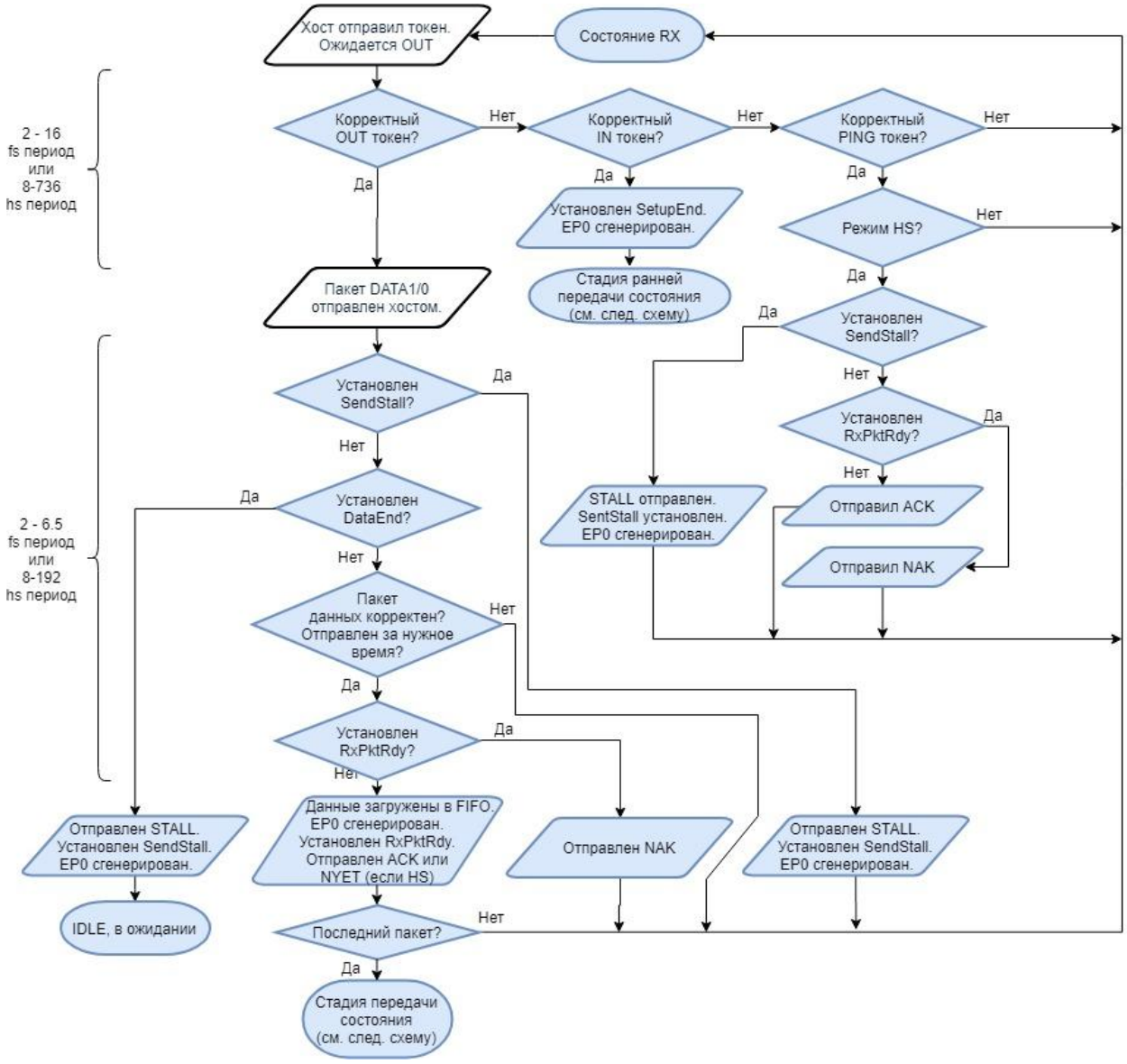



Рисунок 174 – Фаза передачи данных OUT

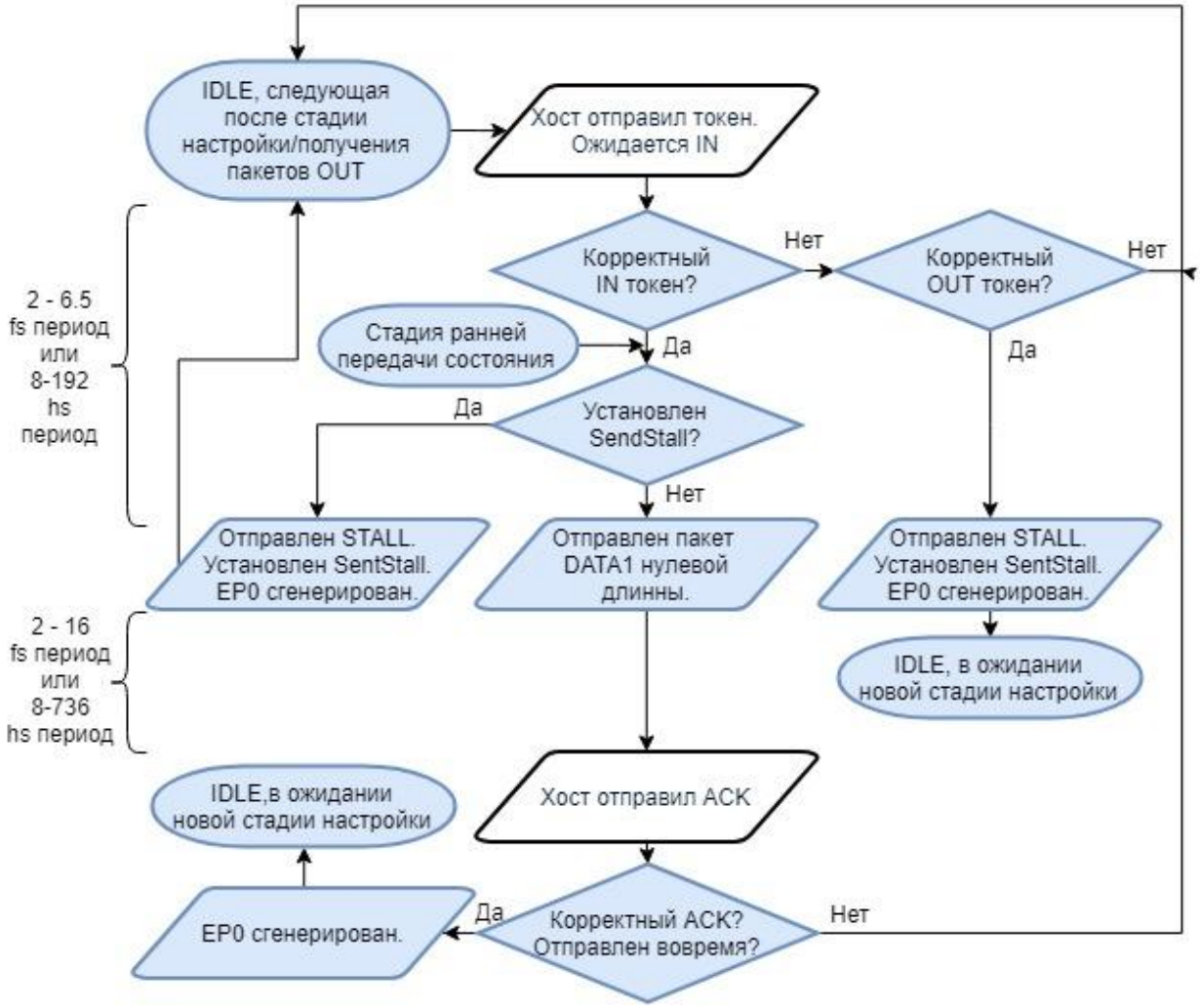



Рисунок 175 – Последующая фаза передачи состояния

### 15.8.8 *Поточная передача/низкопропускная передача по прерываниям*

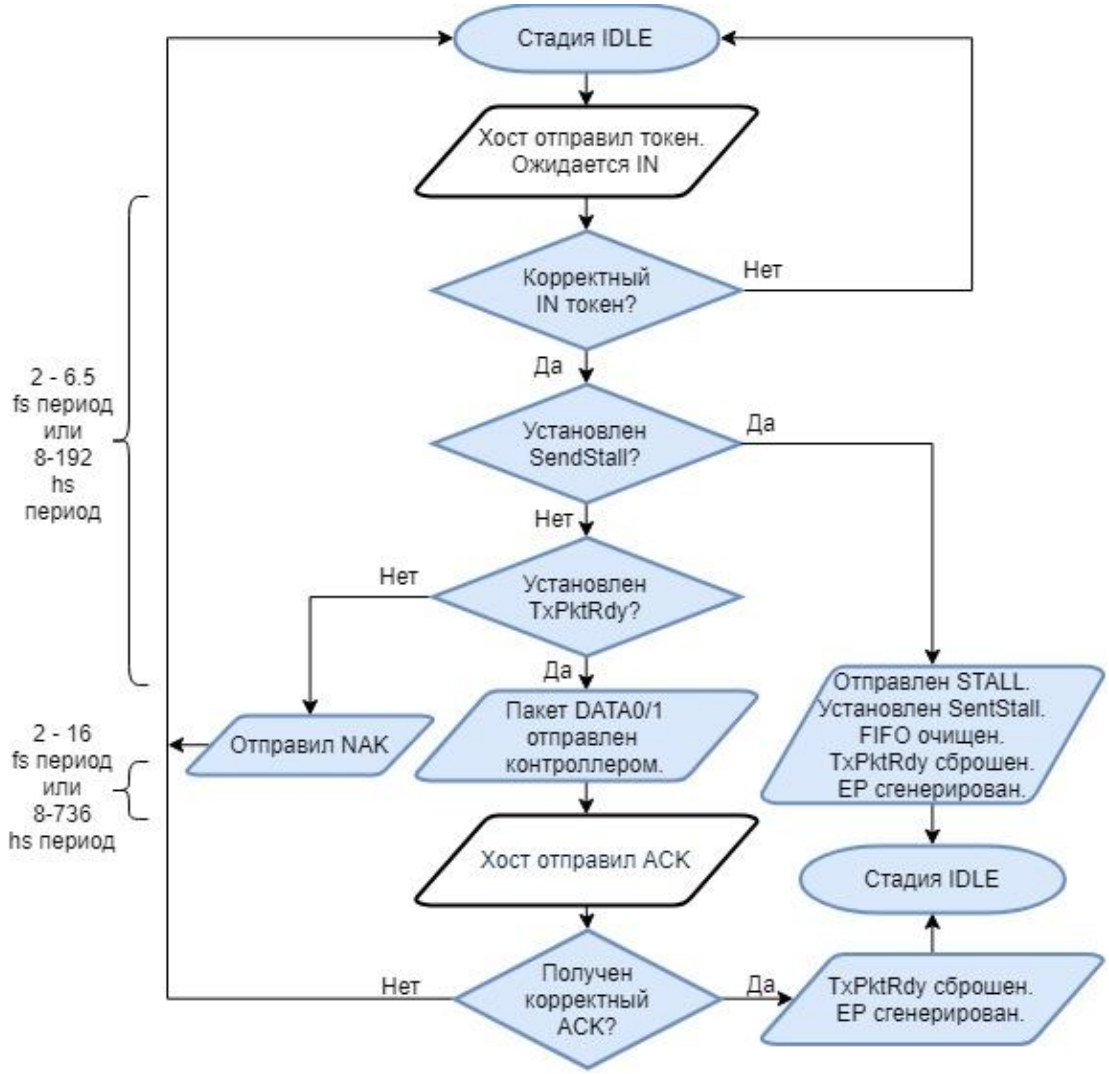



Рисунок 176 – Передача транзакций IN

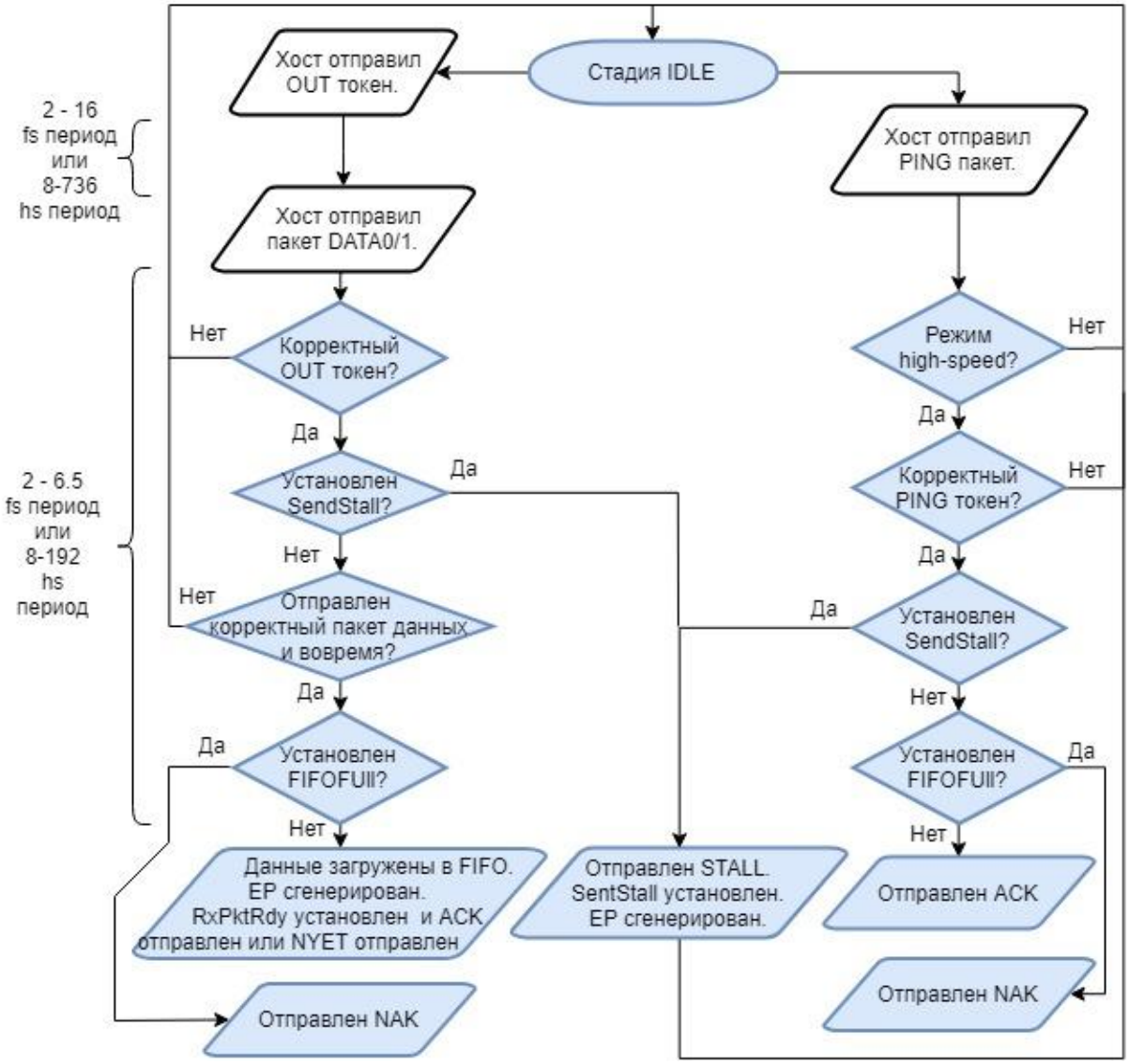



Рисунок 177 – Передача транзакций OUT

### 15.8.9 *High-speed/low-bandwidth изохронные передачи*

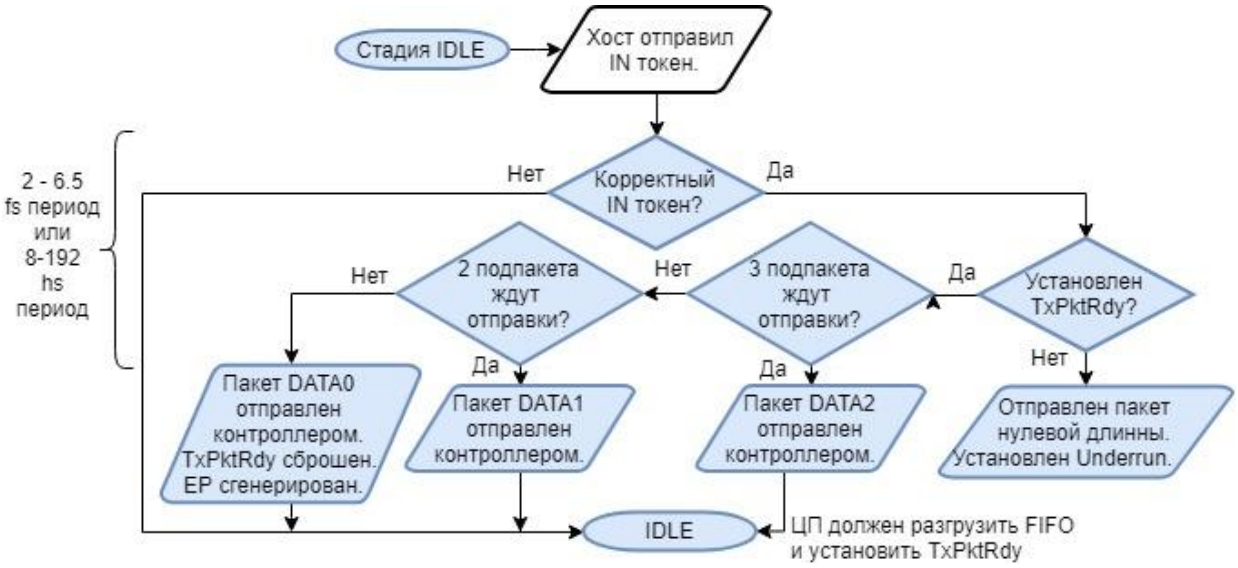



Рисунок 178 – Передача IN транзакций

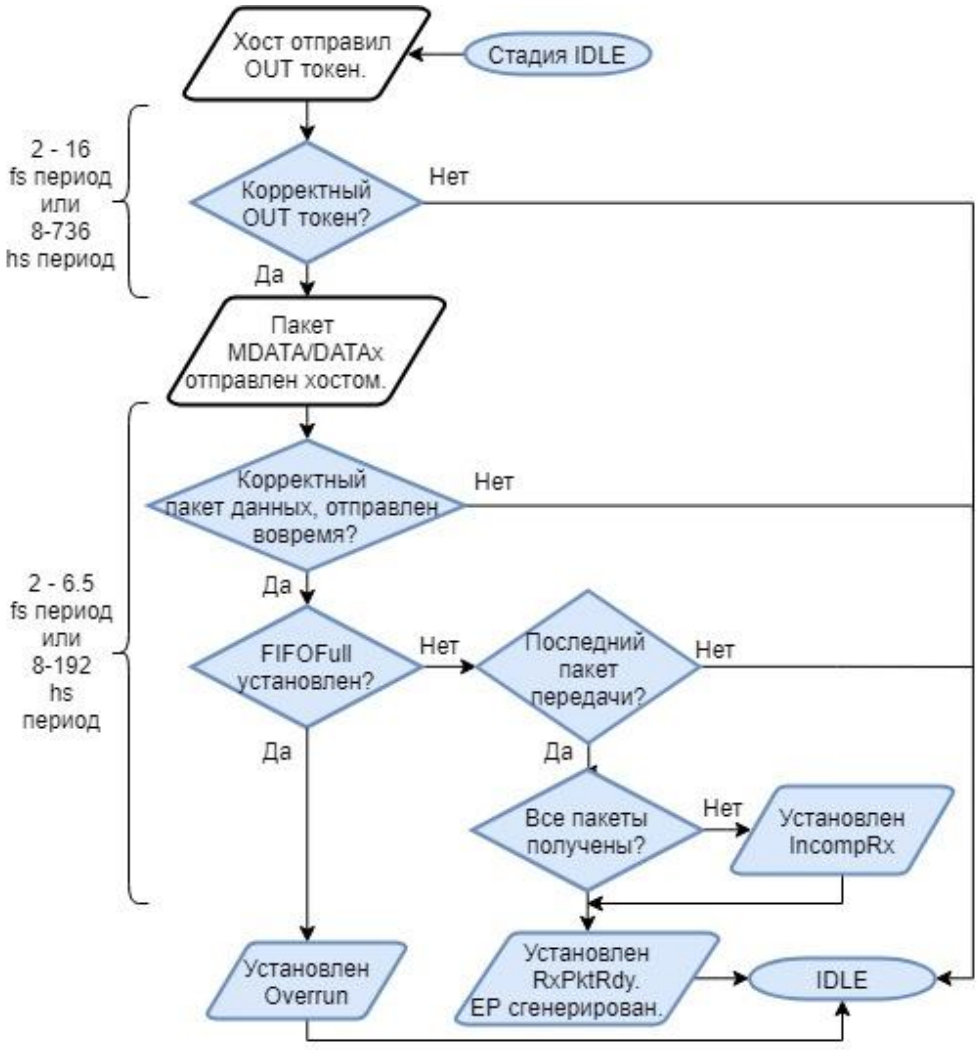



Рисунок 179 – Передача OUT транзакций

### 15.8.10 *Высокопропускные передачи (изохронные/по прерываниям)*

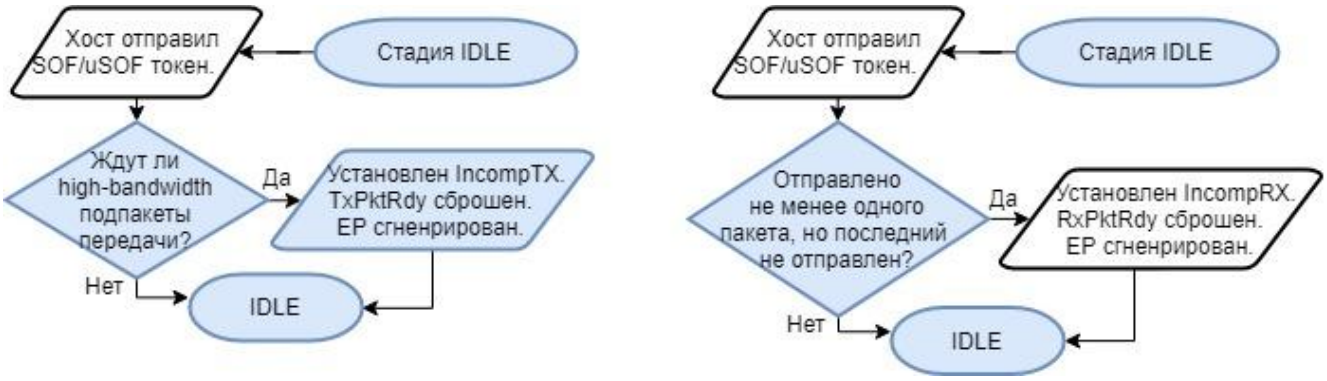



Рисунок 180 – Передача транзакций IN – слева, OUT –справа

### 15.8.11 *Передача транзакций со стороны хоста*

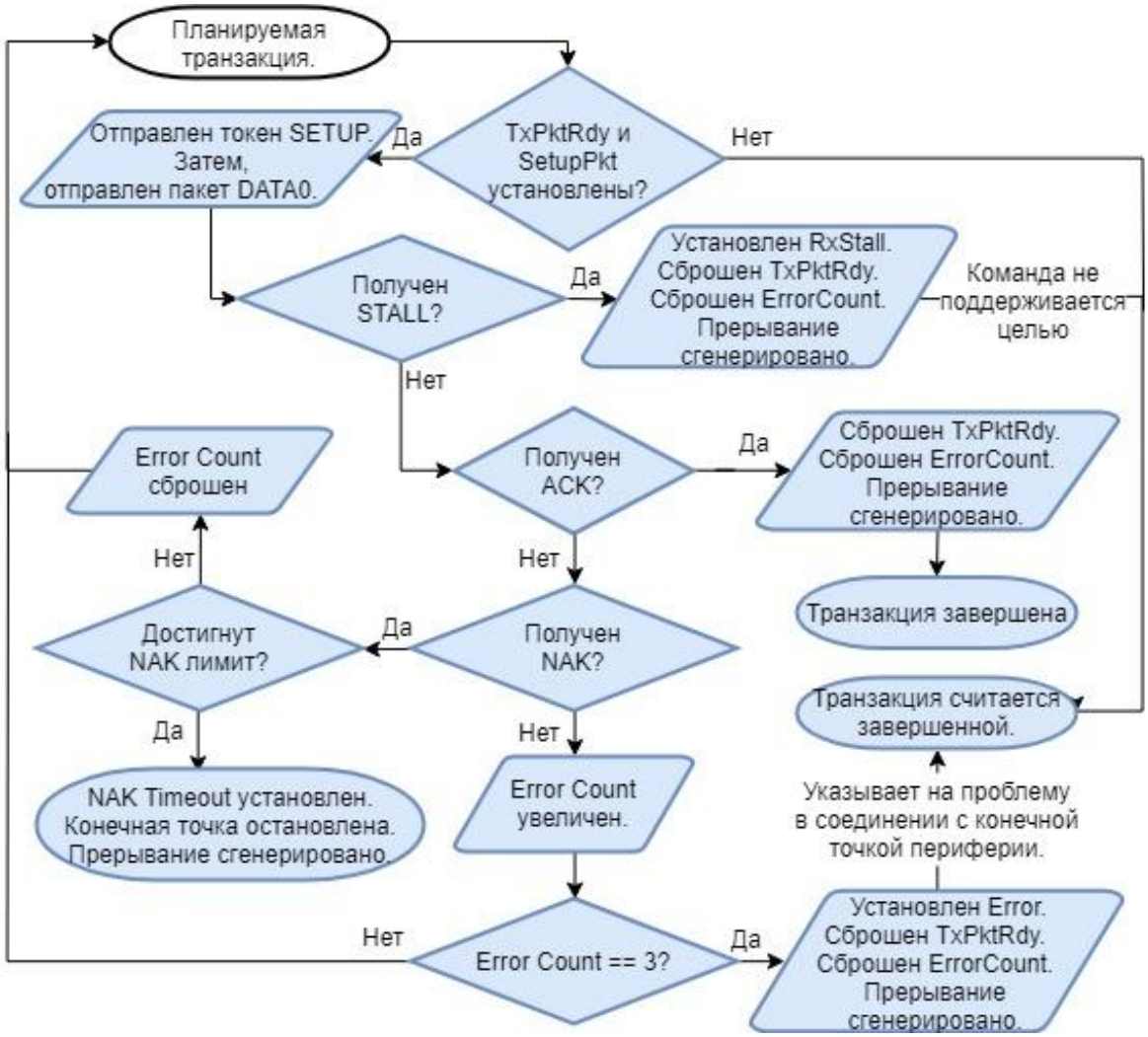



Рисунок 181 – Фаза установки

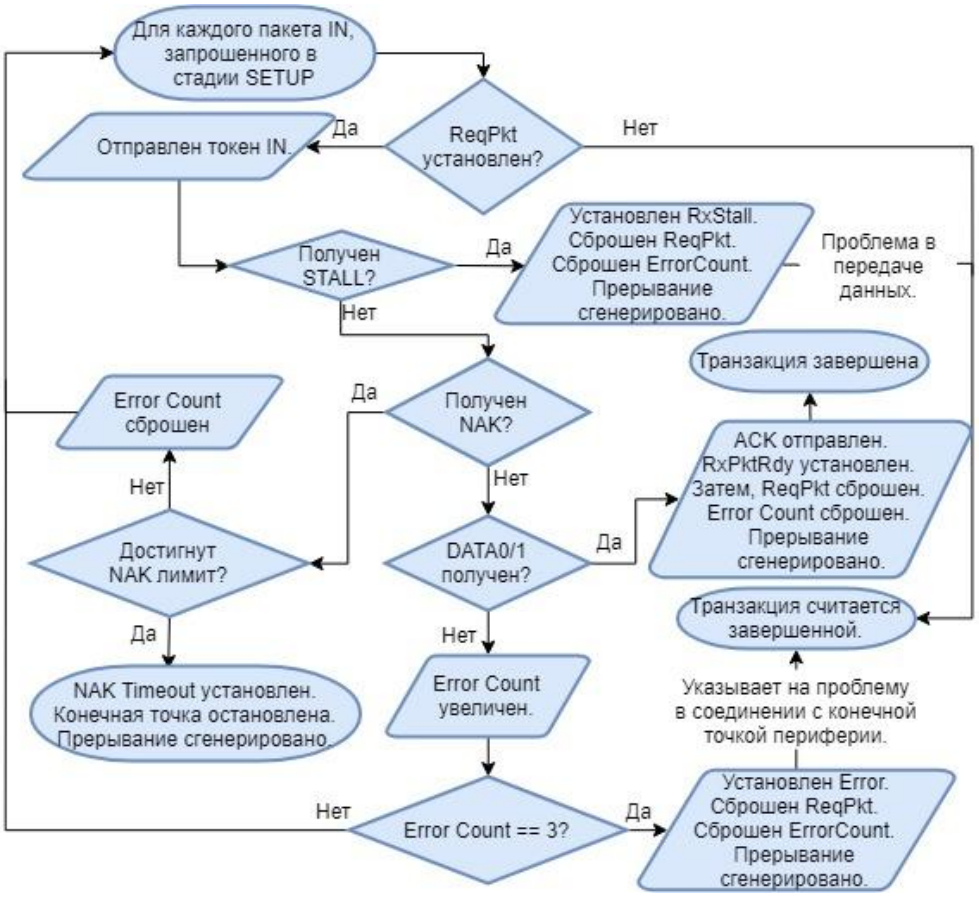



Рисунок 182 – Фаза передачи данных IN

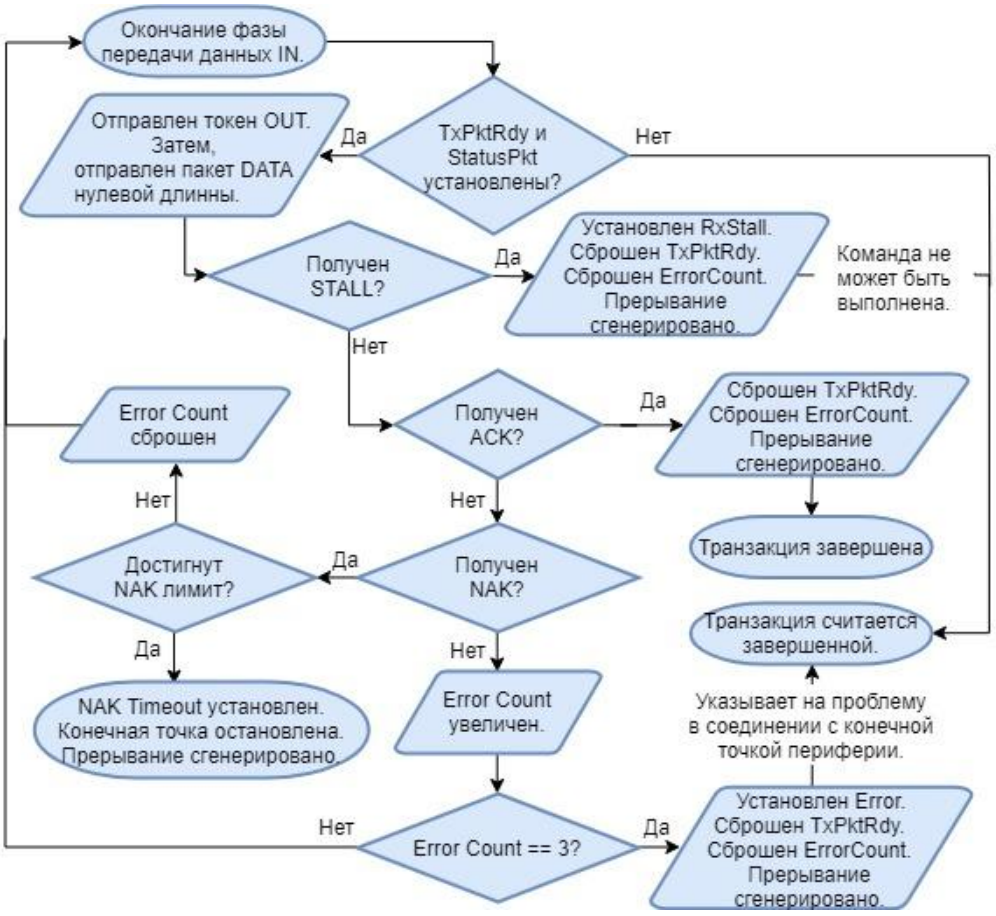



Рисунок 183 – Последующая фаза передачи состояния

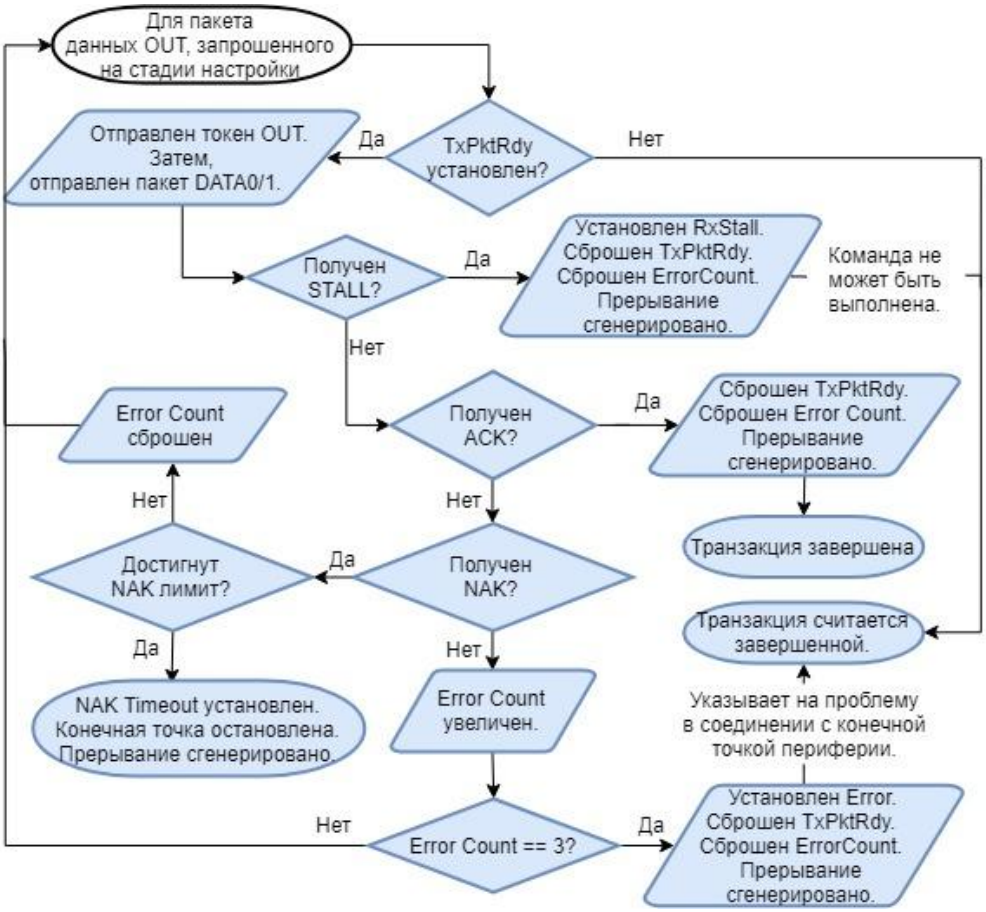



Рисунок 184 – Фаза передачи данных OUT

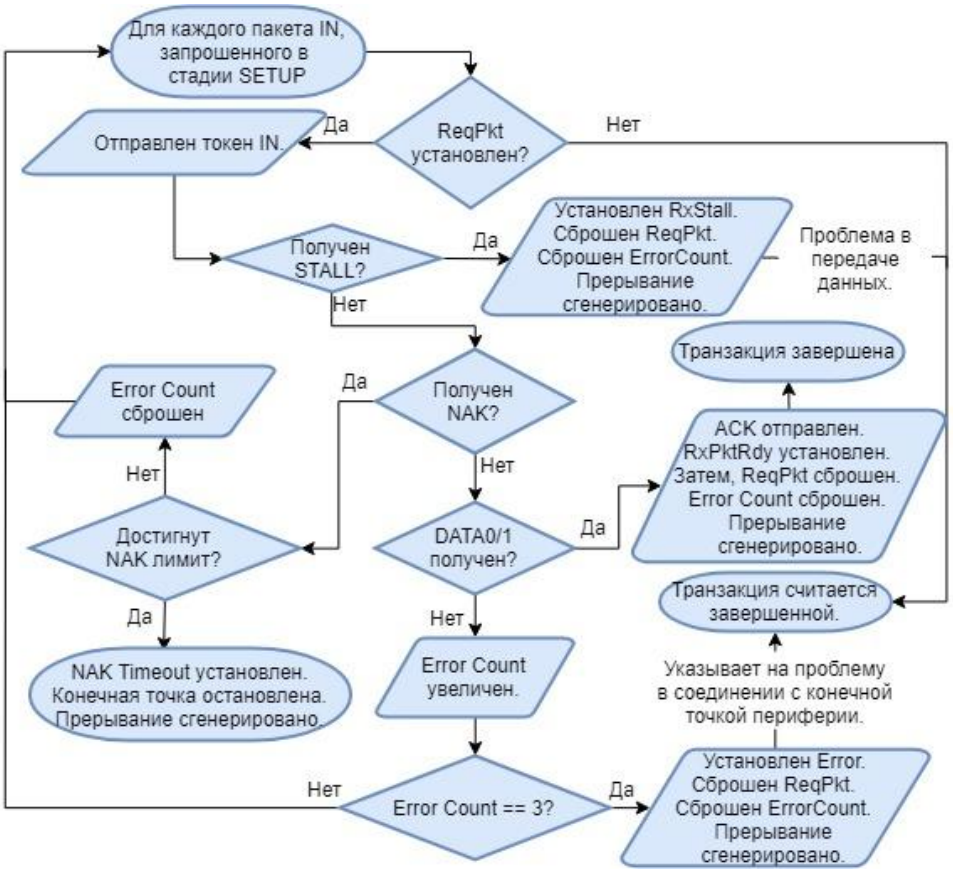



Рисунок 185 – Последующая фаза передачи состояния

### 15.8.12 *Поточная передача/низкопропускная передача по прерываниям*

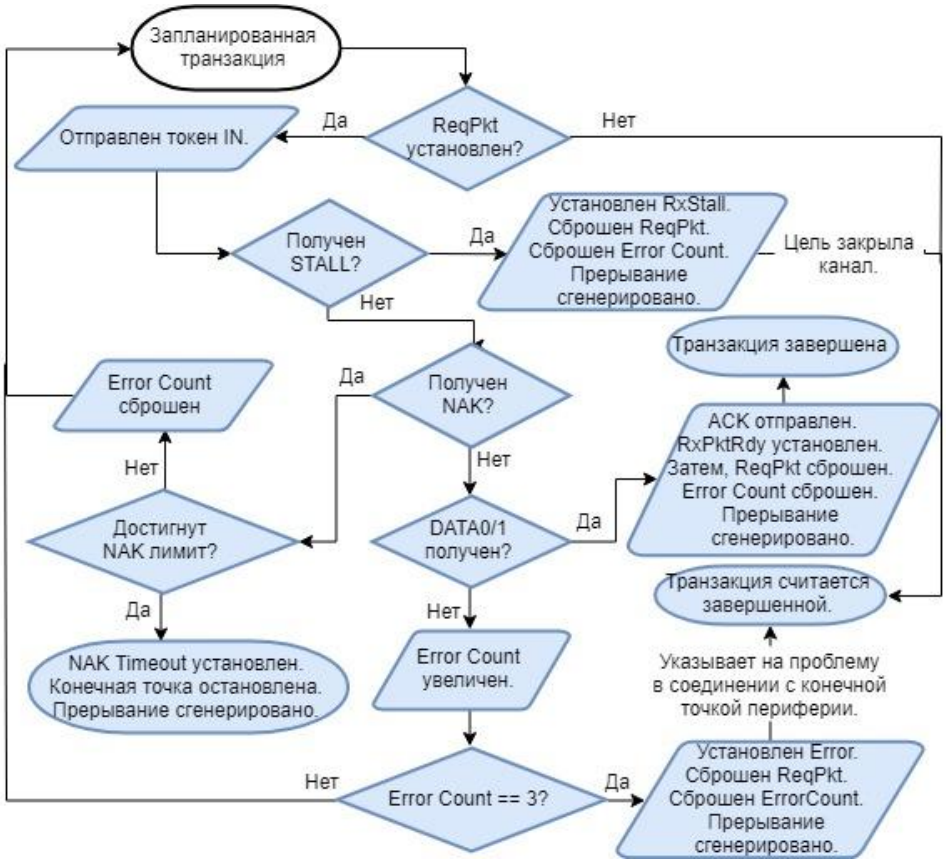



Рисунок 186 – Передача транзакций IN

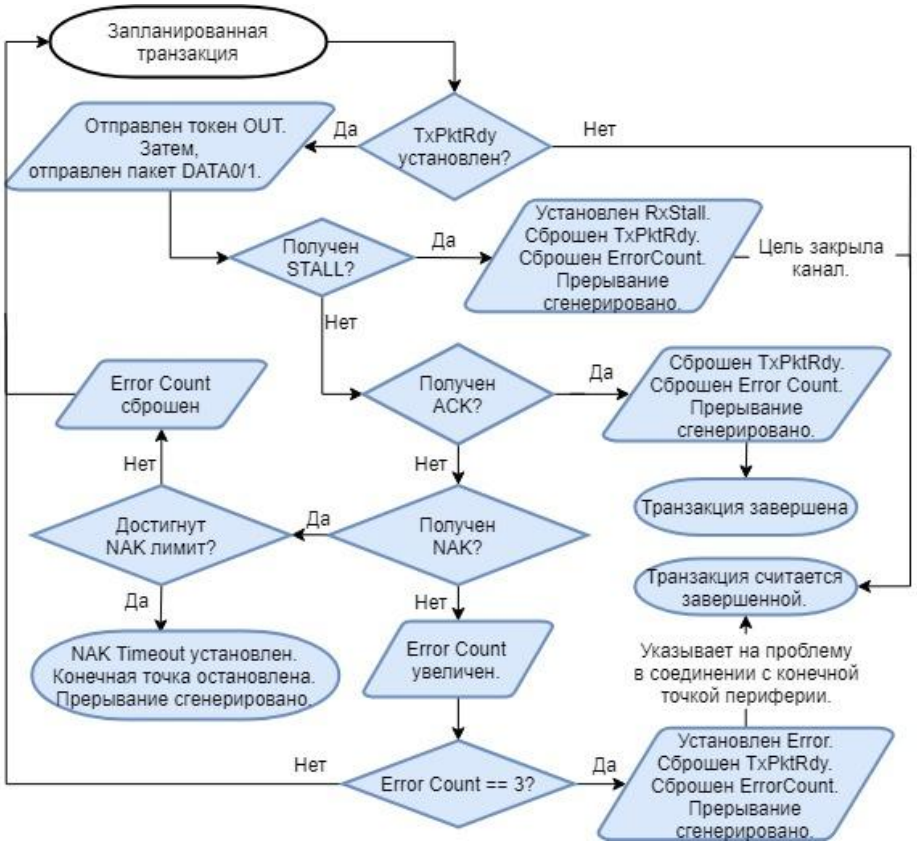



Рисунок 187 – Передача транзакций OUT

### 15.8.13 *Full-speed/низкопропускные передачи по прерываниям*

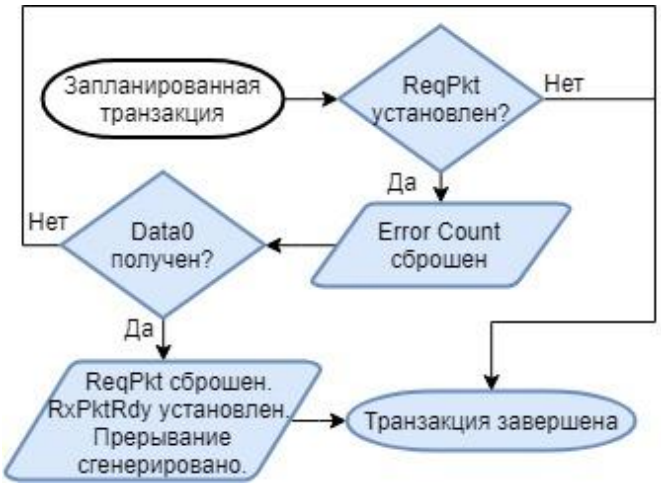



Рисунок 188 – Передача транзакций IN




Рисунок 189 – Передача транзакций OUT

### 15.8.14 *Высокопропускные передачи (изохронные/по прерываниям)*

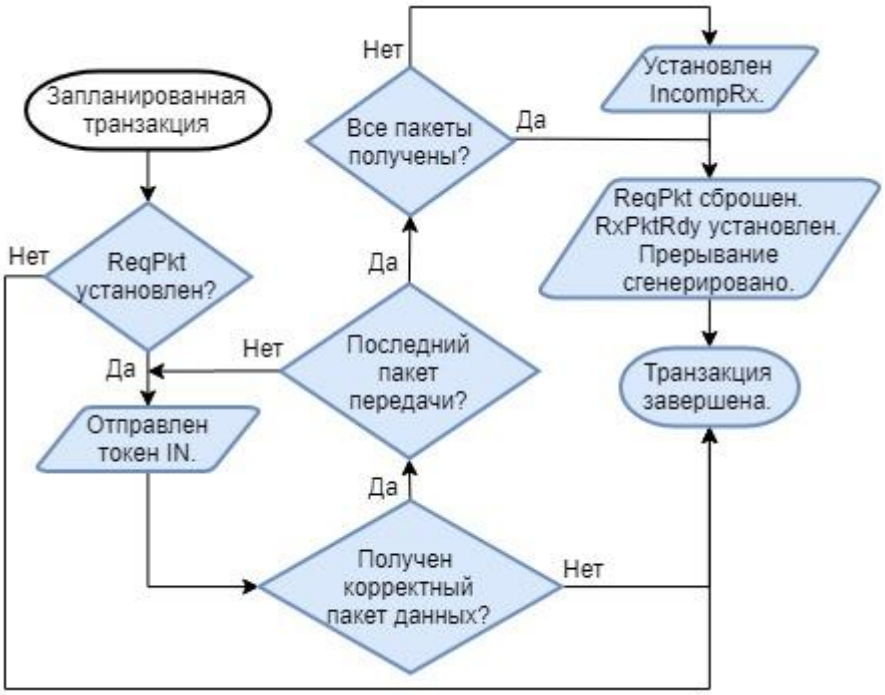



Рисунок 190 – Передача транзакций IN

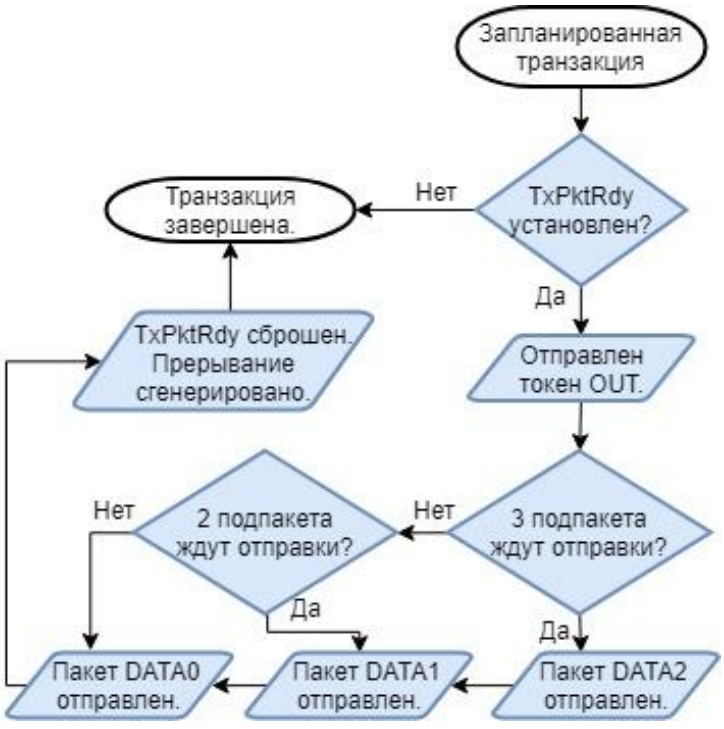



Рисунок 191 – Передача транзакций OUT

### 15.8.15 *Режим тестирования*

USB-контроллер поддерживает четыре режима тестирования USB 2.0, определенных для high-speed функций.

Режимы тестирования устанавливаются путем настройки регистра TestMode (адрес 0Fh). Обычно тестовый режим запрашивается хостом, отправкой запроса SET_FEATURE в конечную точку 0. Когда ПО получает запрос, оно должен дождаться завершения передачи конечной точки 0 (прерывание конечной точки 0, указывающее на то, что фаза согласования завершена), а затем настроить регистр TestMode.

***TEST SE0_NAK***

Чтобы войти в тестовый режим Test_SE0_NAK, программное обеспечение должно установить бит Test_SE0_NAK, записав 8'h01 в регистр TestMode. Тогда контроллер перейдет в режим, в котором он отвечает на любой действительный токен IN с помощью NAK.

***TEST_J***

Чтобы войти в тестовый режим Test_J, программное обеспечение должно установить бит Test_J, записав 8'h02 в регистр TestMode. Тогда контроллер перейдет в режим, в котором он передает сигнал логического уровня J на шину.

***TEST_K***

Чтобы войти в тестовый режим Test_K, программное обеспечение должно установить бит Test_K, записав 8'h04 в регистр TestMode. Тогда контроллер перейдет в режим, в котором он передает сигнал логического уровня K на шину.

***TEST_PACKET***

Чтобы выполнить тест Test_Packet, программное обеспечение должно сначала записать стандартный тестовый пакет (показан ниже) в FIFO конечной точки 0 и установить бит InPktRdy в регистре CSR0 (D1). Затем он должен записать 8'h08 в регистр TestMode, чтобы войти в тестовый режим Test_Packet.

Далее представлен 53-байтовый тестовый пакет, который необходимо загрузить (все байты в шестнадцатеричном виде). Тестовый пакет должен быть загружен только один раз; контроллер продолжит повторную отправку тестового пакета без какого-либо дополнительного вмешательства со стороны программного обеспечения.

00 00 00 00 00 00 00 00
00 AA AA AA AA AA AA AA
AA EE EE EE EE EE EE EE
EE FE FF FF FF FF FF
FF FF FF FF FF FF 7F BF DF
EF F7 FB FD FC 7E BF DF E
F F7 FB FD 7E

Эта последовательность данных определена в спецификации универсальной последовательной шины Revision 2.0, раздел 7.1.20. Контроллер добавит DATA0 PID в начало последовательности и CRC в конец.

## 15.9 Контроллер SDIO (SDIO_CNTR)

Интерфейс SDIO обеспечивает связь между периферийной шиной APB и картами MultiMediaCards (MMC), SD картами и SDIO картами.

Спецификации на MultiMediaCard доступны на сайте MultiMediaCard Association, публикуются техническим комитетом MMCA. Спецификации на карты памяти SD и SD I/O доступны на сайте SD card Association.

Основные функции интерфейса SDIO:

- Полное соответствие спецификации MultiMediaCard System Specification Version 4.2. Поддержка трех различных режимов шины данных: 1-бит (по умолчанию), 4-бит и 8-бит;
- Полное соответствие предыдущим версиям спецификаций MultiMediaCards (совместимость снизу-вверх);
- Полное соответствие спецификации SD Memory Card Specifications Version 2.0;
- Полное соответствие спецификации SD I/O Card Specification Version 2.0: поддержка двух различных режимов шины данных: 1-бит (по умолчанию) и 4-бит;
- Скорость передачи данных до 50 МГц для режима 8-бит;
- Сигнал включения передачи данных и команд для управления внешними приёмопередатчиками.

*Примечания*

1 Интерфейс SDIO не поддерживает режим передачи данных, совместимый с интерфейсом SPI.

2 Протокол передачи карты памяти SD представляет собой расширенный протокол MultiMediaCard, как определено в спецификации MultiMediaCard V2.11. Некоторые команды, необходимые для управления устройствами с SD памятью, не поддерживаются ни картами SD I/O, ни комбинированными картами с узлами I/O. Некоторые из этих команд не находят применения в SD I/O устройствах, например, такие как команды стирания, и тем самым они не поддерживаются протоколом SDIO. В дополнение, некоторые команды отличаются от интерфейсов карт SD и SD I/O, и таким образом они не поддерживаются протоколом SDIO. Для более подробной информации см. спецификацию SD I/O card Specification Version 1.0. При помощи шины MultiMediaCard/SD карты подсоединяются к контроллеру. Текущая версия интерфейса SDIO поддерживает только одну карту SD/SDIO/MMC4.2 единовременно и несколько карт MMC4.1 или предыдущих версий.

## 15.10 Контроллер портов ввода-вывода (PORT_CNTR)

Порты ввода-вывода предназначены для приема и формирования внешних электрических сигналов. Для каждого вывода микроконтроллера может быть назначена одна из 16 функций. В качестве функции вывода выступает использование данного вывода как ввода-вывода сигналов периферийных контроллеров, таких как внешняя шина, UART, SSP и так далее, либо полностью программное управление – пользовательский порт. Для некоторых выводов может быть назначена аналоговая функция, например, канал ввода аналогового сигнала в АЦП или выдачи сигнала с ЦАП.

Порты ввода-вывода могут использоваться пользовательскими задачами без дополнительного программного драйвера. Для того, чтобы избежать взаимовлияния работы различных задач друг на друга, например, при возникновении перехода с одной задачи на другую в момент выполнения операции чтения-модификации-записи, все регистры порта имеют виртуальный регистр установки битов в «1» и виртуальный регистр сброса битов в 0. Таким образом, чтобы установить какой-либо бит регистра в «1» нет необходимости выполнять операции чтения и модификации, достаточно записать «1» в соответствующий бит регистра с префиксом «S», и это взведет соответствующий бит. Для сброса этого бита необходимо записать «1» в регистр с префиксом «C». Запись «0» в биты никакого эффекта не несет. При чтении из всех регистров, кроме RXTX, возвращается состояние триггеров. При чтении регистра RXTX возвращается значение с выводов микросхемы.

Каждый вывод порта может служить источником прерываний по уровню от портов. Время удержания активного уровня на входе порта, для формирования запроса на прерывание, должно быть не менее двух тактов ядра. Схема формирования запроса прерывания представлена на рисунке 192.

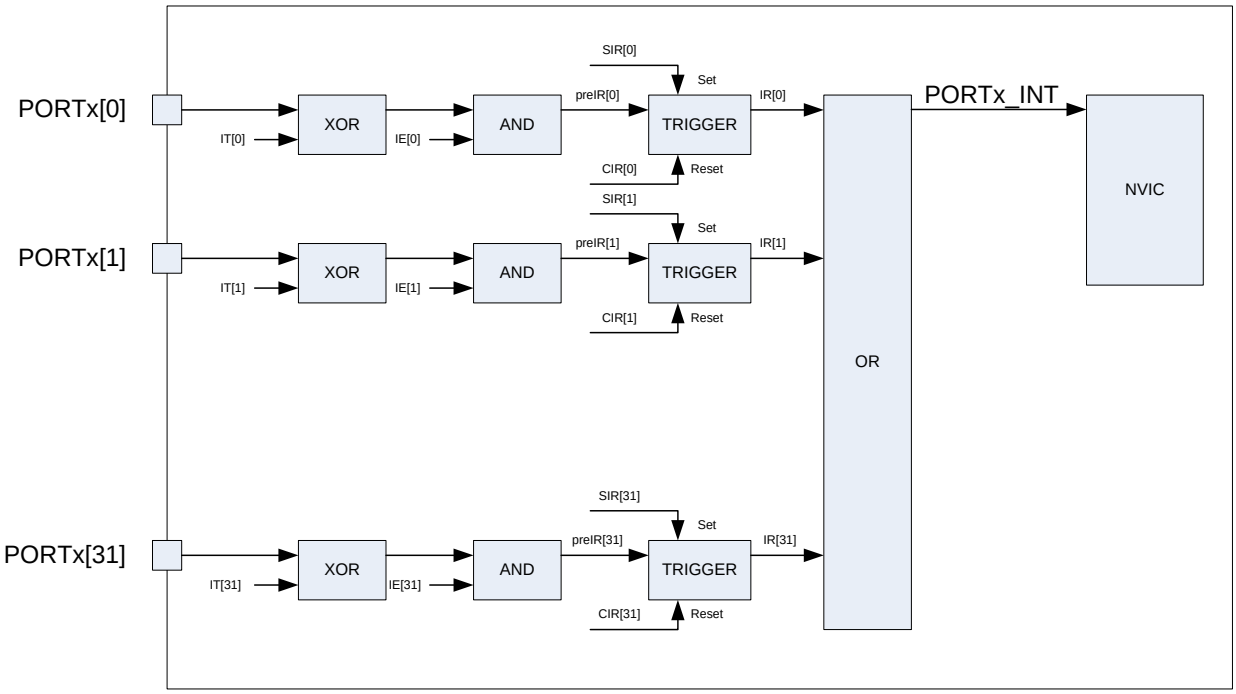



Рисунок 192 – Схема формирования запроса прерывания

В регистре IR запроса прерывания порта факт прерывания фиксируется только при разрешении запроса. При этом запрещение запроса прерывания IE не сбрасывает флаг IR, если он появился до запрета. Сброс запроса прерывания возможен только сбросом соответствующего разряда регистра IR. Формирование прерываний, их обработка и сброс возможны только при включенном тактировании порта.

Для портов ввода-вывода реализована защита от «перегрузки». Если вывод выдает логическую «1», но на линии уровень сигнала соответствует логическому «0», вырабатывается флаг защиты от перегрузки HCUR. При выдаче логического «0» механизм защиты аналогичен. При возникновении флага защиты может быть сгенерировано

прерывание от порта. Для исправления ситуации необходимо программно перевести вывод в третье состояние или, в случае, когда перегрузки кратковременны, сбросить флаг записью «1» в соответствующий бит регистра HCUR.

# 16 Контроллер CRC (CRC_CNTR)

Блок CRC (cyclic redundancy check) используется для вычисления CRC для 8-, 16- и 32-разрядных данных с программируемым полиномом.

Основные особенности блока вычисления CRC:

- использование полинома CRC32-IEEE-802.3 (0x04C1_1DB7) по умолчанию;
- возможность задания полинома и его размера (7, 8, 16, 32 бита);
- обработка 8-, 16- и 32-разрядных данных;
- программируемое начальное значение CRC;
- единый 32-битный регистр ввода/вывода данных;
- наличие входного буфера, позволяющего избежать простоя шины во время вычисления CRC;
- вычисление CRC за четыре такта APB (PCLK) для 32-битного слова данных;
- регистр общего назначения размером 8 бит (может использоваться для временного хранения);
- возможность изменения порядка бит входных/выходных данных.

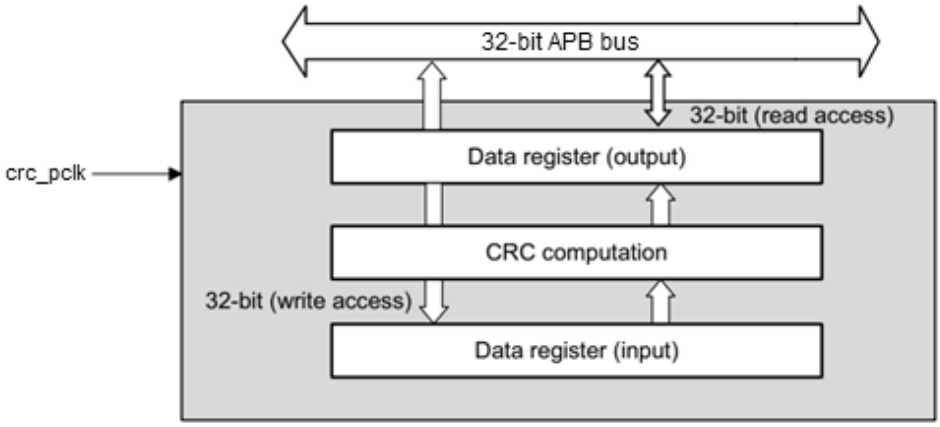



Рисунок 193 – Структурная схема блока CRC

В блоке реализован 8-битный регистр общего назначения **IDR**, который может быть использован для временного хранения байта информации (например, для хранения временного значения, связанного с вычислением CRC) и не сбрасывается битом **RESET** регистра управления **CR**.

Блок вычисления CRC содержит входной буфер, который позволяет избежать простоя шины во время вычисления CRC и осуществлять запись новых данных без ожидания готовности обработки ранее записанных данных.

Регистр данных **DR** имеет доступ как на чтение (считать рассчитанное значение), так и на запись. Каждая операция записи в регистр данных создает комбинацию предыдущего значения CRC (хранится в **DR**) и нового значения. Вычисление CRC выполняется для всего 32-битного слова данных или побайтно в зависимости от формата записываемых данных. Регистр **DR** имеет доступ: 32 бита, 16 бит с выравниванием по правому краю и 8 бит с выравниванием по правому краю.

Длительность расчета CRC зависит от размера данных:

- четыре такта PCLK для 32-битных данных;
- два такта PCLK для 16-битных данных;
- один такт PCLK для 8-битных данных.

Размер данных можно динамически регулировать, чтобы минимизировать количество обращений для записи заданного количества байт. Например, CRC для 5 байтов может быть вычислен с записью слова, за которым следует запись байта.

Для управления различными схемами порядка следования байтов, порядок бит входных данных может быть изменен. Операция реверсирования может выполняться по 8 бит, 16 бит и 32 бита в зависимости от бит **REV_IN[1:0]** в регистре **CR**.

Например, входные данные 0x1A2B_3C4D используются для вычисления CRC как:

- 0x58D4_3CB2 с реверсированием по 8 бит;
- 0xD458_B23C с реверсированием по 16 бит;
- 0xB23C_D458 с реверсированием по 32 бита.

Порядок бит выходных данных также может быть изменен установкой бита **REV_OUT** регистра **CR**. Операция реверсирования выполняется побитово: например, выходной результат 0x1122_3344 будет конвертирован в 0x22CC_4488.

Калькулятор CRC может быть инициализирован программируемым значением с помощью управляющего бита **RESET** в регистре **CR**.

Начальное значение CRC можно запрограммировать с помощью регистра **INIT**. Регистр **DR** автоматически инициализируется при доступе на запись в регистр **INIT**.

Коэффициенты полинома полностью программируются через регистр **POL**, а размер полинома может быть сконфигурирован на 7, 8, 16 или 32 бита путем программирования битов **POLYSIZE[1:0]** в регистре **CR**. Четные полиномы не поддерживаются.

Если размер полинома меньше 32 бит, то вычисленное значение CRC автоматически выравнивается по правой границе регистра **DR**.

Чтобы получить надежное вычисление CRC, изменение значения или размера полинома на лету не может выполняться во время вычисления CRC. Перед изменением полинома необходимо остановить вычисление CRC или считать значение регистра **DR**.

# 17 Контроллер тригонометрических преобразований (CORDIC_CNTR)

## 17.1 Основные функции блока

- Блок осуществляет конвейерную обработку входящих данных;
- Время обработки данных от входа до появления результата – 35 тактов;
- Блок может принимать данные на вход каждый такт;
- Блок работает в трех форматах входных-выходных данных:
  - 32 числа с плавающей точкой;
  - 16 битные числа с плавающей точкой;
  - 32 битные целые знаковые числа. Для данного режима входной диапазон ограничен 24 битами (то есть от 0xFF000000 до 0x00FFFFFF);
- Блок реализует метод двойного повтора с компенсацией деформации;
- Блок работает в двух режимах поворота:
  - Поворот входного вектора на заданный угол;
  - Поворот входного вектора до оси OX по часовой стрелке с определением начального угла и длины вектора (деформированное значение);
- Блок осуществляет поворот входного вектора в диапазоне углов (-360; +360) градусов;
  - 360 градусов для режимов с плавающей точкой соответствует значению 1.0;
  - 360 градусов для целочисленных режимов соответствует значению 0x01000000.

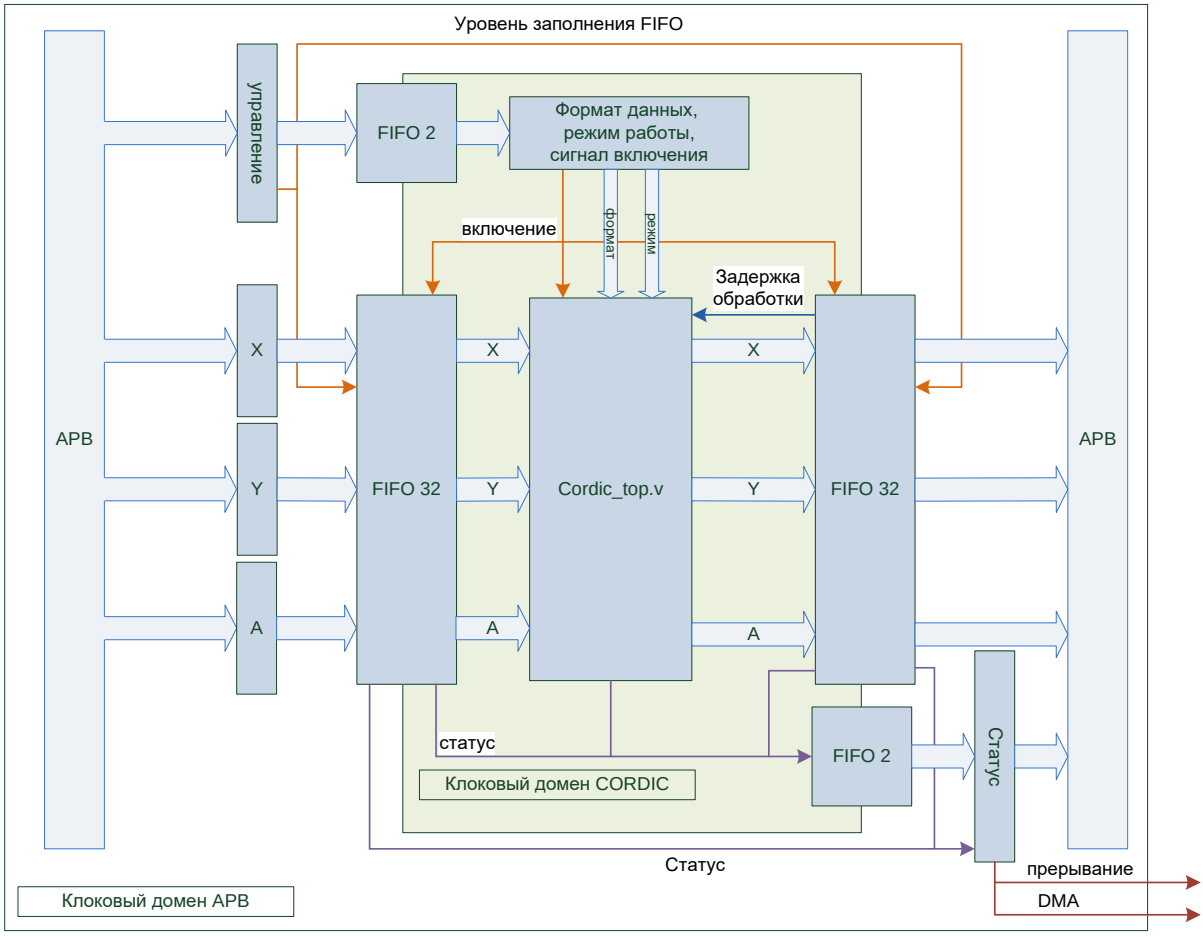



Рисунок 194 – Структурная схема блока тригонометрических преобразований

## 17.2 Входные параметры и результат работы модуля

Входные-выходные данные модуля в режиме поворота на заданный угол:

- Входные:
  - x – координата вектора (32 бита, формат определяется настройками);
  - y – координата вектора (32 бита, формат определяется настройками);
  - α – угол поворота вектора (32 бита, формат определяется настройками);
- Выходные:
  - x – координата повернутого вектора (32 бита, формат определяется настройками);
  - y – координата повернутого вектора (32 бита, формат определяется настройками);
  - α – ошибка поворота, величина «недоворота» вектора (32 бита, формат определяется настройками).

Входные-выходные данные модуля, режим поворота до оси OX:

- Входные:
  - x – координата вектора (32 бита, формат определяется настройками);
  - y – координата вектора (32 бита, формат определяется настройками);
  - α – начальный угол, значение будет прибавлено к вычисленному углу (32 бита, формат определяется настройками);
- Выходные:
  - x – деформированная длина исходного вектора, коэффициент деформации 0.607252935 (32 бита, формат определяется настройками);
  - y – ошибка поворота, величина отклонения вектора по оси OY (32 бита, формат определяется настройками);
  - α – угол отклонения исходного вектора от оси OX с учетом заданного начального угла. (32 бита, формат определяется настройками).

## 17.3 Функционирование модуля

Работа модуля осуществляется на асинхронной по отношению к системной частоте, что позволяет ускорить производимые модулем вычисления. Для работы блока необходимо разрешить выдачу частоты APB интерфейса (регистр PER_CLK) и настроить собственную тактовыую частоту модуля CRDC_CLK (регистр CRDC_CLK).

Блок осуществляет прием и выдачу данных по шине AMBA APB как ведомое устройство, его регистры расположены в памяти периферийных устройств микроконтроллера.

На входе и выходе блок имеет FIFO для данных, глубиной 32 отсчета.

Для загрузки пары координат обрабатываемого вектора в FIFO блок имеет регистры IN_X, IN_Y, для загрузки угла - регистр IN_A. При осуществлении записи в регистр угла IN_A происходит автоматическая загрузка всех рех регистров в очередь FIFO.

Другой вариант загрузки данных в FIFO контроллера – запись в регистр последовательного заполнения FIFO SEQ. При последовательной записи в этот регистр осуществляет поочередная запись во все три входных регистра. При этом в регистре IN_ADDR указывается текущий регистр для записи. Регистр IN_ADDR инкрементируется автоматически, но при необходимости требуемый адрес может быть жестко задан записью в регистр. При значении IN_ADDR == «11» данные из регистров заносятся в FIFO.

Чтение результата осуществляется аналогично, либо через чтение регистров OUT_X, OUT_Y и OUT_A, либо чтением регистра OUT_SEQ. В регистре OUT_ADDR при этом отображается адрес текущего регистра для чтения. Регистр OUT_ADDR инкртементируется автоматически. При осуществлении чтения при OUT_ADDR == 11 происходит обновление данных. Значение адреса для чтения может задаваться записью в регистр OUT_ADDR.

Модуль имеет два сигнал запроса тразакций DMA. Один - модуль готов принять данные (входное FIFO не полно и нет активной записи в него), другой - модуль готов отдать

данные (выходное FIFO не пусто). Работа модуля в режиме DMA наиболее удобна с использованием регистров послежловательного доступа.

Включение модуля осуществляется битом CRD_EN регистра CRD_CNTR, после выхода модуля в рабочий режим выставляется флаг CRD_RDY регистра CRD_STATUS. Также в регистре CRD_STATUS содержаться флаги, информирующие об активности модуля в текущий момент времени и заполненности FIFO входных значений и результатов.

Модуль может осуществлять преобразование в двух режимах работы, которые могут быть настроены битом IN_MODE регистра CRD_CNTR:

Режим поворота входного вектора (x : y) на угол α. На выходе получается новый, повернутый вектор (x : y), угол α обнуляется на заданный угол;

Режим поворота входного вектора (x : y) до оси OX (определение угла по координатам). На выходе получается исходный угол вектора α, и длина исходного вектора в координате x (деформированное значение);

Полем IN_FORMAT регистра CRD_CNTR формат принимаемых данных может быть задан как 32 бита с плавающей запятой, 16 бит с плавающей запятой, либо 32-битный целочисленный.

При появлении на входе всех трех значений (x, y, α), модуль загружает данные и начинает их обработку, через 35 тактов модуль выдаст результат обработки. При этом будут выполнены следующие действия:

- преобразование числа с плавающей точкой в 25-битное знаковое целое (для форматов с плавающей точкой);
- выравнивание порядков переменных X, Y (для форматов с плавающей точкой);
- вычисление дополнительных углов для двойного поворота (α + β) и (α - β);
- нормировка квадранта (для двух параллельных CORDIC). Для режима поворота вектора до оси OX, выполняется до чеырех поворотов на 90 градусов, пока вектор не окажется в первом квадранте. Для режима поворота вектора на угол, выполняется до четырех поворотов на 90 градусов, пока угол поворота не окажется в диапазоне ± 90 градусов;
- 25 последовательных операций на двух параллельных CORDIC;
- усреднение полученных данных с двух CORDIC;
- восстановление порядка (для форматов с плавающей точкой);
- выдача результата на выход.

## 17.4 Прерывания модуля ускорителя тригонометрических преобразований

Модуль имеет 9 источников прерываний, и регистр маски для выбора какие из источников формируют общий запрос к контроллеру прерываний процессора:

- обработка входных данных закончена;
- входное FIFO пусто;
- входное FIFO заполнено ниже указанного уровня;
- входное FIFO заполнено не полностью;
- входное FIFO переполнено;
- выходное FIFO не пусто;
- выходное FIFO заполнено выше указанного уровня;
- выходное FIFO заполнено полностью;
- произошло чтение из пустого выходного FIFO.

Все события прерываний отображаются в регистре флагов прерываний CRD_INTF.

Источники формирования сигнала прерывания к процессорному могут быть разрешены путем установки соответсвующего бита в регистре разрешения прерываний CRD_INTE.

# 18 Защищенный модуль криптографических вычислений

Микроконтроллер имеет в составе защищенную систему на основе вычислительного ядра Cortex-M0, набор аппаратных блоков криптоускорителей и средств защиты от несанкционированного доступа к исполняемому коду и данным. Структурная схема защищенной подсистемы представлена на рисунке 195.

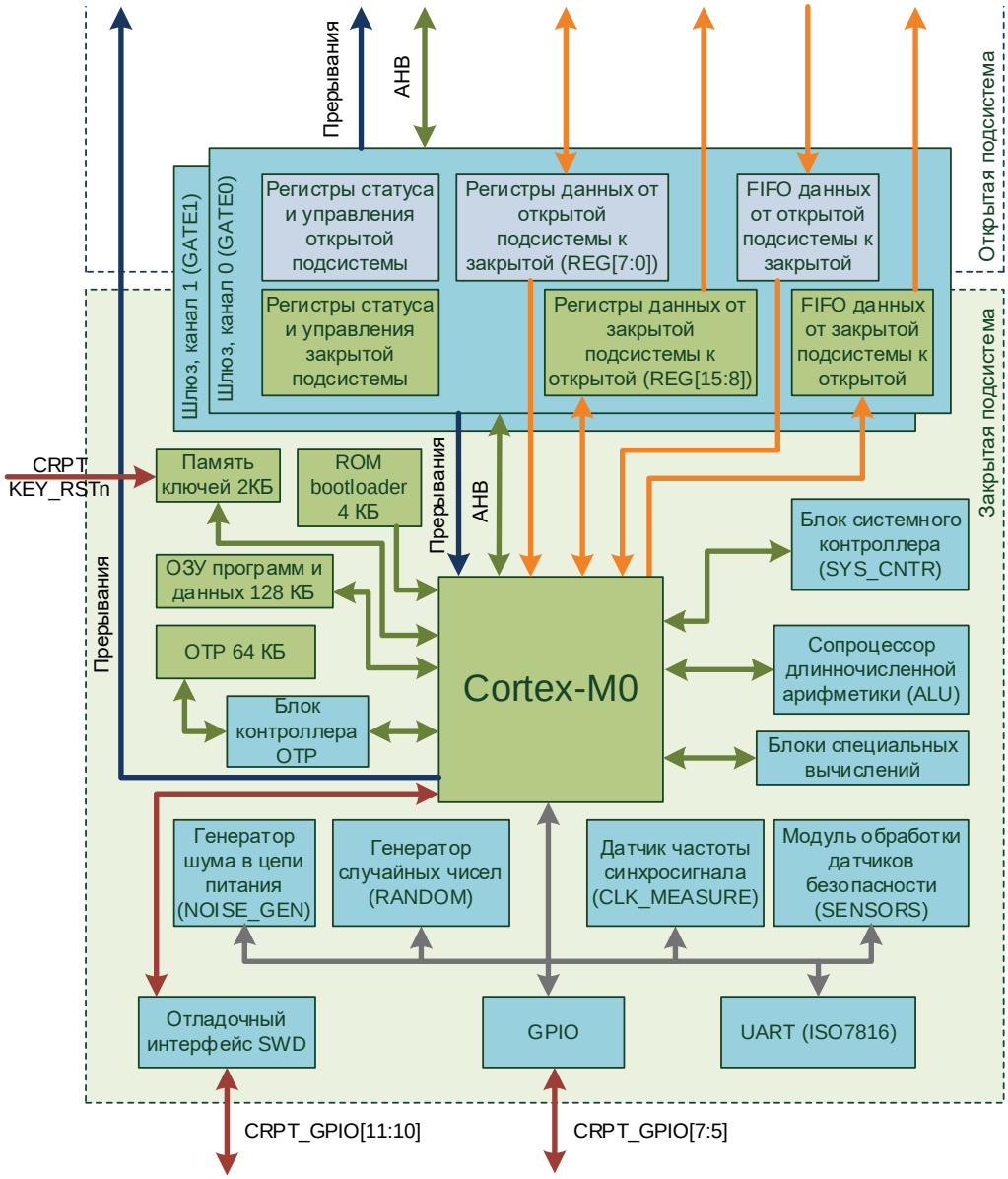



Рисунок 195 – Структурная схема закрытой области микроконтроллера

В закрытой подсистеме микроконтроллера расположено изолированное ядро Cortex-M0. Ядро имеет собственный загрузчик, собственную изолированную память программ и данных, свою собственную энергозависимую память для хранения ключей и прочих секретных данных. Этот процессор выполняет алгоритмы специализированной криптобиблиотеки и имеет модули ускорения криптосчета.

Для обмена данными по инициативе открытого ядра в модуле имеется шлюз. Со стороны открытого ядра он виден как обычная периферия. Внутри имеются 2 FIFO для обмена данными и регистры статуса и управления. Открытое ядро имеет возможность читать статус, задавать управляющий регистр, записывать данные в FIFO входных данных и читать данные из FIFO выходных данных.

Обмен данными через шлюз поддерживается в криптобиблиотеке, поэтому никакие данные из открытого ядра не могут попасть напрямую в закрытое (кроме обмена

по DMA, инициированного закрытым ядром). Также никаким образом открытое ядро не может заставить закрытое ядро выполнить неразрешенные команды. Для ускорения реакций и упрощения обмена имеется набор прерываний (шлюз – открытое ядро, шлюз – закрытое ядро, закрытое ядро – открытое ядро)

Для запуска микроконтроллера в режиме работы с защищенной подсистемой необходимо установить вывод PROTECT_MODE PE[23] в состояние логической единицы на время загрузки микроконтроллера. При этом выводы порта PD[4:1] будут использованы как выводы UARTRX, UARTTX, CLK_UART, RST_KEY защищенной подсистемы. Другие выводы, требуемые для работы защищенной подсистемы, могут быть определены программно ядром Cortex-M4 стандартными средствами переназначения функций портов.

## 18.1 Алгоритм загрузки защищенной подсистемы

Защищенная часть не имеет встроенной FLASH-памяти. Загрузка ОЗУ-программ подсистемы осуществляется каждый раз со стороны открытой подсистемы в зашифрованном виде.

## 18.2 Процессорное ядро

Подробное описание процессорного ядра Cortex-M0 приведено в документе Cortex-M0 Devices Generic User Guide[1].

## 18.3 Контроллер шлюза передачи данных между подсистемами (GATE_CNTR)

Блок шлюза передачи данных между подсистемами микроконтроллера обеспечивает безопасный перенос данных между клоковыми доменами открытого и защищённого ядер. Блок имеет два подключения к AHB-Lite, размер данных 32 бита, размер адреса определяется количеством используемых адресов в модуле. Одно подключение выполняется со стороны открытого ядра, другое - со стороны защищенного. Тактовые сигналы AHB-подключений со стороны открытого и защищенного ядер различные и не синхронные. Модуль имеет в своем составе FIFO передачи данных от открытого ядра к защищённому, и FIFO передачи данных от защищённого ядра к открытому с шириной данных 32 бита и объёмом 16 слов.

Модуль формирует запросы DMA (только на стороне открытого ядра) по одному из следующих событий:

- входное FIFO (от открытого ядра к защищённому) не полно;
- выходное FIFO (от защищённого ядра к открытому) не пусто.

Модуль формирует запрос прерываний от FIFO по одному из следующих событий:

- входное FIFO пусто;
- входное FIFO заполнено ниже указанного уровня;
- входное FIFO не заполнено;
- запись в заполненное входное FIFO;
- выходное FIFO не пусто;
- выходное FIFO заполнено выше указанного уровня;
- выходное FIFO полно;
- чтение из пустого выходного FIFO.

Флаги прерываний FIFO снимаются автоматически по исчезновению события, кроме флагов «чтения из пустого FIFO» и «запись в заполненное FIFO». Данные флаги снимаются по записи 1 в соответствующий бит статусного регистра, запись 0 не должна менять состояние флагов.

Модуль имеет 8 выходных регистров данных для записи и 8 входных для чтения. Данные, записанные в выходной регистр одной стороны, правильно переходят между

[1] Перевод документа высылается по запросу, отправленному на support@milandr.ru.

клоковыми доменами и попадают во входной регистр другой стороны. Управление переходом обеспечивается дополнительным регистром. Входные регистры доступны только для чтения.

В регистре управления переходом для каждого выходного регистра есть биты занятости регистра междоменным переходом. Данный бит выставляется в 1 в момент записи в регистр и снимается при переносе данных в другой клоковый домен. Запись в занятый регистр игнорируется.

Модуль формирует запрос прерываний от регистров:

- изменился 0 входной регистр;
- изменился 1 входной регистр;
- …
- изменился 7 входной регистр;
- запись в занятый 0 выходной регистр;
- запись в занятый 1 выходной регистр;
- …
- запись в занятый 7 выходной регистр.

Сброс флагов прерываний «Изменился X входной регистр» производится автоматически по чтению из регистра. Флаги «запись в занятый X выходной регистр», сбрасываются записью 1 в регистр статуса, в соответствующий бит.

Отображение флагов прерывания от FIFO и регистров производится в отдельных регистрах, доступных только для чтения, кроме процедуры сброса флагов через запись единицы в соответствующий бит флага.

### 18.4 Память ключей

Специальная защищенная память ключей выполняется на регистрах и имеет асинхронный механизм удаления данных, то есть данные могут удаляться в отсутствии тактового сигнала памяти и работы процессора. Память расположена в отдельном домене с батарейным питанием и сохраняет при выключеном питании процессора.

Источники сигнала удаления данных:

- внешняя ножка процессора;
- внутренний регистр процессора (программное управление сбросом);
- сигнал от модуля датчиков безопасности.

Память имеет встроенный механизм защиты данных, основанный на скремблировании адреса и данных. Алгоритм скремблирования зависит от задаваемого пользователем параметра. Ко всем записываемым данным автоматически добавляется контрольная сумма, проверяемая при считывании данных. Объем сохраняемых данных при записи равен 40 бит - 32 бита данные и 8 бит контрольная сумма. Эти 40 бит представляются в виде 5 слов по 8 бит, в зависимости от адреса слова переставляются, после чего каждое слово записывается в свой банк памяти.

Доступ к памяти обеспечивает «Модуль скремблирования и проверки целостности данных памяти».

В режиме standby память ключей нельзя сбросить ни по какому из указанных событий.

При отключении питания кроме батарейного или, если перейти в standby и выйти из него, то память ключей также не сбрасывается.

### 18.5 Контроллер генератора шума

Модуль служит для управления 256 кольцевыми генераторами, их включения, тестирования, а также формирования случайного 32 числа:

- модуль содержит 256 генераторов;
- модуль позволяет включать и выключать генераторы группами по 64;
- модуль позволяет задать паузу после включения генераторов, перед началом сбора случайного числа;

- модуль позволяет включать генераторы по внешнему сигналу или по биту регистра управления;
- модуль имеет несколько режимов запуска сбора случайного числа (одиночный, постоянный, по чтению прошлого значения);
- модуль формирует сигнал прерывания по окончанию сбора случайного числа.

Датчик случайных чисел представляет собой механизм получения непредсказуемой последовательности чисел, основанный на физическом недетерменированном процессе.

Для увеличения количества случайных чисел и улучшения их статистических характеристик, пользователь может воспользоваться аппаратно-программным механизмом генерации псевдослучайных чисел. В этом случае выход генератора случайных чисел является рандомизирующим фактором для выполнения операции шифрования с использованием стойкого шифра.

В загрузчике защищенной подсистемы реализовано программное статистическое тестирование блока. Если блок генератора случайных чисел не проходит статистические тесты, то загрузчик блокирует дальнейшую загрузку подсистемы. Загрузчик осуществляет тестирование после каждого сброса.

## 18.6 Блокировка отладочного интерфейса

Подключение к защищенной подсистеме через отладочный порт является одним из самых распространённых видов атаки. Доступ к порту отладки (DAP) или сканирующей цепочке (ATPG) позволяет получить доступ ко всем внутренним ресурсам защищенной подсистемы – регистрам CPU, встроенной OTP-памяти, ОЗУ и периферийным регистрам. По этой причине блокировка отладочного интерфейса должна быть первоочередной задачей, выполняемой при защите конечной системы.

В защищенной подсистеме блокировка отладочного интерфейса осуществляется путем записи слова безопасности SecurityWord в OTP-память по адресу 0. Описание битов слова безопасности приведено в таблице ниже. Адресное пространство защищенной подсистемы представлено в разделе «Программная модель защищенной подсистемы выполнения криптографических преобразований».

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31..8 | ATPG_EN | 0 | Запись 1 в любой бит запрещает производственное тестирование (ATPG) и переключение порта в режим SWD |
| 7 | DT_ACK_EN1<br>DT_DP_ENABLE3 | 0 | Запись 1 запрещает подтверждение включения модуля DAP и запрещает работу модуля DP |
| 6 | DT_RESET_REL3<br>DT_DP_ENABLE2 | 0 | Запись 1 запрещает снятие сигнала «Сброс» с модуля DAP и запрещает работу модуля DP |
| 5 | DT_ACK_EN0 | 0 | Запись 1 запрещает подтверждение включения модуля DAP |
| 4 | DT_RESET_REL2 | 0 | Запись 1 запрещает снятие сигнала «Сброс» с модуля DAP |
| 3 | DT_REQ_EN1 | 0 | Запись 1 запрещает запрос включения модуля DAP |
| 2 | DT_RESET_REL1 | 0 | Запись 1 запрещает снятие сигнала «Сброс» с модуля DAP |
| 1 | DT_REQ_EN0<br>DT_DP_ENABLE1 | 0 | Запись 1 запрещает запрос включения модуля DAP и запрещает работу модуля DP |
| 0 | DT_RESET_REL0<br>DT_DP_ENABLE0 | 0 | Запись 1 запрещает снятие сигнала «Сброс» с модуля DAP и запрещает работу модуля DP |

## 18.7 Блоки специальных вычислений

### 18.7.1 *Контроллер прямого и обратного L-преобразования (L_BLOCK_CNTR)*

Блок преобразования L / $L^{-1}$ соответствует таковому в ГОСТ Р34.12-2015.

Блок обладает следующими возможностями:

- возможность задания направления преобразования L / $L^{-1}$;
- возможность выбора числа R преобразований (0 - 31);
- возможность задания коэффициентов l преобразования через 16 таблиц соответствия a → K*a;
- время выполнения преобразований равно числу преобразований R плюс 1 такт;
- AHB-подключение;
- сигнал занятости модуля текущим преобразованием.

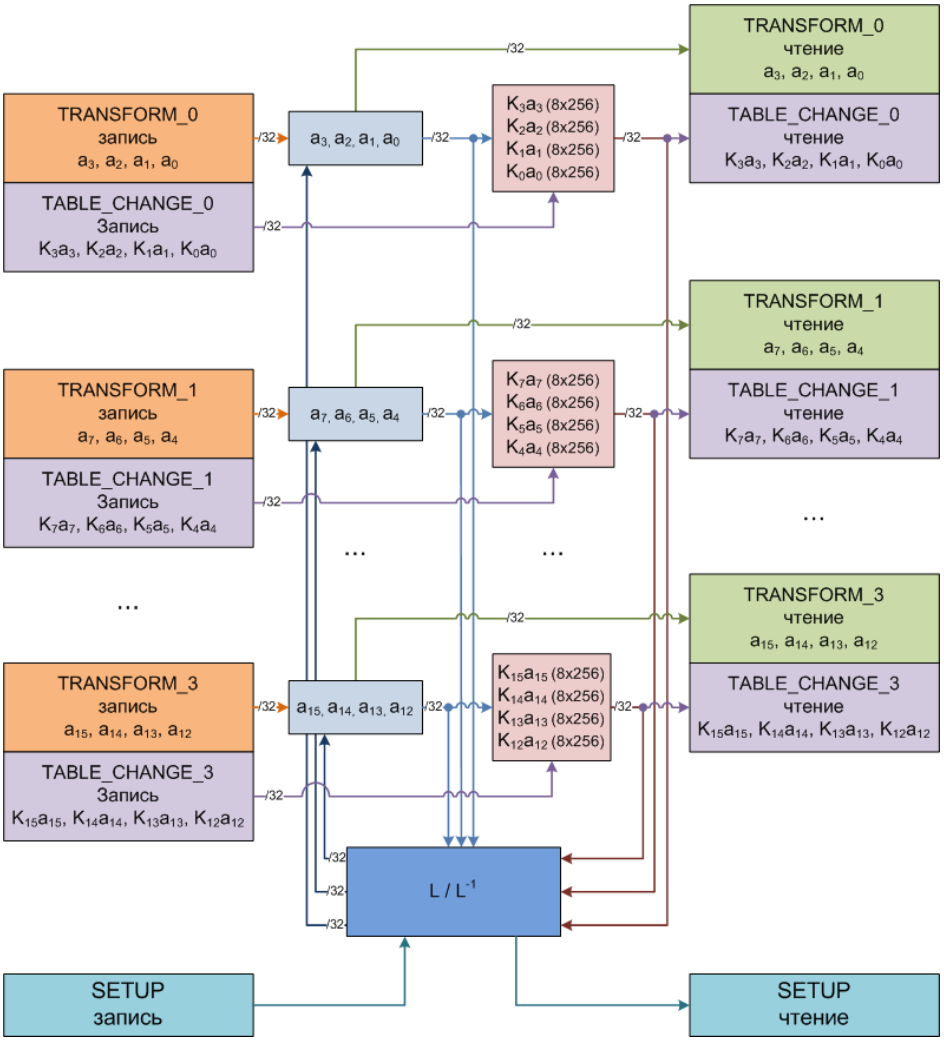



Рисунок 196

***Инициализация***

- В регистр SETUP задать значение 0x00000000;
- В регистры TRANSFORM_0 ... TRANSFORM_3 задать значение 0x00_00_00_00;
- В регистры TABLE_CHANGE_0 задать значение 0xTT_PP_RR_SS, где TT = $K_3$a, PP = $K_2$a, RR = $K_1$a, SS = $K_0$a, для a = 0x00;
- В регистры TABLE_CHANGE_1 задать значение 0xTT_PP_RR_SS, где TT = $K_7$a, PP = $K_6$a, RR = $K_5$a, SS = $K_4$a, для a = 0x00;
- В регистры TABLE_CHANGE_2 задать значение 0xTT_PP_RR_SS, где TT = $K_{11}$a, PP = $K_{10}$a, RR = $K_9$a, SS = $K_8$a, для a = 0x00;

- В регистры TABLE_CHANGE_3 задать значение 0xTT_PP_RR_SS, где TT = $K_{15}$a, PP = $K_{14}$a, RR = $K_{13}$a, SS = $K_{12}$a, для a = 0x00;
- В регистры TRANSFORM_0 ... TRANSFORM_3 задать значение 0x01_01_01_01;
- В регистры TABLE_CHANGE_0 задать значение 0xTT_PP_RR_SS, где TT = $K_3$a, PP = $K_2$a, RR = $K_1$a, SS = $K_0$a, для a = 0x01;
- В регистры TABLE_CHANGE_1 задать значение 0xTT_PP_RR_SS, где TT = $K_7$a, PP = $K_6$a, RR = $K_5$a, SS = $K_4$a, для a = 0x01;
- В регистры TABLE_CHANGE_2 задать значение 0xTT_PP_RR_SS, где TT = $K_{11}$a, PP = $K_{10}$a, RR = $K_9$a, SS = $K_8$a, для a = 0x01;
- В регистры TABLE_CHANGE_3 задать значение 0xTT_PP_RR_SS, где TT = $K_{15}$a, PP = $K_{14}$a, RR = $K_{13}$a, SS = $K_{12}$a, для a = 0x01;
- ...
- В регистры TRANSFORM_0 ... TRANSFORM_3 задать значение 0xFF_FF_FF_FF;
- В регистры TABLE_CHANGE_0 задать значение 0xTT_PP_RR_SS, где TT = $K_3$a, PP = $K_2$a, RR = $K_1$a, SS = $K_0$a, для a = 0xFF;
- В регистры TABLE_CHANGE_1 задать значение 0xTT_PP_RR_SS, где TT = $K_7$a, PP = $K_6$a, RR = $K_5$a, SS = $K_4$a, для a = 0xFF;
- В регистры TABLE_CHANGE_2 задать значение 0xTT_PP_RR_SS, где TT = $K_{11}$a, PP = $K_{10}$a, RR = $K_9$a, SS = $K_8$a, для a = 0xFF;
- В регистры TABLE_CHANGE_3 задать значение 0xTT_PP_RR_SS, где TT = $K_{15}$a, PP = $K_{14}$a, RR = $K_{13}$a, SS = $K_{12}$a, для a = 0xFF;
- В регистр SETUP задать значение необходимого преобразования, направление и число повторений преобразования R / $R^{-1}$.

***Выполнение преобразования***

В регистры TRANSFORM_0 ... TRANSFORM_3 записать данные для преобразования. При записи в старший байт регистра TRANSFORM_3 автоматически запускается преобразование. Результат преобразования доступен через (SET_R_COUNT + 1) тактов, SET_R_COUNT – значение поля регистра SETUP. Также ход преобразование можно отслеживать по счетчику CURR_R_COUNT, регистра SETUP. При достижении счетчиком значения 0, преобразование закончено. Результат преобразования считывается из регистров TRANSFORM_0 ... TRANSFORM_3. Для смены направления преобразования перед записью регистров TRANSFORM_0 ... TRANSFORM_3 необходимо изменить значение регистра SETUP. Изменять значение регистра SETUP можно сразу после старта очередного преобразования, новое значение будет применено к следующему преобразованию.

### 18.7.2 *Контроллер замены (S_BLOCK_CNTR)*

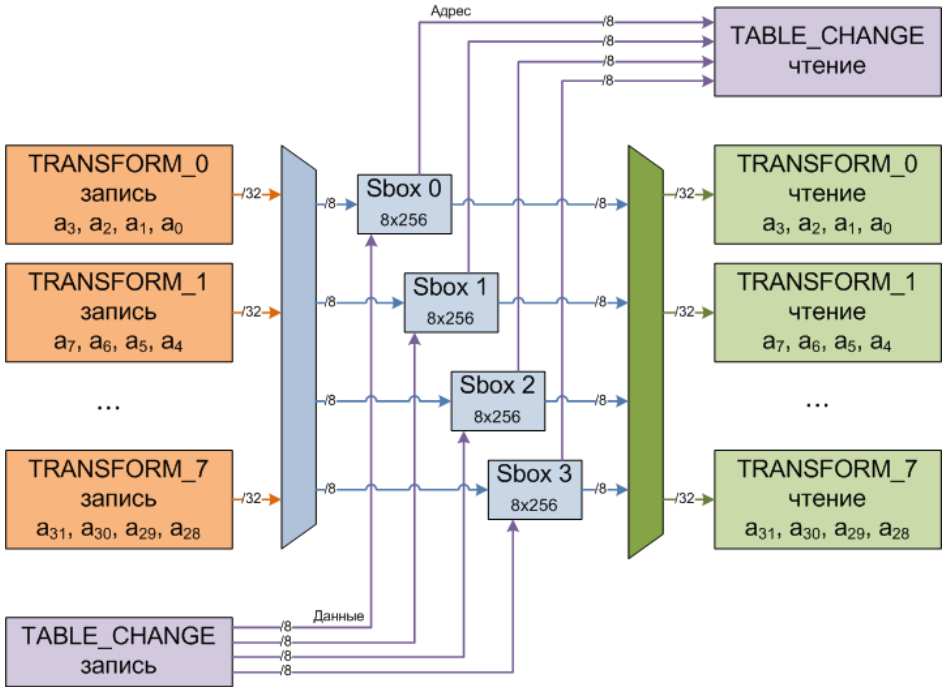



Рисунок 197

Выполняет ускорение типовой задачи блочной замены. При работе данного блока производится преобразование 8bit → 8bit по таблице из 256 элементов. Каждое входное число заменяется на число из ячейки таблицы с индексом соответствующим входному числу.

Основные особенности блока:

- возможность задания таблицы замены 8bit → 8bit;
- обработка 32-битного слова за один такт;
- прием до восьми 32-битных слов с конвейерной обработкой и сохранением результата преобразования;
- AHB-подключение.

***Инициализация 8-битного S-BOX***

- В регистр TRANSFORM_0 задать значение 0x00_00_00_00;
- В регистр TABLE_CHANGE задать значение 0xTT_TT_TT_TT, где TT - значение преобразования, 0x00 → TT;
- В регистр TRANSFORM_0 задать значение 0x01_01_01_01;
- В регистр TABLE_CHANGE задать значение 0xTT_TT_TT_TT, где TT - значение преобразования, 0x01 → TT;
- ...
- В регистр TRANSFORM_0 задать значение 0xFF_FF_FF_FF;
- В регистр TABLE_CHANGE задать значение 0xTT_TT_TT_TT, где TT - значение преобразования, 0xFF → TT.

***Инициализация 4-битного S-BOX***

- В регистр TRANSFORM_0 задать значение 0x00_00_00_00;
- В регистр TABLE_CHANGE задать значение 0xTP_TP_TP_TP, где T, P - значение преобразования, 0x0 → T, 0x0 → P;
- В регистр TRANSFORM_0 задать значение 0x01_01_01_01;
- В регистр TABLE_CHANGE задать значение 0xTP_TP_TP_TP, где T, P - значение преобразования, 0x0 → T, 0x1 → P;
- ...

- В регистр TRANSFORM_0 задать значение 0x10_10_10_10;
- В регистр TABLE_CHANGE задать значение 0xTP_TP_TP_TP, где T, P - значение преобразования, 0x1 → T, 0x0 → P;
- ...
- В регистр TRANSFORM_0 задать значение 0xFF_FF_FF_FF;
- В регистр TABLE_CHANGE задать значение 0xTT_TT_TT_TT, где T, P - значение преобразования, 0xF → T, 0xF → P.

***Выполнение преобразования***

В регистры TRANSFORM_0 ... TRANSFORM_7 записать необходимое число слов для преобразования, затем считать преобразованное значение из регистров TRANSFORM_0 ... TRANSFORM_7. В случае преобразования одного слова (преобразование 32 бит) между записью в регистр TRANSFORM_0 и чтением результата из этого регистра необходима пауза в один такт. В случае преобразования более 32 бит, чтение можно выполнять сразу же после записи всех регистров. Для преобразования можно использовать любое подмножество регистров, результаты преобразования будут в регистрах с соответствующими номерами.

### 18.7.3 *Контроллер битовой замены (P_BIT_CNTR)*

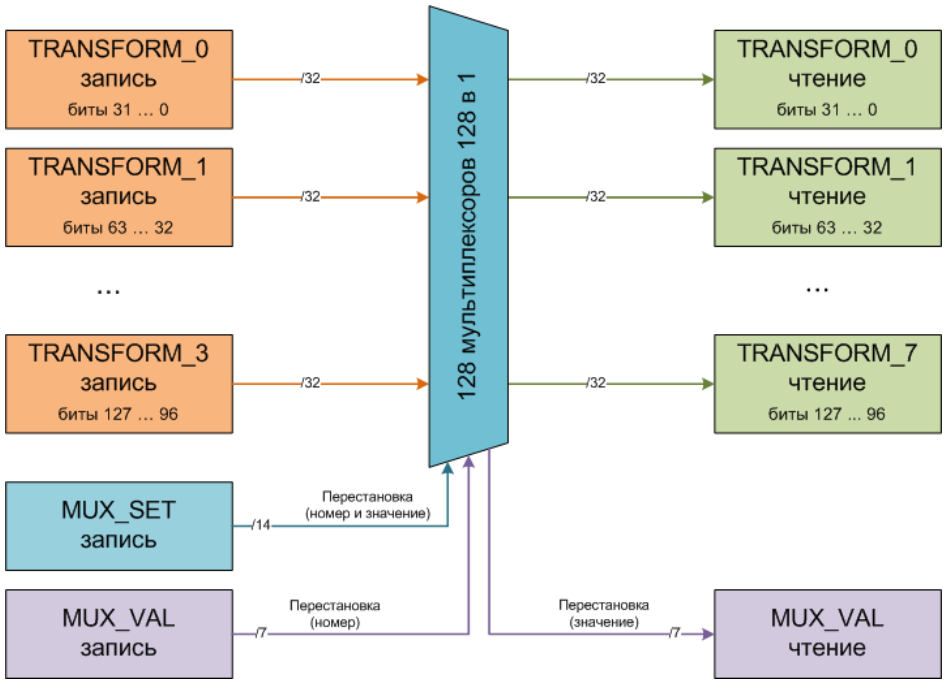



Рисунок 198

Блок выполняет перестановку по таблице бит входного 128 битного слова.

Основные особенности блока:

- возможность задания таблицы перестановки бит (каждому выходному биту задается номер используемого входного бита);
- обновление выходного слова в том же такте после изменения входного;
- конвейерный прием и обработка до четырех 32-битных слов;
- AHB-подключение.

***Инициализация***

- В регистр MUX_SET задать значение 0x0000_00_TT, где TT - номер входного бита, который необходимо выдавать на 0 бит;
- В регистр MUX_SET задать значение 0x0000_01_TT, где TT - номер входного бита, который необходимо выдавать на 1 бит;

– ...
– В регистр MUX_SET задать значение 0x0000_7F_TT, где TT - номер входного бита, который необходимо выдавать на 127 бит.

***Выполнение преобразования***

В регистры TRANSFORM_0 ... TRANSFORM_3 записать данные для преобразования. Сразу же из регистров TRANSFORM_0 ... TRANSFORM_3 можно считать результат перестановки. Обновление выходных данных происходит одновременно с записью любой части входных данных.

### 18.7.4 *Контроллер байтовой замены (P_BYTE_CNTR)*

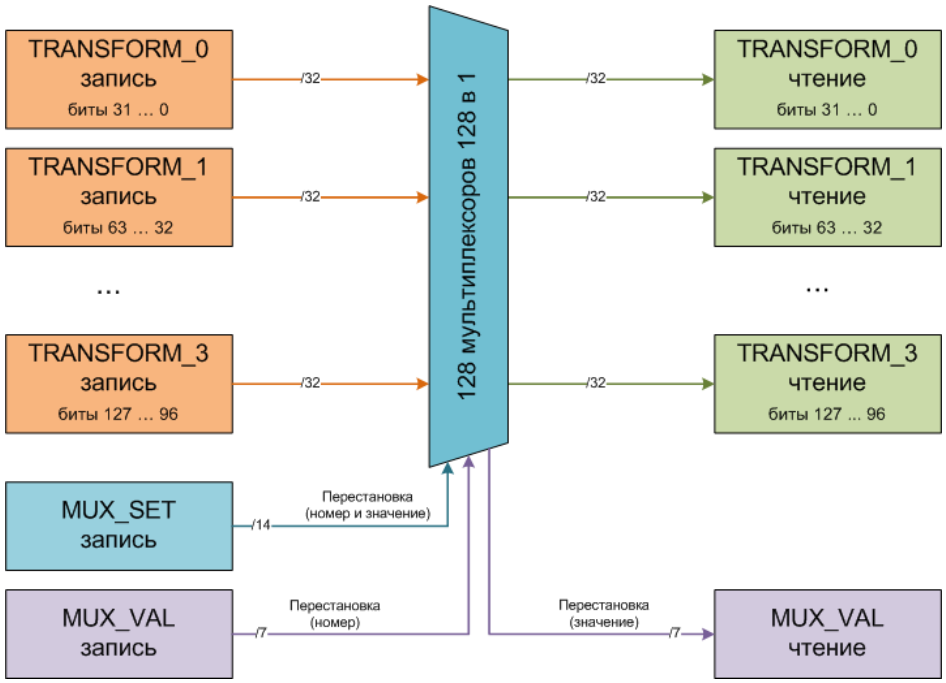



Рисунок 199

Блок осуществляет перестановку байт входного 64-байтового массива.

Основные особенности блока:

– настраиваемая таблица перестановки (каждому выходному байту задается номер используемого входного байта);
– обновление выходного массива данных за такт после изменения входного массива;
– конвейерный прием и обработка до 16-ти слов размером 32 бита;
– AHB-подключение.

***Инициализация***

– В регистр MUX_SET задать значение 0x0000_00_TT, где TT - номер входного байта, который необходимо выдавать на 0 байт;
– В регистр MUX_SET задать значение 0x0000_01_TT, где TT - номер входного бита, который необходимо выдавать на 1 байт;
– ...
– В регистр MUX_SET задать значение 0x0000_3F_TT, где TT - номер входного бита, который необходимо выдавать на 63 байт;

***Выполнение преобразования***

В регистры TRANSFORM_0 ... TRANSFORM_15 записать данные для преобразования. Сразу же из регистров TRANSFORM_0 ... TRANSFORM_15 можно считать

результат перестановки. Обновление выходных данных происходит одновременно с записью любой части входных данных.

### 18.7.5 *Контроллер шифрования (DES_CNTR)*

Блок выполняет операции одного раунда шифрования DES. Блок принимает на вход 32-битные данные и 48-битный раундовый ключ (за две 32-битные посылки). Модуль производит расширение входных данных, сложение их с ключом, подстановку через S-box и финальную перестановку бит.

Основные особенности:

- настраиваемые таблицы расширения, подстановки и перестановки;
- конвейерный прием данных для обработки и ключа 32-битными словами;
- завершение преобразование на следующий такт после передачи ключа;
- AHB-подключение.

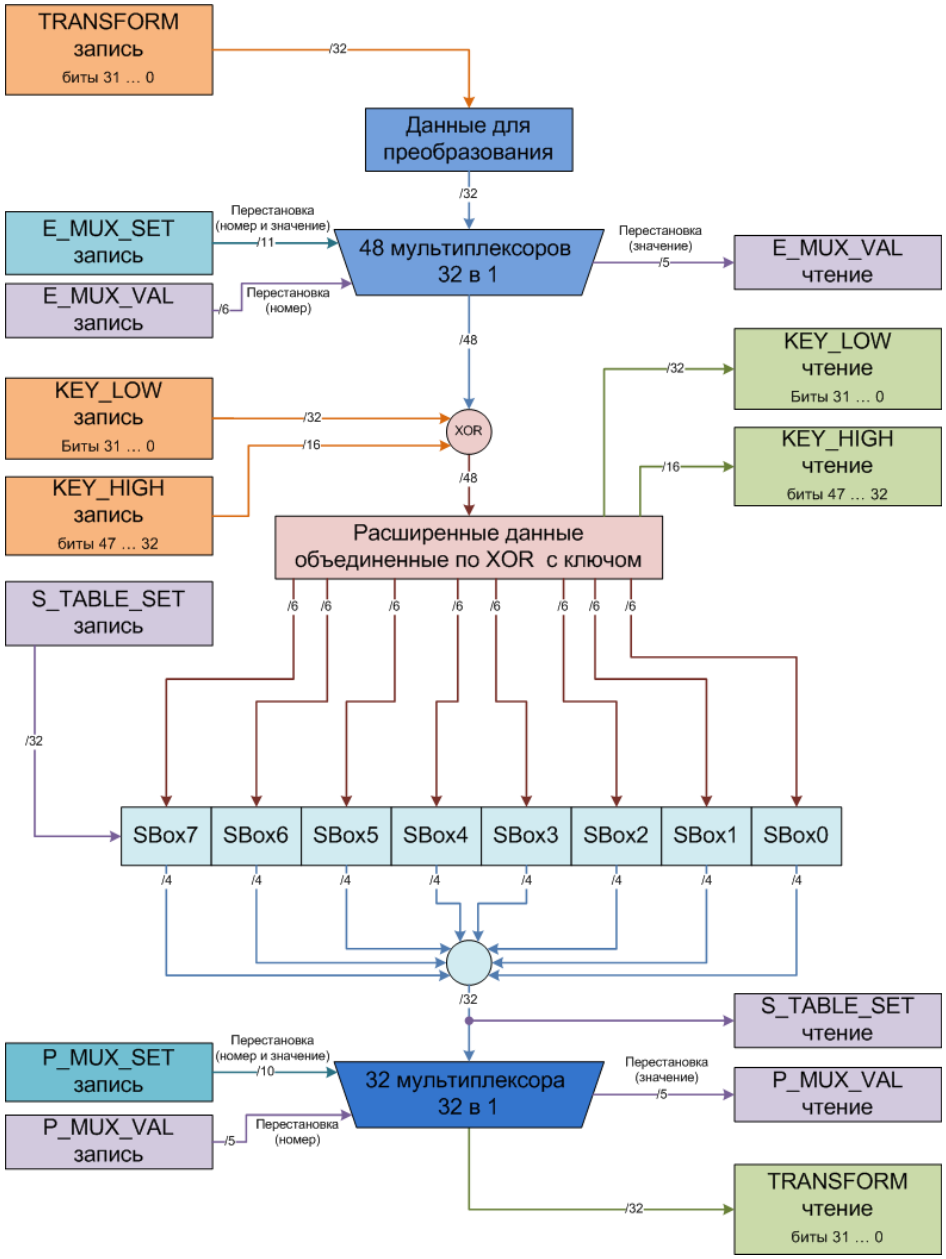



Рисунок 200

***Инициализация***

- В регистр E_MUX_SET задать значение 0x0000_00_TT, где TT - номер входного бита, который необходимо выдавать на 0 бит расширенных данных;
- В регистр E_MUX_SET задать значение 0x0000_01_TT, где TT - номер входного бита, который необходимо выдавать на 1 бит расширенных данных;
- ...
- В регистр E_MUX_SET задать значение 0x0000_2F_TT, где TT - номер входного бита, который необходимо выдавать на 47 бит расширенных данных;
- В регистр P_MUX_SET задать значение 0x0000_00_TT, где TT - номер бита после S преобразования, который необходимо выдавать на 0 бит результата;
- В регистр P_MUX_SET задать значение 0x0000_01_TT, где TT - номер бита после S преобразования, который необходимо выдавать на 1 бит результата;
- ...
- В регистр P_MUX_SET задать значение 0x0000_1F_TT, где TT - бита после S преобразования, который необходимо выдавать на 31 бит результата;
- в регистр TRANSFORM задать значение 0x0000_0000
- В регистры KEY_HIGH и KEY_LOW задать значения 0x0000_0000 и 0x00000000 соответственно;
- В регистр S_TABLE_SET задать значение S7_S6_S5_S4_S3_S2_S1_S0, где Si – это значение S преобразования для нулевых входных данных;
- В регистры KEY_HIGH и KEY_LOW задать значения 0x0000_0410 и 0x41041041 соответственно (0x01 со смешением 6 бит);
- В регистр S_TABLE_SET задать значение S7_S6_S5_S4_S3_S2_S1_S0, где Si – это значение S преобразования для входных данных 0x01;
- ...
- В регистры KEY_HIGH и KEY_LOW задать значения 0x0000_FFFF и 0xFFFFFFFF соответственно (0x3F со смешением 6 бит);
- В регистр S_TABLE_SET задать значение S7_S6_S5_S4_S3_S2_S1_S0, где Si – это значение S преобразования для входных данных 0x3F

***Выполнение преобразования***

В регистр TRANSFORM записать данные для преобразования, затем в регистры KEY_LOW записать младшие 32 бита сессионного ключа, а в регистр KEY_HIGH записать старшие 16 бит сессионного ключа. Через 1 такт после записи сессионного ключа из регистра TRANSFORM можно забирать преобразованные данные.

## 18.8 Контроллер сопроцессора длинночисленной арифметики (ALU_CNTR)

Сопроцессор длинночисленной арифметики должен выполнять аппаратно-ускоренные математические операции. Операндами в данном случае являются длинные целые числа. Ускорение работы достигается за счёт высокой разрядности выполняемых вычислений. Предлагается реализация данного сопроцессора как внешнее АЛУ с длинными регистрами.

***Блок обладает следующими возможностями:***

- доступ со стороны ядра M0 в регистры по AHB, размерность доступа 32 бита;
- доступ со стороны АЛУ в регистры по 128 бит;
- шаг адреса со стороны ядра M0 32 бита;
- шаг адреса по стороны АЛУ 128 бита;
- выбираемые АЛУ данные для операции из регистра могут быть использованы как напрямую, так и инвертировано;
- все команды дают ответ на следующий такт;
- для результатов умножителя выделено четыре специальных регистра;

- АЛУ имеет внутреннюю память команд, два блока по 1024 команды и механизм последовательного исполнения команд;
- модуль поддерживает обмен только 32-битными словами, при записи 16- и 8-битных данных поведение модуля непредсказуемо;
- модуль может выполнять следующие операции:
  - **ADD** - сложение 128-битных слов без учета бита переноса прошлой операции;
  - **C_ADD** - сложение 128-битных слов с учетом бита переноса прошлой операции;
  - **O_ADD** - сложение 128-битных слов с добавлением единицы как бита переноса;
  - **SHL** - сдвиг 128-битного слова на 1 бит влево, новый бит заполняется нулем;
  - **C_SHL -** сдвиг 128-битного слова на 1 бит влево, новый бит заполняется битом переноса;
  - **S_SHL** - сдвиг 128-битного слова на 1 бит влево, новый бит заполняется прошлым выдвинутым битом;
  - **O_SHL** - сдвиг 128-битного слова на 1 бит влево, новый бит заполняется единицей;
  - **SHR** - сдвиг 128-битного слова на 1 бит вправо, новый бит заполняется нулем;
  - **C_SHR** - сдвиг 128-битного слова на 1 бит вправо, новый бит заполняется битом переноса;
  - **S_SHR** - сдвиг 128-битного слова на 1 бит вправо, новый бит заполняется прошлым выдвинутым битом;
  - **O_SHR** - сдвиг 128-битного слова на 1 бит вправо, новый бит заполняется единицей;
  - **MOV** - перемещение 128-битного слова (чтение из одного адреса и запись в другой адрес);
  - **C_MOV** - перемещение 128-битного слова (чтение из одного адреса и запись в другой адрес) при условии установленного бита переноса, если бит переноса с прошлой операции 0, перемещение не производится;
  - **MUL** - умножение 128-битного слова на 128-битное слово с сохранением 256 битного результата.

    Особенности работы:

    – Операнды для операции умножителя задаются так же, как и для других операций, ограничений нет;

    – 256-битные регистры результатов умножителя (MA, MB, MC, MD) недоступны для считывания со стороны защищенной подсистемы (не отображаются в адресном пространстве), поэтому для чтения результатов умножителя необходимо выполнить команду MOV (OP_CODE = 0b1), указав в качестве "аргумента A" регистр результата умножителя;

    – Результаты умножителя можно использовать только как аргумент для последующих операций. В этом случае необходимо использовать адреса аргументов 112-119 (4 регистра по два 128-битных слова соответствуют регистрам MA, MB, MC, MD).
  - **NOP** – бездействие.

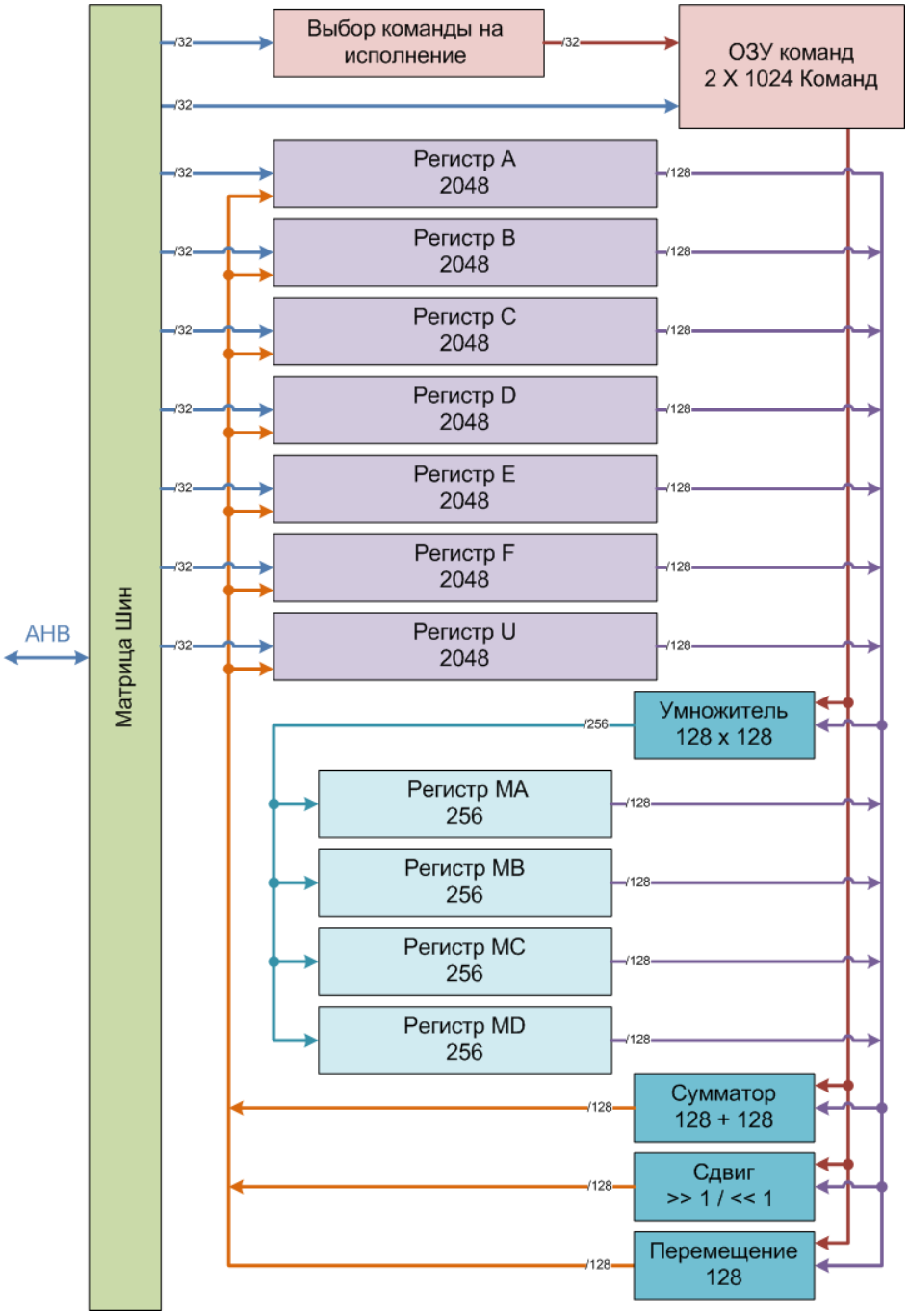

AHB
Матрица Шин
Выбор команды на исполнение
ОЗУ команд
2 X 1024 Команд
Регистр A
2048
Регистр B
2048
Регистр C
2048
Регистр D
2048
Регистр E
2048
Регистр F
2048
Регистр U
2048
Умножитель
128 x 128
Регистр MA
256
Регистр MB
256
Регистр MC
256
Регистр MD
256
Сумматор
128 + 128
Сдвиг
>> 1 / << 1
Перемещение
128
/32
/128
/256

Рисунок 201

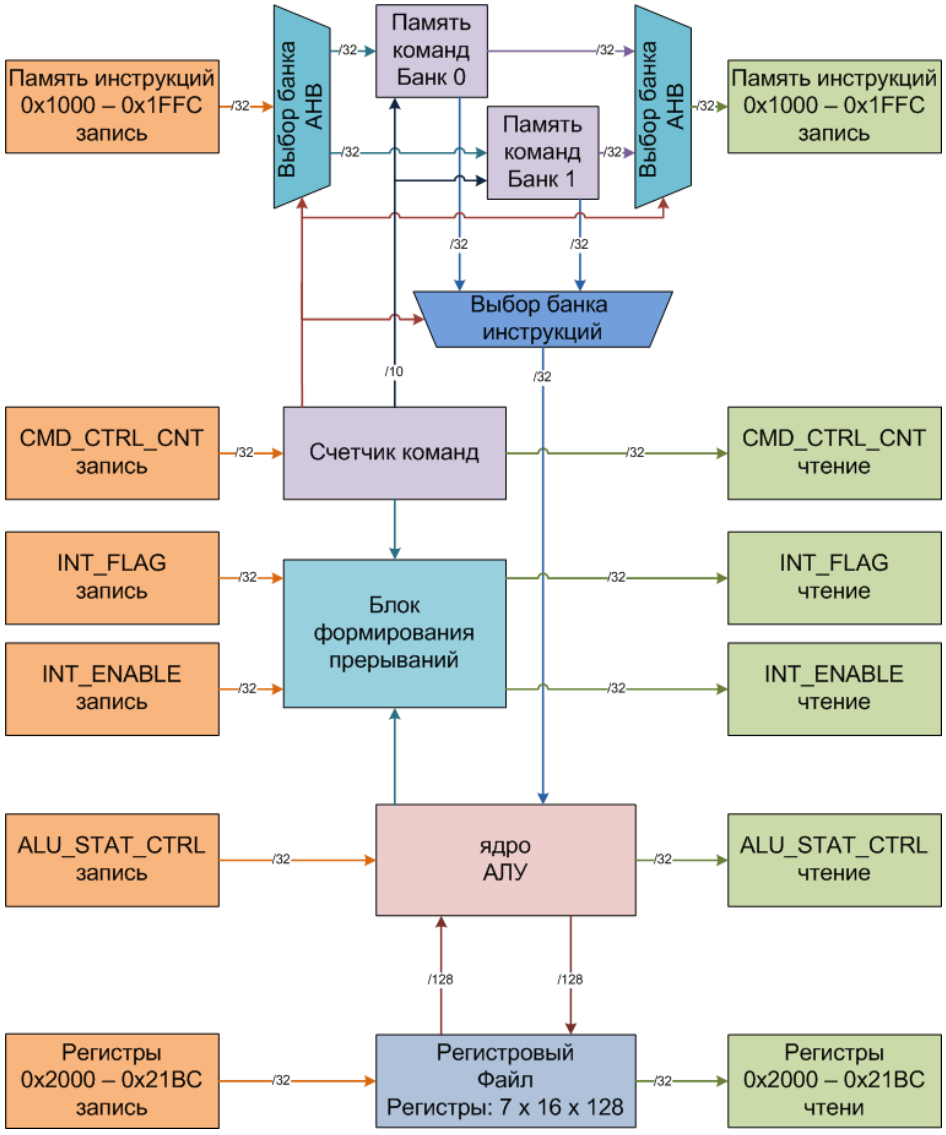



Рисунок 202

### 18.8.1 *Реализация длинных операций*

#### 18.8.1.1 условные обозначения

- ***REG_X***[E:S] - выбор битов с S по E из регистра X;
- ***~REG_X***[E:S] - выбор инверсии битов с S по E из регистра X;
- ***CONST_0***[S] - константное значение все нули, размерности S;
- ***CONST_1***[S] - константное значение все единицы, размерности S.

#### 18.8.1.2 Сложение 2048 бит (A + B → C)

Выполнение 16 команд

- **ADD**(*REG_A*[127:0], *REG_B*[127:0]) → *REG_C*[127:0];
- **C_ADD**(*REG_A*[255:128], *REG_B*[255:128]) → *REG_C*[255:128];
- **...**
- **C_ADD**(*REG_A*[2047:1920], *REG_B*[2047:1920]) → *REG_C*[2047:1920].

#### 18.8.1.3 Вычитание 2048 бит (A - B → C)

Выполнение 16 команд

- **O_ADD**(*REG_A*[127:0], **~***REG_B*[127:0]) → *REG_C*[127:0];
- **C_ADD**(*REG_A*[255:128], **~***REG_B*[255:128]) → *REG_C*[255:128];

- **...**
- **C_ADD**(*REG_A*[2047:1920], **~***REG_B*[2047:1920]) → *REG_C*[2047:1920].

#### 18.8.1.4 Умножение 1024 бита (A * B → C)

Выполнение 536 команд (при умножении за 2 такта без конвейера)

*Строка 0*

- **MUL**(*REG_A*[127:0], *REG_B*[1023:896]) → *REG_MA*[255:0];
- **NOP**;
- **ADD**(*REG_MA*[127:0], *REG_C*[1023:896]) → *REG_C*[1023:896];
- **C_ADD**(*REG_MA*[255:128], *REG_C*[1151:1024]) → *REG_C*[1151:1024];
- **C_ADD**(***CONST_0***[128], *REG_C*[1279:1152]) → *REG_C*[1215:1088];
- **MUL**(*REG_A*[127:0], *REG_B*[895:768]) → *REG_MA*[255:0];
- **NOP**;
- **ADD**(*REG_MA*[127:0], *REG_C*[895:768]) → *REG_C*[895:768];
- **C_ADD**(*REG_MA*[255:128], *REG_C*[1023:896]) → *REG_C*[1023:896];
- **C_ADD**(***CONST_0***[128], *REG_C*[1151:1024]) → *REG_C*[1151:1024];
- **C_ADD**(***CONST_0***[128], *REG_C*[1279:1152]) → *REG_C*[1279:1152];

- **...**

- **MUL**(*REG_A*[127:0], *REG_B*[127:0]) → *REG_MA*[255:0];
- **NOP**;
- **ADD**(*REG_MA*[127:0], *REG_C*[127:0]) → *REG_C*[127:0];
- **C_ADD**(*REG_MA*[255:128], *REG_C*[255:128]) → *REG_C*[255:128];
- **C_ADD**(***CONST_0***[128], *REG_C*[383:256]) → *REG_C*[383:256];
- **C_ADD**(***CONST_0***[128], *REG_C*[511:384]) → *REG_C*[511:384];
- **...**
- **C_ADD**(***CONST_0***[128], *REG_C*[1279:1152]) → *REG_C*[1279:1152];

*Строка 1*

- **MUL**(*REG_A*[255:128], *REG_B*[1023:896]) → *REG_MA*[255:0];
- **NOP**;
- **ADD**(*REG_MA*[255:128], *REG_C*[1151:1024]) → *REG_C*[1151:1024];
- **C_ADD**(*REG_MA*[255:128], *REG_C*[1279:1152]) → *REG_C*[1279:1152];
- **C_ADD**(***CONST_0***[128], *REG_C*[1407:1280]) → *REG_C*[1407:1280];
- **MUL**(*REG_A*[255:128], *REG_B*[895:768]) → *REG_MA*[255:0];
- **NOP**;
- **ADD**(*REG_MA*[127:0], *REG_C*[1023:896]) → *REG_C*[1023:896];
- **C_ADD**(*REG_MA*[255:128], *REG_C*[1151:1024]) → *REG_C*[1151:1024];
- **C_ADD**(***CONST_0***[128], *REG_C*[1279:1152]) → *REG_C*[1279:1152];
- **C_ADD**(***CONST_0***[128], *REG_C*[1407:1280]) → *REG_C*[1407:1280];

- **...**

- **MUL**(*REG_A*[255:128], *REG_B*[127:0]) → *REG_MA*[255:0];
- **NOP**;
- **ADD**(*REG_MA*[127:0], *REG_C*[255:128]) → *REG_C*[255:128];
- **C_ADD**(*REG_MA*[255:128], *REG_C*[383:256]) → *REG_C*[383:256];
- **C_ADD**(***CONST_0***[128], *REG_C*[511:384]) → *REG_C*[511:384];
- **C_ADD**(***CONST_0***[128], *REG_C*[639:512]) → *REG_C*[639:512];
- **...**
- **C_ADD**(***CONST_0***[128], *REG_C*[1407:1280]) → *REG_C*[1407:1280];

– **...**

*Строка 7*

– **MUL**(*REG_A*[1023:896], *REG_B*[1023:896]) → *REG_MA*[255:0];
– **NOP**;
– **ADD**(*REG_MA*[127:0], *REG_C*[1919:1792]) → *REG_C*[1919:1792];
– **C_ADD**(*REG_MA*[255:128], *REG_C*[2047:1920]) → *REG_C*[2047:1920];
– **MUL**(*REG_A*[1023:960], *REG_B*[895:768]) → *REG_MA*[255:0];
– **NOP**;
– **ADD**(*REG_MA*[127:0], *REG_C*[1791:1664]) → *REG_C*[1791:1664];
– **C_ADD**(*REG_MA*[255:128], *REG_C*[1919:1792]) → *REG_C*[1919:1792];
– **C_ADD**(***CONST_0***[128], *REG_C*[2047:1920]) → *REG_C*[2047:1920];
–
– **...**
–
– **MUL**(*REG_A*[1023:896], *REG_B*[127:0]) → *REG_MA*[255:0];
– **NOP**;
– **ADD**(*REG_MA*[127:0], *REG_C*[1023:896]) → *REG_C*[1023:896];
– **C_ADD**(*REG_MA*[255:128], *REG_C*[1151:1024]) → *REG_C*[1151:1024];
– **C_ADD**(***CONST_0***[128], *REG_C*[1279:1152]) → *REG_C*[1279:1152];
– **C_ADD**(***CONST_0***[128], *REG_C*[1407:1280]) → *REG_C*[1407:1280];
– **...**
– **C_ADD**(***CONST_0***[128], *REG_C*[2047:1920]) → *REG_C*[2047:1920];

#### 18.8.1.5 Деление 2048 (A / B → C, остаток A)

Выполнение 122944 команд, при делении 2048 на 128 бит. Для 2048 бит деление занимает 64 такта на бит результата.

Расчет 1 бита частного:

– **O_ADD**(*REG_A*[127:0], **~***REG_B*[127:0]) → *REG_D*[127:0];
– **C_ADD**(*REG_A*[255:128], **~***REG_B*[255:128]) → *REG_D*[255:128];
– **...**
– **C_ADD**(*REG_A*[2047:1920], **~***REG_B*[2047:1920]) → *REG_D*[2047:1920];
– **C_MOV(***REG_D*[127:0]**)** → *REG_A*[127:0]**;**
– **C_MOV(***REG_D*[255:128]**)** → *REG_A*[255:128]**;**
– **...**
– **C_MOV(***REG_D*[2047:1920]**)** → *REG_A*[2047:1920]**;**
– **C_SHL**(***REG_C[127:0]***) → ***REG_C[127:0]***
– **S_SHL(***REG_C**[255:128***) → ***REG_C***;
– ...
– **S_SHL(***REG_C**[2047:1920]***) → *REG_C**[2047:1920];***
– **SHR**(***REG_B[2047:1920]***) → ***REG_B[2047:1920];***
– **S_SHR**(***REG_B[1919:1792]***) → ***REG_B[1919:1792];***
– **...**
**S_SHR**(***REG_B[127:0]***) → ***REG_B[127:0];***

В начальный момент в регистре A находится делимое, в регистре B находится делитель, старший бит делителя должен быть не правее старшего бита делимого (B >= A). Операцию получения очередного бита частного необходимо повторять до тех пор, пока делитель не займет крайне правое положение (с каждой итерацией делитель сдвигается на один такт), после чего выполнить еще одну итерацию. В регистре A останется остаток от деления, в регистре C будет находится частное.

## 18.9 Контроллер обработки датчиков безопасности (SENSORS_CNTR)

### 18.9.1 Общее описание

Блок используется для обработки датчиков обнаружения атак на микросхему. Модуль обрабатывает активную защитную сетку и датчики ЭМИ.

Защитная сетка включает в себя блок формирования последовательности данных по ГОСТ 28147-89, работающий в режиме простой замены. В качестве входных данных задается 256-битный ключ (регистры MESH_KEY0-MESH_KEY7) и 64-битные данные инициализации (MESH_INIT0-MESH_INIT1). Эти данные передаются на плавающую сетку с последующим контролем соответствия переданных и полученных по сетке уровней сигналов (нарушение сетки или замыкание ее слоев приводит к расхождению заданного и принятого уровней, что вызывает формирование сигнала тревоги). Защитная сетка находится в верхнем слое микросхемы и покрывает защищенную подсистему.

Защитная сетка начинает работать с нулевым ключом и данными инициализации на частоте генератора HSI после включения основного питания микросхемы и перевода сигнала сброса микросхемы в неактивное состояние. Запись ключа и данных инициализации в регистры MESH_KEY0-MESH_KEY7 и MESH_INIT0-MESH_INIT1 можно осуществить только один раз после сброса. После каждых 32 раундов ГОСТ 28147-89 блок формирования данных осуществляет повтор рандомизации. Если данные инициализации блока формирования данных по ГОСТ 28147-89 после сброса микросхемы были нулевые, то после каждых 32 раундов они меняют свое значение и каждые следующие 32 раунда начинаются с новыми данными. Значения регистров MESH_KEY0-MESH_KEY7 и MESH_INIT0-MESH_INIT1 при этом не изменяются. Предусмотрена возможность настройки интервала переключения защитной сетки в диапазоне от 3 до 256 тактов рабочей частоты HSI. Настройка интервала переключения после сброса осуществляется однократно в регистре MESH_CNTR.

Датчики ЭМИ обрабатывают периодически переключаемые к нулю и питанию линии (в противофазе, то есть в каждый момент времени сигналы на линиях должны быть разных уровней) с последующим контролем, что сигналы на линиях отличаются. При наличии на линиях одинаковых уровней формируется сигнал тревоги. Переключение линий осуществляется на частоте HSI.

Условная схема блока SENSORS_CNTR представлена на рисунке 203.

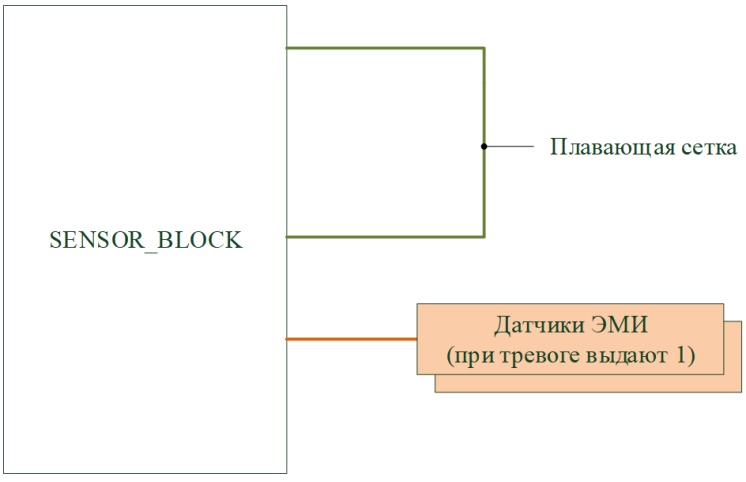



Рисунок 203 – Условная схема блока SENSORS_CNTR

По сигналу тревоги от датчиков возможнен сброс памяти регистров и криптографических ключей и формирование сигнала прерывания.

### 18.9.2 Описание функционирования блока и режимов работы

Блок непрерывно отслеживает состояние входов датчиков и защитной сетки. Работа датчиков разрешается записью в регистр SENS_ENABLE, возможна только однократная запись. Состояние датчиков отражается в регистры SENS_STATE (фиксируется факт срабатывания датчиков за все время после сброса микросхемы) и SENS_RT_STATE (текущее состояние сигналов тревоги от датчиков). Сброс флагов срабатывания датчиков ЭМИ производится записью 0 в биты 1-3 STATE_REG, только при условии отсутствия ненулевого входа. Сброс флага срабатывания защитной сетки производится записью 0 в бит 4 регистра SENS_STATE, при записи в момент проверки (конец временного интервала, заданного в регистре MESH_CNTR) сброс произойдет только при условии совпадения уровней входа и выхода. Запись 0 в середине интервала переключения сетки сбросит флаг срабатывания до ближайшей проверки. Запись 1 в биты регистра SENS_STATE не меняет его состояния.

Для плавающей защитной сетки могут быть однократно настроены ключ и данные инициализации по ГОСТ 28147-89 через регистры MESH_KEY0-MESH_KEY7 и MESH_INIT0-MESH_INIT1, а также время ее переключения в тактах частоты HSI записью в регистр MESH_CNTR.

Модуль SENSORS_CNTR имеет специальные регистры настройки с выбором датчиков, вызывающих формирование сигнала запроса прерывания и сигнала запроса сброса памяти ключей. Сигнал запроса прерывания и сброса памяти ключей формируется при наличии в SENS_STATE единичных битов, помеченных маской в регистрах SENS_INT_MASK и SENS_KEY_RESET_MASK соответственно.

## 18.10 Контроллер монитора частоты (CLK_MEASURE_CNTR)

Блок датчика частоты является важным элементом защиты защищенной части микроконтроллера. Для избежания возможных атак через изменение тактовой частоты реализован датчик частоты, который сравнивает подаваемый синхросигнал с опорным, и генерирует один из двух типов событий: сброс ключей батарейного домена, прерывание, в случае существенного отличия синхросигналов. Необходимо обеспечить невозможность отключения синхросигнала HSI.

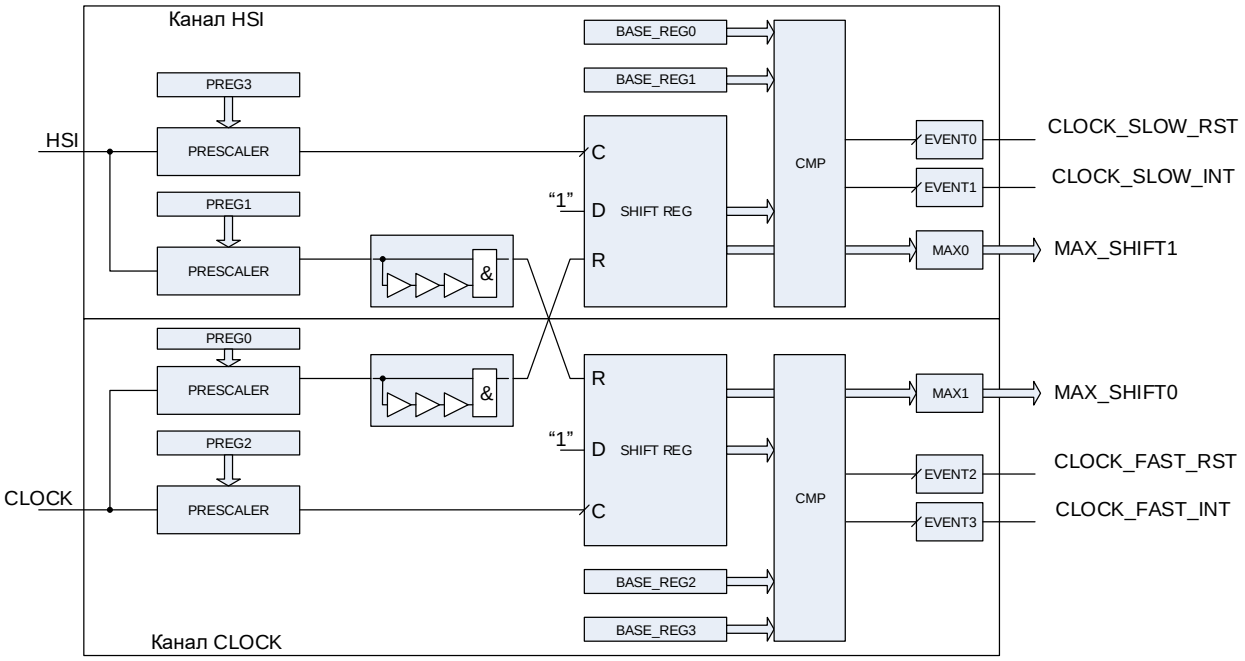



Рисунок 204 – Схема блока датчика частоты

При настройке блока, исходя из ожидаемых значений частот сигналов HSI и CLOCK, необходимо задать значения предварительных делителей в регистрах ***PREGx*** так, чтобы:

$$\begin{cases} \dfrac{\dfrac{f_{HSI}}{PREG3+1}}{\dfrac{f_{CLOCK}}{PREG0*}} = K_0, \text{где } K_0 \in [1..15] \\ \dfrac{\dfrac{f_{CLOCK}}{PREG2+1}}{\dfrac{f_{HSI}}{PREG1*}} = K_1, \text{где } K_1 \in [1..15] \end{cases}$$

$*PREG0$ – при $f_{HSI} > f_{CLOCK}$,

$PREG0+1$ – при $f_{HSI} < f_{CLOCK}$,

$*PREG1$ – при $f_{HSI} < f_{CLOCK}$,

$PREG1+1$ – при $f_{HSI} > f_{CLOCK}$.

Тогда в регистре ***SHIFT_REG*** канала HSI будут возникать значения от 0 до $2^{K0}$-1, а в канале CLOCK – от 0 до $2^{K1}$-1.

При снижении частоты CLOCK:

- в регистре канала HSI будут возникать значения от 0 до $2^{K2}$-1, причем $K_2 > K_0$;
- в регистре канала CLOCK будут возникать значения от 0 до $2^{K3}$-1, причем $K_3 < K_1$.

Если значение $2^{K2}$-1 превысит или равно значению ***BASE_REG0***=$2^{K0}$-1, то возникнет событие EVENT0, а если $2^{K2}$-1 превысит или равно значению ***BASE_REG1***=$2^{K0}$-1, то возникнет событие EVENT1.

$$\begin{cases} f_{CLOCK} \leq \dfrac{f_{HSI} \times PREG0*}{(PREG3+1) \times (K_0+1)} \Rightarrow EVENT0 \\ f_{CLOCK} \leq \dfrac{f_{HSI} \times PREG0*}{(PREG3+1) \times (K_0+1)} \Rightarrow EVENT1 \end{cases}$$

При увеличении частоты CLOCK:

- в регистре канала HSI будет возникать значения от 0 до $2^{K2}$-1, причем $K_2 < K_0$;
- в регистре канала CLOCK будет возникать значения от 0 до $2^{K3}$-1, причем $K_3 > K_1$.

Если значение $2^{K3}$-1 превысит или равно значению ***BASE_REG2***=$2^{K1}$-1, то возникнет событие EVENT2, если $2^{K3}$-1 превысит или равно значению ***BASE_REG3***=$2^{K1}$-1, то возникнет событие EVENT3.

$$\begin{cases} f_{CLOCK} \geq \dfrac{f_{HSI} \times (PREG2+1) \times (K_1+1)}{PREG1*} \Rightarrow EVENT2 \\ f_{CLOCK} \geq \dfrac{f_{HSI} \times (PREG2+1) \times (K_1+1)}{PREG1*} \Rightarrow EVENT3 \end{cases}$$

Таблица 114 содержит возможные комбинации настроек для иллюстрации принципов работы блока.

Таблица 114 – Примеры работы блока CLK_MEASURE

| Частота HSI, МГц | Ожидаемая частота CLOCK, МГц | Реальная частота CLOCK, МГц | PREG3 | PREG1 | PREG0 | PREG2 | BASEREG0 | BASEREG1 | BASEREG2 | BASEREG3 | Пример |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | 60 | 60 | 1 | 4 | 29 | 9 | 4 | 4 | 8 | 8 | K0 = 2<br>K1 = 3<br>Нет событий |
| 8 | 60 | 80 | 1 | 4 | 29 | 9 | 4 | 4 | 8 | 8 | K0 = 2<br>K1 = 3<br>K2 = 1.5<br>K3 = 4<br>EVENT2<br>EVENT3<br>Высокая частота |
| 8 | 60 | 30 | 1 | 4 | 29 | 9 | 4 | 4 | 8 | 8 | K0 = 2<br>K1 = 3<br>K2 = 4<br>K3 = 1,5<br>EVENT0<br>EVENT1<br>Низкая частота |

Также можно программно считать максимальное значение регистра ***SHIFT_REG*** каждого из каналов, которое он достигал. При чтении регистра ***SHIFT_REG*** считывается теневой регистр, в котором отражается максимальное достигнутое регистром ***SHIFT_REG*** значение за период с последнего сброса теневого регистра. Сброс теневого регистра осуществляется программно, записью управляющего бита EN=0.

## 18.11 Контроллер генератора случайных чисел (RANDOM_CNTR)

Раздел находится в разработке.

Описание регистров контроллера генератора шума - см. подраздел 21.6 "Описание регистров контроллера генератора шума (NOISE_GEN_CNTR).

## 18.12 Контроллер USART ISO7816 (USART_CNTR)

Универсальный синхронно-асинхронный приемопередатчик (Universal Synchronous Asynchronous Receiver Transmitter, USART) предлагает гибкие средства полнодуплексного обмена с внешним оборудованием, требующим стандартного формата кодирования данных NRZ. Модуль USART поддерживает широкий диапазон скорости передачи данных с использованием генератора дробной скорости передачи данных.

Модуль поддерживает синхронную одностороннюю передачу данных и полудуплексную однопроводную передачу данных. Также поддерживается протокол Smartcard и операции модема CTS/RTS.

### 18.12.1 *Особенности модуля USART*

Модуль USART имеет следующие характеристики:

- полный дуплекс, асинхронный обмен данными;
- стандартный формат данных NRZ (Mark/Space);
- конфигурируемый метод передискретизации на 16 или 8 для обеспечения гибкости между скоростью и допуском по тактовой частоте;
- системы генератора дробной скорости передачи данных:
  - программируемая скорость передачи и приема данных (значение скорости передачи данных при максимальной частоте шины APB см. в спецификациях);
- программируемая длина слова данных (8 или 9 бит);
- Поддержка 2 стоп-бит;
- тактовый выход передатчика для синхронной передачи данных;
- возможность эмуляции смарт-карты (режим Smartcard):
  - интерфейс Smartcard поддерживает асинхронный протокол для смарт-карт в соответствии со стандартом ISO 7816-3;
  - стоп-биты 0.5, 1.5 для интерфейса Smartcard;
- однопроводная полудуплексная передача данных;
- отдельные биты разрешения (enable) для передатчика и приемника;
- флаги обнаружения передачи:
  - буфер приемника полон;
  - буфер передатчика пуст;
  - флаг завершения передачи;
- проверка четности:
  - передача бита четности;
  - проверка четности принятого байта данных;
- флаги обнаружения ошибки:
  - переполнение;
  - обнаружение шума;
  - ошибка фрейма;
  - ошибка четности;
- прерывания с флагами:
  - изменение состояния CTS;
  - регистр данных передатчика пуст;
  - передача завершена;
  - регистр данных приемника заполнен;
  - ошибка переполнения;
  - ошибка фрейма;
  - обнаружение шума;
  - ошибка четности.

### 18.12.2 *Функциональное описание USART*

Интерфейс подключается к внешним приборам при помощи трех выводов (см. рисунок 205). Любая двунаправленная передача данных USART требует минимум два вывода: вход для данных (RX) и выход для передачи данных (TX):

- **RX**: вход для последовательных принимаемых данных. Для восстановления данных используется техника передискретизации, чтобы отделить нужные входящие данные от шума;
- **TX**: выход для передачи данных. Когда передатчик запрещен, вывод возвращается в состояние, заданное конфигурацией порта ввода-вывода (I/O).

Когда передатчик разрешен, но никакие данные не передаются, на выводе TX устанавливается высокий уровень (логическая «1»). В однопроводном режиме и режиме смарт-карты, данный порт ввода-вывода используется для передачи и приема данных (SW_RX).

При помощи данных выводов происходит передача и прием последовательных данных в нормальном режиме работы модуля USART, в виде фреймов, содержащих:

- сигнал ожидания линии (Idle) до передачи или приема;
- стартовый бит;
- слово данных (8 или 9 бит), LSB передается первым;
- 0.5, 1.5, 2 стоп-бита, показывающие, что фрейм завершен;
- данный интерфейс использует дробный генератор скорости передачи данных – с 12-битной мантиссой и 4-битной дробной частью;
- регистр статуса (USART_SR);
- регистр данных (USART_DR);
- регистр скорости передачи (USART_BRR) – 12-битная мантисса и 4-битная дробная часть;
- регистр защитного интервала (Guardtime register) (USART_GTPR) в случае использования режима Smartcard.

Описание регистров и их бит приведено в разделе «Регистры USART».

Для работы в синхронном режиме требуется дополнительный вывод:

- **CK:** выход тактов передатчика. На этот вывод выдаются такты данных для синхронной передачи, соответствующей режиму ведущий SPI (нет тактовых импульсов в стартовых и стоп- битах, и программная опция отправки тактового импульса на последнем бите данных). Параллельно данные могут синхронно приниматься по выводу RX. Это можно использовать для управления внешними периферийными устройствами, у которых есть регистры сдвига (например, LCD драйверы). Фаза и полярность тактов выбирается программно. В режиме Smartcard, вывод CK обеспечивает такты для смарт-карты.

В режиме аппаратного управления потоком данных требуются дополнительно выводы:

- **CTS:** сигнал Clear To Send блокирует передачу данных по окончании текущей передачи при высоком уровне сигнала;
- **RTS:** сигнал Request to send показывает готовность USART к приему данных (при низком уровне сигнала).

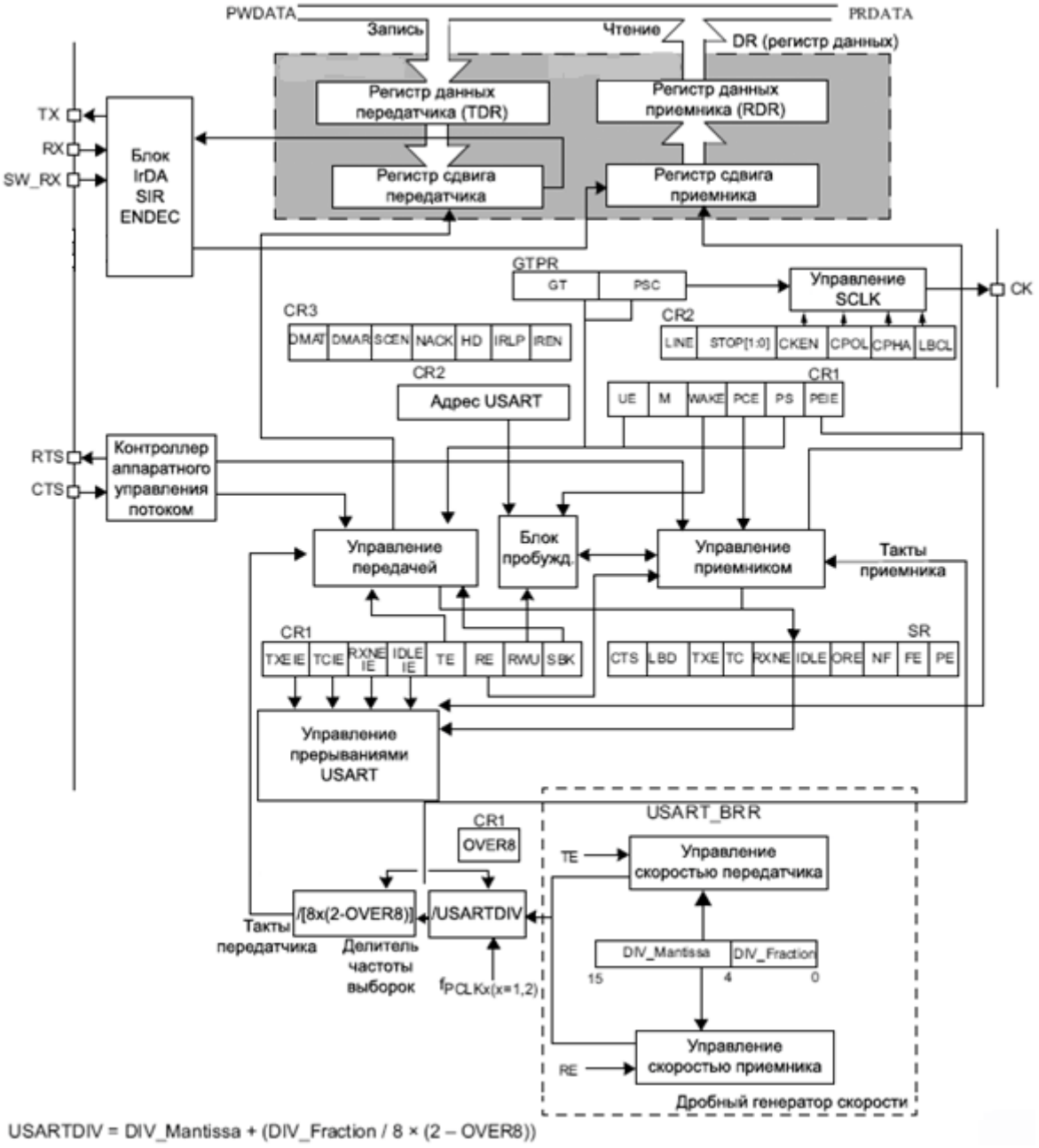



Рисунок 205 – Блок-схема модуля USART

### 18.12.2.1 Описание символов USART

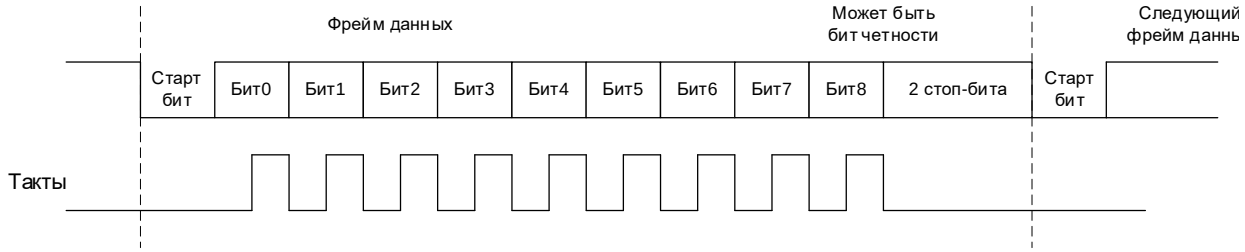
Для передачи можно выбрать длину слова 8 или 9 бит путем программирования бита M в регистре USART_CR1 (см. Рисунок 205).

Вывод TX находится в состоянии логического «0» во время передачи стартового бита, и в состоянии логической «1» во время передачи стоп-бита.

Передача и прием управляются генератором скорости передачи. Такты для передатчика и приемника генерируются, когда установлен соответствующий бит разрешения.

Ниже приведено подробное описание каждого блока.

**Длина слова 9 бит (бит М установлен), 2 стоп-бита**




**Длина слова 8 бит (бит М сброшен), 2 стоп-бита**

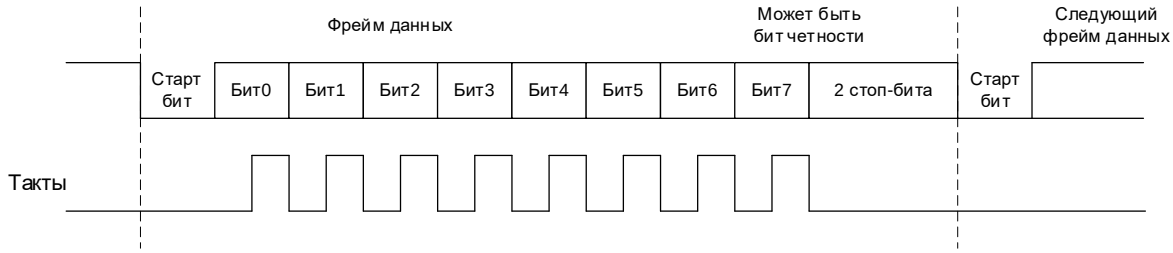



Рисунок 206 – Программирование длины слова

### 18.12.2.2 Передатчик

Передатчик может отправлять слова данных по 8 или 9 бит, в зависимости от статуса бита М.

Когда установлен бит разрешения передачи (TE), данные из регистра сдвига выводятся на вывод TX, и соответствующие тактовые импульсы выводятся на вывод CK.

***Передача символа***

Во время передачи USART-данные сдвигаются на вывод TX, младший значащий бит (LSB) идет первым. В данном режиме, регистр USART_DR состоит из буфера (TDR) между внутренней шиной и регистром сдвига передачи (см. рисунок 205).

Каждому символу предшествует стартовый бит низкого уровня (логический 0) в течение длительности одного бита. Символ завершается конфигурируемым количеством стоп-битов.

Модуль USART поддерживает следующие стоп-биты: 0.5, 1.5 и 2 стоп-бита.

Примечание – бит TE не должен сбрасываться по время передачи данных. Сброс бита TE во время передачи приведет к повреждению данных на выводе TX, так как счетчики генератора скорости остановятся. Текущие передаваемые данные будут потеряны.

После разрешения бита TE будет отправлен фрейм idle.

***Конфигурируемые стоп-биты***

Количество передаваемых стоп-бит с каждым символом может быть запрограммировано в Регистре Управления 2 битами 13, 12.

- *2 стоп-бита*: Это значение поддерживается в обычном режиме USART, однопроводном режиме и режиме модема;
- *0.5 стоп-бит*: для использования при приеме данных в режиме Smartcard;
- *1.5 стоп-бит*: для использования при передаче и приеме данных в режиме Smartcard.

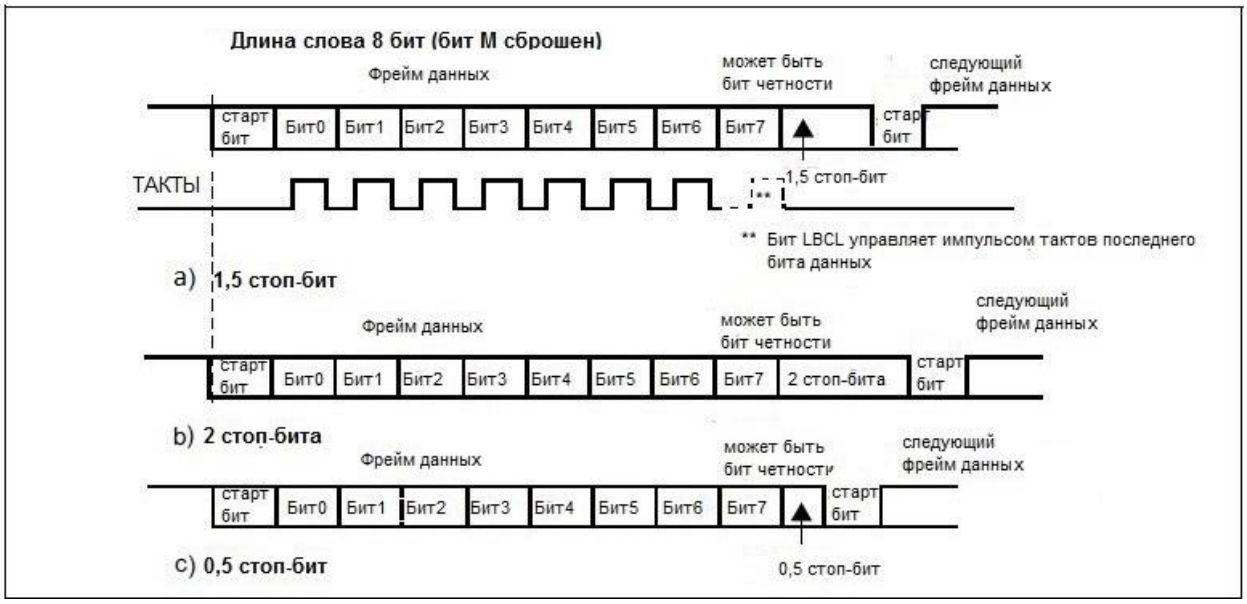



Рисунок 207 – Конфигурируемые стоп-биты

Процедура программирования:

1. Разрешить USART записью в 1 бита UE в регистре USART_CR1;
2. Запрограммировать бит M в регистре USART_CR1, чтобы определить длину слова данных;
3. Запрограммировать количество стоп-бит в регистре USART_CR2;
4. Выбрать необходимую скорость передачи при помощи регистра USART_BRR;
5. Установить бит TE в регистре USART_CR1;
6. Записать данные для передачи в регистр USART_DR (это сбросит бит TXE). Повторить данную операцию для каждого передаваемого символа в случае использования одиночного буфера;
7. После записи последних данных в регистр USART_DR, подождать, пока TC станет равным 1. Это покажет, что передача последнего фрейма завершена. Например, это требуется, когда модуль USART отключен или переходит в режиме Halt, чтобы избежать повреждения последней передачи.

### *Однобайтный обмен*

Бит TXE всегда сбрасывается при записи в регистр данных.

Бит TXE устанавливается аппаратно, и указывает, что:

- данные были перемещены из регистра TDR в регистр сдвига, и началась передача;
- регистр TDR пуст;
- следующие данные могут быть записаны в регистр USART_DR без риска перезаписи предыдущих данных.

Данный флаг генерирует прерывание, если установлен бит TXE.

Во время передачи данных команда записи в регистр USART_DR помещает данные в регистр TDR, откуда затем данные копируются в регистр сдвига при завершении текущей передачи.

Если передачи данных не происходит, команда записи в регистр USART_DR помещает данные напрямую в регистр сдвига, начинается передача данных, и бит TXE незамедлительно устанавливается.

Если передается фрейм (после стоп-бита) и бит TXE установлен, бит TC устанавливается в высокий уровень (логическая «1»). Генерируется прерывание, если бит TCIE установлен в регистре USART_CR1.

После записи последних данных в регистр USART_DR важно дождаться, когда бит TC установится в 1 перед запретом USART или переходом микроконтроллера в режим пониженного потребления (см. рисунок 208).

Бит TC сбрасывается следующей программной последовательностью:

1. Чтение из регистра USART_SR;
2. Запись в регистр USART_DR.

Примечание – Бит TC может быть также сброшен при помощи записи «0». Рекомендуется использовать данную последовательность для сброса бита только в случае мультибуферного обмена данными.

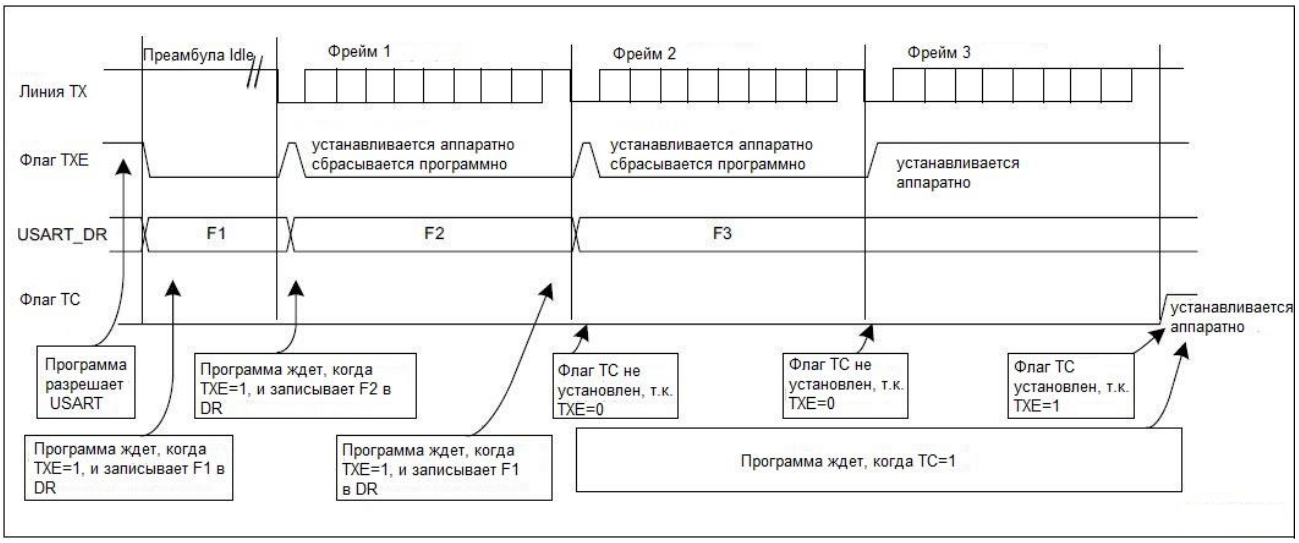



Рисунок 208 – Состояние TC/TXE во время передачи

#### 18.12.2.3 Приемник

USART может принимать слово данных разрядностью 8 или 9 бит, в зависимости от значения бита M в регистре USART_CR1.

***Обнаружение стартового бита***

Последовательность обнаружения стартового бита одинаковая при передискретизации сигнала с кратностью 16 или 8.

В приемопередатчике USART стартовый бит детектируется, когда распознана определенная последовательностью выборки сигнала: 1110X0X0X0000.

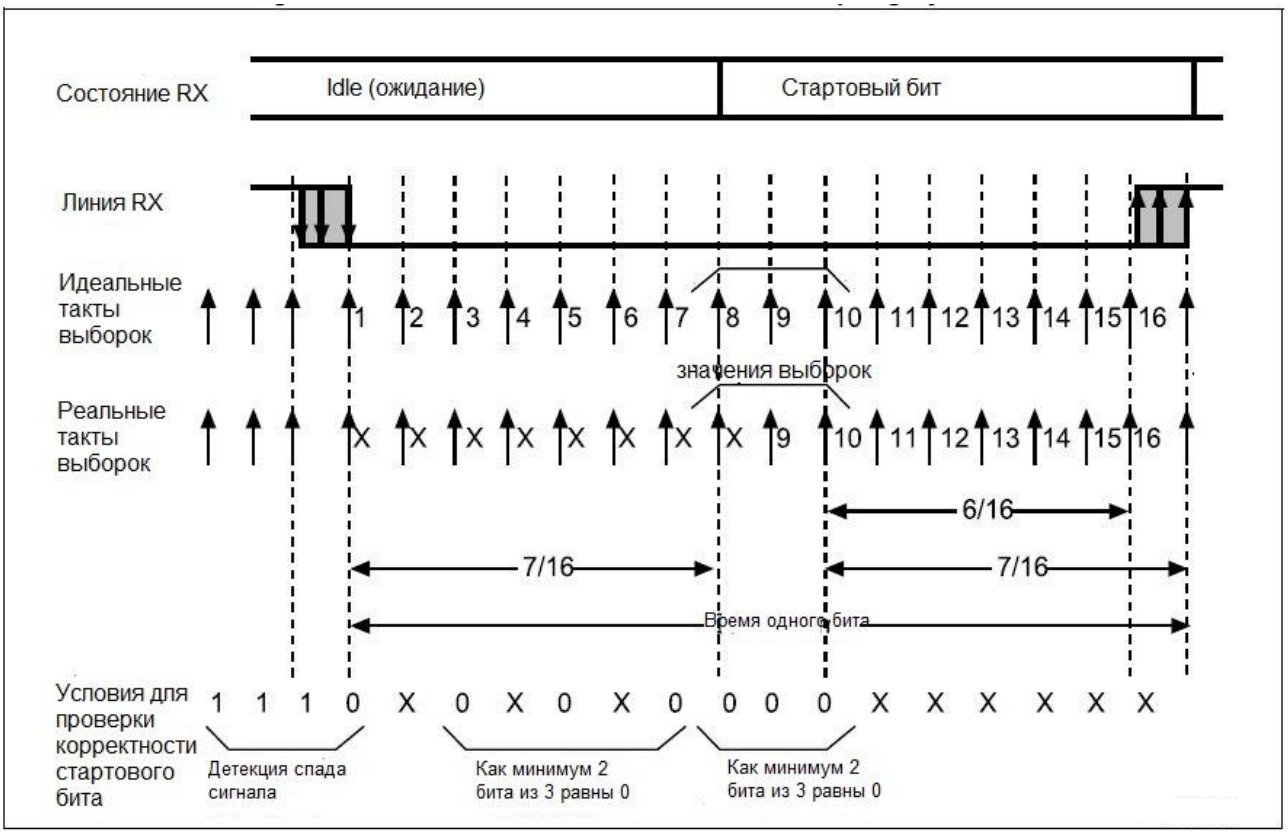



Рисунок 209 – Обнаружение стартового бита при передискретизации на 16 и 8

Примечание – Если последовательность не завершена, обнаружение стартового бита прекращается, и приемник переходит в состояние ожидания (idle) (никакие флаги не установлены) и ожидает заднего фронта.

Стартовый бит подтвержден (флаг RXNE установлен, генерируется прерывание при RXNEIE=1), если 3 бита выборки равны 0 (первая выборка на 3, 5 и 7 бите находит три бита в логическом 0, а вторая выборка на 8, 9 и 10 битах также находит три бита в 0).

Стартовый бит подтвержден (флаг RXNE установлен, генерируется прерывание при RXNEIE=1), но устанавливается флаг NF, если при обеих выборках как минимум 2 из 3 бит установлены в 0 (выборка на 3, 5 и 7 битах и выборка на 8, 9 и 10 битах). Если данное условие не выполняется, не происходит обнаружение стартового бита, и приемник возвращается в состояние ожидания (idle) (нет установленных флагов).

Если при одной из выборок (выборка на 3, 5 и 7 битах или выборка на 8, 9 и 10 битах) 2 из 3 бит находятся в логическом 0, стартовый бит подтверждается, но устанавливается флаг NF.

***Прием символа***

При приеме данных данные перемещаются на вывод RX, начиная с LSB. В данном режиме регистр USART_DR состоит из буфера (RDR) между внутренней шиной и регистром сдвига приемника.

Процедура программирования:

1. Разрешить USART путем записи в 1 бита UE в регистре USART_CR1.
2. Запрограммировать бит M в регистре USART_CR1 для определения длины слова данных.
3. Запрограммировать количество стоп-бит в регистре USART_CR2.
4. Выбрать необходимую скорость передачи данных при помощи регистра USART_BRR
5. Установить бит RE в регистре USART_CR1. Это активирует приемник, и он начнет обнаружение стартового бита.

Когда принят символ:

- устанавливается бит RXNE. Он показывает, что содержимое сдвигового регистра передано в RDR. Другими словами, данные были получены и могут быть прочитаны (а также связанные с ними флаги ошибок);
- генерируется прерывание, если установлен бит RXNEIE;
- флаг ошибки устанавливается, если во время приема была обнаружена ошибка фрейма, шума или переполнения;
- в мультибуферном режиме бит RXNE устанавливается после каждого принятого байта и сбрасывается чтением в регистре данных;
- в режиме одиночного буфера бит RXNE сбрасывается программно при чтении регистра;
- USART_DR. Флаг RXNE сбрасывается записью 0. Бит RXNE должен быть сброшен перед окончанием приема следующего символа, чтобы избежать ошибки переполнения.

Примечание – бит RE не должен сбрасываться при приеме данных. Если бит RE запрещен во время приема, прием текущего байта будет прерван.

***Ошибка переполнения (overrun)***

Ошибка переполнения возникает, когда получен символ, но бит RXNE не сброшен. Данные не могут быть переданы из регистра сдвига в регистр RDR, пока бит RXNE не будет сброшен.

Флаг RXNE устанавливается после каждого полученного байта. Ошибка переполнения возникает, если флаг RXNE установлен, когда приняты следующие данные. При появлении ошибки переполнения:

- бит ORE будет установлен;
- содержимое регистра RDR не будет потеряно. Предыдущие данные будут доступны при чтении регистра USART_DR;
- регистр сдвига будет перезаписан. В этот момент любые данные полученные во время ошибки переполнения будут потеряны;
- генерируется прерывание, если либо установлены биты RXNEIE или EIE;
- бит ORE сбрасывается чтением регистра USART_SR, за которым следует операция чтения регистра USART_DR.

Примечание – установленный бит ORE показывает, что как минимум один элемент данных потерян. Существует два варианта:

- если RXNE=1, то последние корректные данные сохраняются в регистр RDR и могут быть прочитаны,
- если RXNE=0, это означает, что последние корректные данные уже были прочитаны, поэтому нечего считывать из регистра RDR. Это может произойти, когда последние корректные данные были прочитаны из регистра RDR одновременно с получением новых данных. Также это может произойти, когда новые данные приняты во время последовательности чтения (между доступом на чтение регистра USART_SR и доступом на чтение регистра USART_DR).

***Выбор правильного метода передискретизации***

Приемник использует разные техники передискретизации, конфигурируемые пользователем (кроме синхронного режима), для восстановления данных путем отделения друг от друга входящих данных и шума.

Метод передискретизации можно выбрать путем программирования бита OVER8 в регистре USART_CR1, передискретизация может быть или кратная 16-ти или 8-ми тактам скорости (Рисунки 210 и 211).

В зависимости от применения выберите:

- передискретизацию на 8 (OVER8=1) для достижения увеличенной скорости (до $f_{PCLK}/8$). В данном случае максимальный допуск на отклонение тактов для приемника снижается (см. подраздел «Допуск ухода тактовой частоты для приемника USART»);

– передискретизацию на 16 (OVER8=0) для увеличения допуска приемника к отклонению тактов. В данном случае максимальная скорость ограничена $f_{PCLK}/16$.

Программирование бита ONEBIT в регистре USART_CR3 выбирает метод, используемый для оценки логического уровня. Есть две опции:

– мажоритарная выборка из 3 выборок по центру принятого бита. В данном случае, если эти 3 выборки не одинаковые, установится бит NF;
– одна выборка в центре принятого бита.

В зависимости от применения:

– выберите метод мажоритарности из 3-х выборок (ONEBIT=0) при работе в условиях шума и отклоните данные, если обнаружен шум, потому что это показывает, что во время выборки произошел сбой.
– выберите метод одиночной выборки (ONEBIT=1), если нет шумов на линии, чтобы увеличить допуск на отклонение тактов приемника (см. Допуск на отклонения тактовой частоты USART приемника). В данном случае бит NF не будет установлен.

Когда во фрейме обнаружен шум:

– бит NF будет установлен по переднему фронту бита RXNE;
– некорректные данные будут перемещены из регистра сдвига в регистр USART_DR;
– не будет генерироваться прерывание в случае однобайтной передачи данных. Однако, этот бит будет установлен одновременно с битом RXNE, который сам генерирует прерывание. В случае мультибуферного обмена данными прерывание возникнет, если установлен бит EIE в регистре USART_CR3.

Бит NF сбрасывается чтением регистра USART_SR, после чего следует чтение регистра USART_DR.

Примечание – Передискретизация на 8 недоступна в режимах Smartcard. В данном режиме бит OVER8 аппаратно сбрасывается в 0.

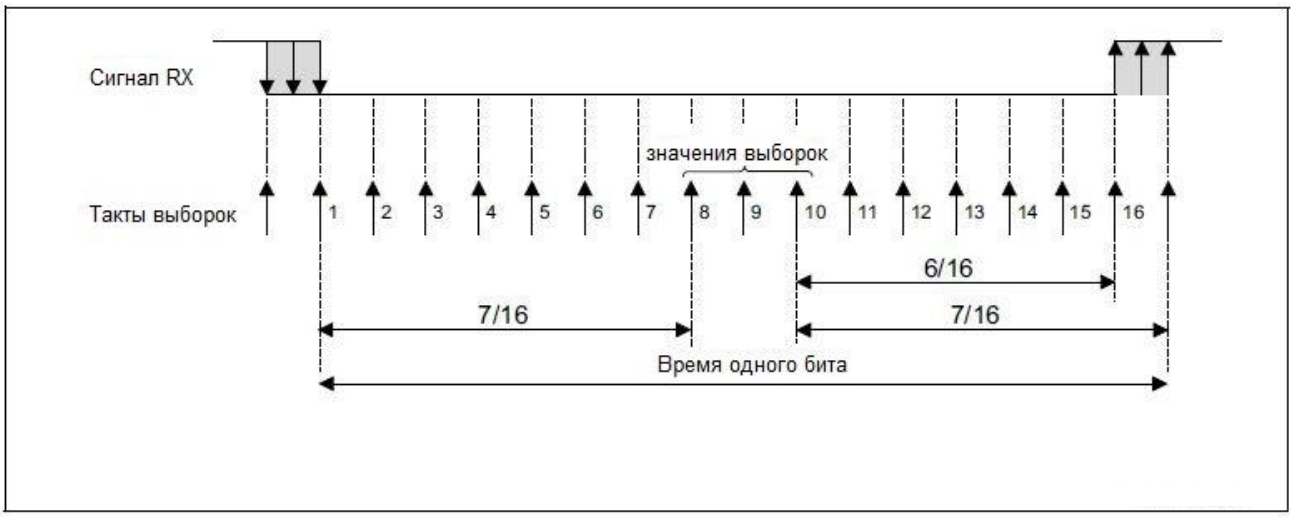



Рисунок 210 – Выборка данных при передискретизации на 16

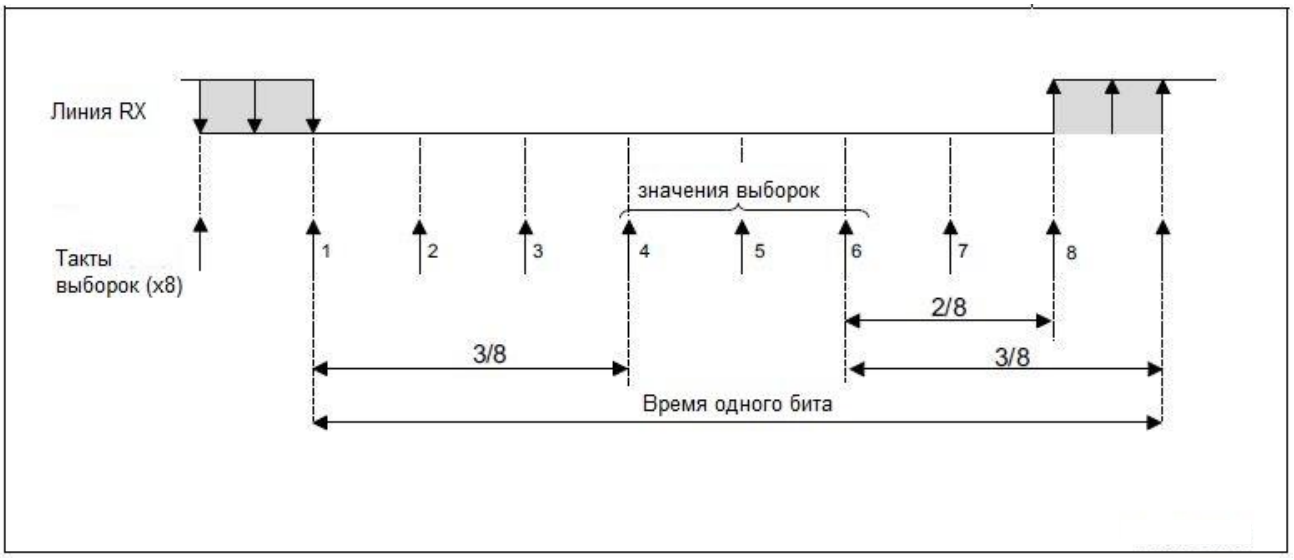



Рисунок 211 – Выборка данных при передискретизации на 8

Таблица 115 – Обнаружение шумов для считанных данных

| Значение | NF статус | Значение принятого бита |
|---|---|---|
| 000 | 0 | 0 |
| 001 | 1 | 0 |
| 010 | 1 | 0 |
| 011 | 1 | 1 |
| 100 | 1 | 0 |
| 101 | 1 | 1 |
| 110 | 1 | 1 |
| 111 | 0 | 1 |

***Ошибка фрейма (framing error)***

Ошибка фрейма обнаруживается, когда стоп-бит не распознан во время приема данных в ожидаемый промежуток времени из-за рассинхронизации или из-за чрезмерного шума.

Когда обнаружена ошибка фрейма, происходит следующее:

- бит FE устанавливается аппаратно;
- некорректные данные передаются из регистра сдвига в регистр USART_DR;
- в случае однобайтного обмена никакое прерывание не генерируется. Однако, данный бит устанавливается одновременно с битом RXNE, который сам генерирует прерывание. В случае мультибуферного обмена прерывание генерируется, если установлен бит EIE в регистре USART_CR3.

Бит FE сбрасывается чтением регистра USART_SR, после чего следует чтение регистра USART_DR.

***Конфигурируемые стоп-биты во время приема***

Количество принимаемых стоп-бит можно конфигурировать в Регистре Управления 2. Это может быть 2 бита в нормальном режиме и 0,5 или 1,5 в режиме Smartcard.

1 ***0,5 стоп-бит (прием в режиме Smartcard)***: для данного режима выборки не делаются. Как следствие, при выборе 0,5 стоп-бита ошибка фрейма или сигнал break не могут быть обнаружены.

2 ***1,5 стоп-бита (режим Smartcard)***: при передаче данных в режиме smartcard, прибор должен проверить корректность отправляемых данных. Таким образом, блок приемника должен быть разрешен (RE=1 в регистре USART_CR1), а стоп-бит проверяется на наличие ошибки четности. В случае наличия ошибки четности, смарт-карта подтягивает сигнал данных в логический 0 во время выборки (сигнал NACK), который опознается как ошибка фрейма. После чего флаг FE устанавливается с RXNE по окончанию 1,5 стоп-бита. Анализ уровня для 1,5 стоп-бита осуществляется на 16, 17 и 18 выборках (1 период скорости после начала стоп-бита). 1,5 стоп-бит может быть разложен на 2 части: одна часть из 0,5 периода скорости, во время которой ничего не происходит, и вторая часть – 1 период нормального стоп-бита, во время которого происходит анализ сигнала. Подробнее см. подраздел «Режим Smartcard».

3 ***2 стоп-бита***: анализ для 2 стоп-бит осуществляется на 8, 9 и 10 выборках первого стоп-бита. Если была обнаружена ошибка фрейма во время первого стоп-бита, устанавливается флаг ошибки. Второй стоп-бит не проверяется на наличие ошибки фрейма. Флаг RXNE устанавливается по окончанию первого стоп-бита.

#### 18.12.2.4 Дробный генератор скорости

Скорость обмена для приемника и передатчика (Rx и Tx) устанавливается в одинаковое значение, программируемое коэффициентами Mantissa (целая часть) и Fraction (дробная часть) делителя USARTDIV.

Скорость для стандартного USART (включая режим SPI) вычисляется по формуле

$$\text{Tx/Rx baud} = \frac{f_{CK}}{8 \cdot (2 - OVER8) \cdot USARTDIV}. \quad (15)$$

Скорость для режимов Smartcard вычисляется по формуле

$$\text{Tx/Rx baud} = \frac{f_{CK}}{16 \times USARTDIV}, \quad (16)$$

где USARTDIV – это число с фиксированной запятой без знака, запрограммированное в регистре USART_BRR.

- Если OVER8=0, дробная часть кодируется 4 битами и программируется битами DIV_fraction [3:0] в регистре USART_BRR
- Если OVER8=1, дробная часть кодируется 3 битами и программируется битами DIV_fraction [2:0] в регистре USART_BRR, бит DIV_fraction [3] должен быть сброшен.

Примечание – Счетчики скорости обновляются в регистрах скорости после операции записи в регистр USART_BRR. Следовательно, значение регистра скорости не должно изменяться во время активного обмена данными.

***Примеры расчета USARTDIV для значений регистра USART_BRR при OVER8=0***

Пример 1:

Если DIV_Mantissa = 0d27 и DIV_Fraction = 0d12 (USART_BRR = 0x1BC), тогда
Mantissa (USARTDIV) = 0d27
Fraction (USARTDIV) = 12/16 = 0d0.75
Следовательно, USARTDIV = 0d27.75

Пример 2:

Для программирования USARTDIV = 0d25.62
получается:
DIV_Fraction = 16 • 0d0.62 = 0d9.92
Ближайшее действительное число 0d10 = 0xA
DIV_Mantissa = mantissa (0d25.620) = 0d25 = 0x19
Тогда, USART_BRR = 0x19A, следовательно, USARTDIV = 0d25.625

Пример 3:

Для программирования USARTDIV = 0d50.99
получается:
DIV_Fraction = 16 • 0d0.99 = 0d15.84
Ближайшее действительное число 0d16 = 0x10 => переполнение DIV_frac[3:0] => должен быть добавлен перенос (carry) к мантиссе
DIV_Mantissa = mantissa (0d50.990 + carry) = 0d51 = 0x33
Тогда, USART_BRR = 0x330, следовательно, USARTDIV = 0d51.000

***Примеры расчета USARTDIV для значений регистра USART_BRR при OVER8=1***

Пример 1:

Если DIV_Mantissa = 0x27 и DIV_Fraction[2:0]= 0d6 (USART_BRR = 0x1B6), тогда
Mantissa (USARTDIV) = 0d27
Fraction (USARTDIV) = 6/8 = 0d0.75
Therefore USARTDIV = 0d27.75

Пример 2:

Для программирования USARTDIV = 0d25.62
получается:
DIV_Fraction = 8 • 0d0.62 = 0d4.96
Ближайшее действительное число 0d5 = 0x5
DIV_Mantissa = mantissa (0d25.620) = 0d25 = 0x19
Тогда, USART_BRR = 0x195 => USARTDIV = 0d25.625

Пример 3:

Для программирования USARTDIV = 0d50.99

Получается:
DIV_Fraction = 8 • 0d0.99 = 0d7.92
Ближайшее действительное число 0d8 = 0x8 => переполнение DIV_frac[2:0] => должен быть добавлен перенос (carry) к мантиссе
DIV_Mantissa = mantissa (0d50.990 + carry) = 0d51 = 0x33
Тогда, USART_BRR = 0x0330 => USARTDIV = 0d51.000

Таблица 116 – Определение погрешности для запрограммированных скоростей передачи при $f_{PCLK}$ = 130 МГц(1)

| № | Требуемая скорость | Передискретизация на 16 (OVER8=0) | | | Передискретизация на 8 (OVER8=1) | | |
|---|---|---|---|---|---|---|---|
| | | Реальная скорость | Значение в регистре скорости передачи USART_BRR | Ошибка, % | Реальная скорость | Значение в регистре скорости передачи USART_BRR | Ошибка, % |
| 1 | 2,4 Кбит/с | 2,4 Кбит/с | 3385,4375 | 0 | 2,4 Кбит/с | NA | NA |
| 2 | 9,6 Кбит/с | 9,6 Кбит/с | 846,375 | 0 | 9,6 Кбит/с | 1692,75 | 0 |
| 3 | 19,2 Кбит/с | 19,2 Кбит/с | 423,1875 | 0 | 19,2 Кбит/с | 846,375 | 0 |
| 4 | 57,6 Кбит/с | 57,599 Кбит/с | 141,0625 | 0 | 57,599 Кбит/с | 282,125 | 0 |
| 5 | 115,2 Кбит/с | 115,248 Кбит/с | 70,5 | 0,04 | 115,248 Кбит/с | 141 | 0,04 |
| 6 | 230,4 Кбит/с | 230,496 Кбит/с | 35,25 | 0,04 | 230,496 Кбит/с | 70,5 | 0,04 |
| 7 | 460,8 Кбит/с | 460,993 Кбит/с | 17,625 | 0,04 | 460,993 Кбит/с | 35,25 | 0,04 |
| 8 | 896 Кбит/с | 896,552 Кбит/с | 9,0625 | 0,06 | 896,552 Кбит/с | 18,125 | 0,06 |
| 9 | 921,6 Кбит/с | 921,986 Кбит/с | 8,8125 | 0,04 | 921,986 Кбит/с | 17,625 | 0,04 |
| 10 | 2 Мбит/с | 2 Мбит/с | 4,0625 | 0 | 2 Мбит/с | 8,125 | 0 |
| 11 | 3 Мбит/с | 3,023 Мбит/с | 2,6875 | 0,77 | 3,023 Мбит/с | 5,375 | 0,77 |
| 12 | 4 Мбит/с | 3,939 Мбит/с | 2,0625 | 1,53 | 3,939 Мбит/с | 4,125 | 1,53 |
| 13 | 5 Мбит/с | 5 Мбит/с | 1,625 | 0 | 5 Мбит/с | 3,25 | 0 |
| 14 | 6 Мбит/с | 5,909 Мбит/с | 1,375 | 1,52 | 5,909 Мбит/с | 2,75 | 1,52 |
| 15 | 7 Мбит/с | 6,842 Мбит/с | 1,1875 | 2,26 | 6,842 Мбит/с | 2,375 | 2,26 |
| 16 | 8 Мбит/с | 8,125 Мбит/с | 1 | 1,56 | 8,125 Мбит/с | 2 | 1,56 |
| 17 | 9 Мбит/с | NA | NA | NA | 9,286 Мбит/с | 1,75 | 3,18 |

(1) Чем ниже тактовая частота центрального процессора, тем меньше точность для конкретного значения скорости передачи. Верхний предел достижимой скорости передачи может быть зафиксирован с этими данными.

#### 18.12.2.5 Допуск ухода тактовой частоты для приемника USART

Асинхронный приемник USART корректно работает, только если общее отклонение тактовой частоты меньше, чем допуск приемника USART. Причины, которые способствуют общему отклонению:

- DTRA: отклонение из-за ошибки передатчика (что также включает отклонение от локального тактового генератора передатчика);
- DQUANT: ошибка квантования скорости приемника;
- DREC: отклонение локального генератора тактов приемника;
- DTCL: отклонение из-за линии передачи (обычно из-за того, что приемопередатчики могут давать ассиметричные перепады от 0 к 1 по сравнению с перепадами от 1 к 0).

$$DTRA + DQUANT + DREC + DTCL < \text{допуск приемника } USART$$

Допуск приемника USART для правильного приема данных равен максимально допустимому отклонению и зависит от следующих параметров:

- длина символа 10 или 11 бит, что определяется битом M в регистре USART_CR1
- передискретизация на 8 или 16, что определяется битом OVER8 в регистре USART_CR1
- используется или нет дробная установка скорости
- используется 1 или 3 бита для оцифровки данных, в зависимости от бита ONEBIT в регистре USART_CR3.

Таблица 117 – Допуск приемника USART при DIV_fraction = 0

| **Бит M** | **OVER8 = 0** | | **OVER8 = 1** | |
|---|---|---|---|---|
| | **ONEBIT=0** | **ONEBIT=1** | **ONEBIT=0** | **ONEBIT=1** |
| 0 | 3,75% | 4,375% | 2,50% | 3,75% |
| 1 | 3,41% | 3,97% | 2,27% | 3,41% |

Таблица 118 – Допуск приемника USART при DIV_Fraction, не равным 0

| **Бит M** | **OVER8 = 0** | | **OVER8 = 1** | |
|---|---|---|---|---|
| | **ONEBIT=0** | **ONEBIT=1** | **ONEBIT=0** | **ONEBIT=1** |
| 0 | 3,33% | 3,88% | 2% | 3% |
| 1 | 3,03% | 3,53% | 1,82% | 2,73% |

#### 18.12.2.6 Многопроцессорный обмен

Существует возможность многопроцессорного обмена данными через USART (несколько приемопередатчиков USART, объединенных в одну сеть). Например, один USART может быть ведущим устройством, его выход TX подключается ко входу RX другого приемопередатчика USART. Другие приемопередатчики являются ведомыми, их соответствующие выходы TX логически объединены операцией «логическое И» и подсоединены ко входу RX ведущего устройства.

В многопроцессорных конфигурациях предпочтительно, чтобы только один получатель сообщения активно принимал все сообщение целиком, чтобы уменьшить обработку избыточных данных для приемников, которым эти данные не адресованы.

Неадресованные устройства могут быть переведены в режим «молчания» (mute mode) при помощи функции приостановки. В данном режиме:

- никакой статус бит приема не может быть установлен;
- все прерывания приема запрещены;
- бит RWU в регистре USART_CR1 установлен в 1. Бит RWU может управляться автоматически аппаратно или может быть записан программой при определенных условиях.

USART может войти в режим молчания или выйти из него при помощи одного из двух методов, в зависимости от значения бита WAKE в регистре USART_CR1:

- обнаружение линии Idle, если бит WAKE сброшен;
- обнаружение сигнала метки адреса (Address Mark), если бит WAKE установлен.

***Обнаружение линии Idle (WAKE=0)***

USART переключится в режим молчания, если бит RWU установлен в 1.

Приемопередатчик выйдет из режима молчания, когда будет обнаружен фрейм Idle. После чего бит RWU сбрасывается аппаратной частью, но бит IDLE в регистре USART_SR не устанавливается. Бит RWU может быть записан в 0 программно.

На рисунке 212 показан пример поведения устройства в режиме молчания с использованием обнаружения линии Idle.

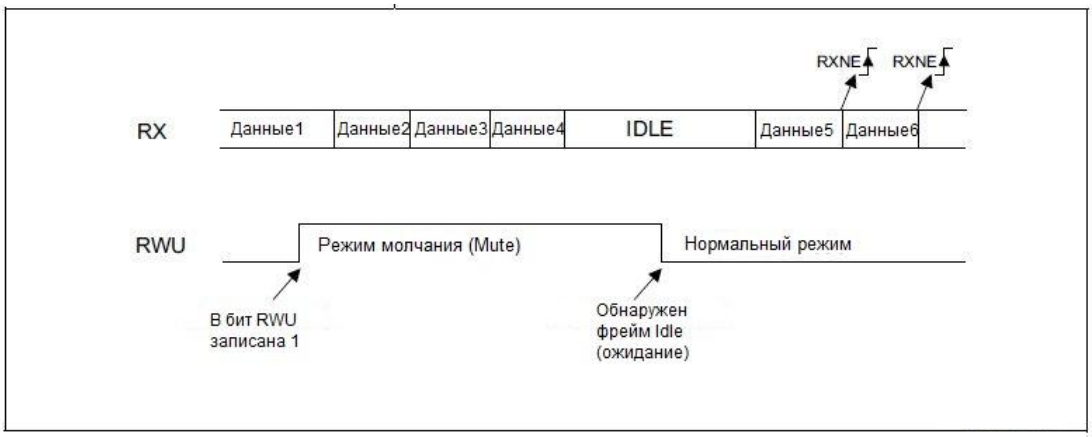



Рисунок 212 – Режим молчания (Mute) с обнаружением линии Idle

***Обнаружение метки адреса (Address mark) (WAKE=1)***

В данном режиме, байты распознаются как адреса, если MSB равен 1, в другом случае они распознаются как данные. В байте адреса адрес приемника помещается в 4 LSB. Приемник сравнивает это 4-битное слово с собственным адресом, запрограммированным в битах ADD в регистре USART_CR2.

USART переключится в режим молчания, когда будет получен символ адреса, который не соответствует запрограммированному режиму. В данном случае, бит RWU устанавливается аппаратно. Флаг RXNE не установится для данного байта адреса, и не будет сгенерировано прерывание, так как приемопередатчик перейдет в режим молчания.

Приемопередатчик выйдет из режима молчания, когда будет принят символ адреса, совпадающий с запрограммированным адресом. После чего бит RWU будет сброшен и последующие биты будут приняты. Символ адреса будет установлен в бите RXNE, так как бит RWU был сброшен.

Бит RWU может быть записан в 0 или 1, когда буфер приемника не содержит данных (RXNE=0 в регистре USART_SR). Иначе попытка записи будет проигнорирована.

Пример поведения приемопередатчика в режиме молчания с использованием режима обнаружения метки адреса приведен на рисунке 213.

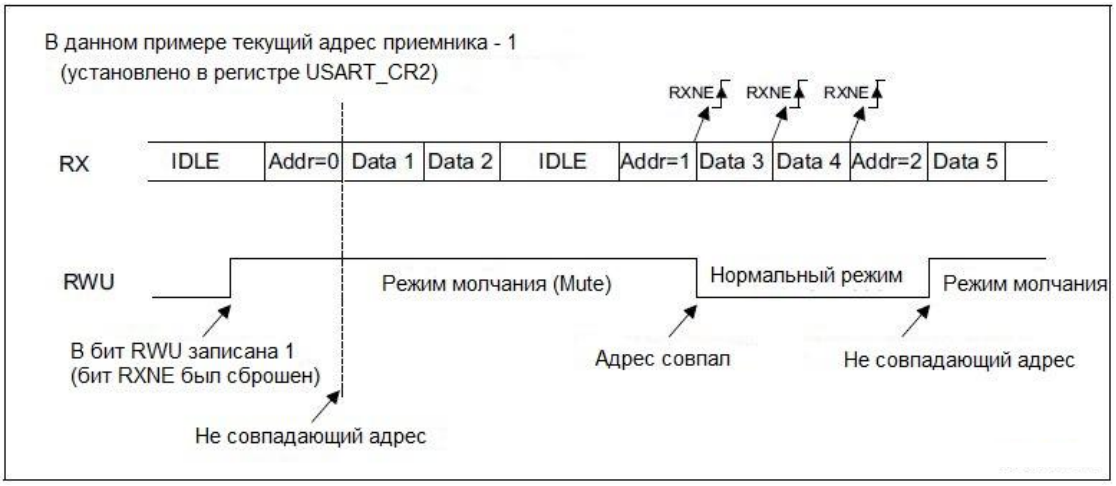



Рисунок 213 – Режим молчания (Mute) с обнаружением метки адреса

#### 18.12.2.7 Контроль четности

Контроль четности бит (генерация бита четности при передаче и контроль бита четности при приеме) разрешается установкой бита PCE в регистре USART_CR1. В зависимости от длины фрейма, определяемой битом M, возможные форматы фрейма приемопередатчика USART приведены в таблице 119.

Таблица 119 – Форматы фрейма

| Бит M | Бит PCE | Фрейм USART (1) |
|---|---|---|
| 0 | 0 | \| SB \| 8 бит данных \| STB \| |
| 0 | 1 | \| SB \| 7 бит данных \| PB \| STB \| |
| 1 | 0 | \| SB \| 9 бит данных \| STB \| |
| 1 | 1 | \| SB \| 8 бит данных \| PB \| STB \| |
| (1) SB: стартовый бит, STB: стоп-бит, PB: бит четности | | |

***Проверка на четность***

Бит четности вычисляется так, чтобы получить четную сумму всех «1» во фрейме из 7 или 8 бит (в зависимости от значения бита M) и бита четности.

Например, data=00110101; установлено 4 бита => бит четности = 0, если выбрана проверка на четность (бит PS в USART_CR1 = 0).

***Проверка на нечетность***

Бит четности вычисляется так, чтобы получить нечетную сумму всех «1» во фрейме из 7 или 8 бит (в зависимости от значения бита M) и бита четности.

Например, data=00110101; установлено 4 бита => бит четности = 1, если выбрана проверка на нечетность (бит PS в USART_CR1 = 1).

***Проверка четности при приеме***

Если проверка четности показала ошибку данных, устанавливается флаг PE в регистре USART_SR, и генерируется прерывание, если в регистре USART_CR1 установлен бит PEIE. Флаг PE сбрасывается программной последовательностью (чтение регистра статуса, за которым следует чтение или запись регистра данных USART_DR).

Примечание – в случае пробуждения по метке адреса: MSB используется для идентификации адреса, а не бит четности. Приемник не проверяет бит четности данных адреса (бит PE не устанавливается в случае ошибки четности).

***Генерация бита четности при передаче***

Если в регистре USART_CR1 установлен бит PCE, то MSB данных, записанных в регистре данных, будет передан, но он будет изменен битом четности (четное количество «1», если выбрана проверка на четность (PS=0) или нечетное количество «1», если выбрана проверка на нечетность (PS=1)).

Примечание – Программная часть, которая контролирует передачу, может активировать программную последовательность для сброса флага PE (чтение регистра статуса, за которым следует доступ на чтение или запись регистра данных). При работе в полудуплексном режиме, в зависимости от программной части, это может привести к неожиданному сбросу флага PE.

#### 18.12.2.8 Синхронный режим USART

Синхронный режим выбирается записью бита CLKEN в «1» в регистре USART_CR2.

В синхронном режиме следующие биты должны быть сброшены:

- SCEN и HDSEL в регистре USART_CR3.

USART дает возможность пользователю управлять двунаправленным синхронным последовательным обменом данных в режиме ведущего приемопередатчика. Вывод CK является выходом тактов передатчика USART. Во время стартового и стоп-битов никакие тактовые импульсы не посылаются на вывод CK. В зависимости от состояния бита LBCL в регистре USART_CR2 тактовые импульсы будут или не будут генерироваться во время последнего достоверного бита данных (маркер адреса). Бит CPOL в регистре USART_CR2 позволяет пользователю выбрать полярность тактов, а бит CPHA в регистре USART_CR2 позволяет выбрать фазу внешних тактов (см. рисунки 214, 215 и 216)

Во время состояния ожидания (Idle), преамбулы и отправки символа break, внешний тактовый сигнал CK не активируется.

В синхронном режиме, передатчик работает точно также, как и в асинхронном режиме. Но так как сигнал CK синхронизируется с сигналом TX (в зависимости от CPOL и CPHA), данные на TX синхронные.

Приемник в данном режиме работает не так, как в асинхронном режиме. Если RE=1, данные тактируются синхронно с сигналом CK (передний и задний фронт, в зависимости от значений CPOL и CPHA), без какой-либо передискретизации. Время установки и удержания сигнала должно соблюдаться (зависит от скорости обмена: 1/16 от времени бита).

Примечание – Вывод CK работает совместно с выводом TX. Таким образом, тактирование обеспечивается только, если разрешен передатчик (TE=1) и передаются данные (регистр данных USART_DR записан). Это означает, что невозможно принимать синхронные данные без передачи данных.

Должны быть выбраны биты LBCL, CPOL и CPHA, когда передатчик и приемник отключены (TE=RE=0) для обеспечения корректной работы тактовых импульсов. Значения данных бит не должны изменяться во время включения передатчика или приемника.

Рекомендуется устанавливать биты TE и RE одной командой с целью минимизации времени установки и удержания приемника.

Приемопередатчик USART поддерживает только режим ведущего устройства: он не может принимать или передавать данные, связанные со входом тактирования (CK всегда работает как выход).

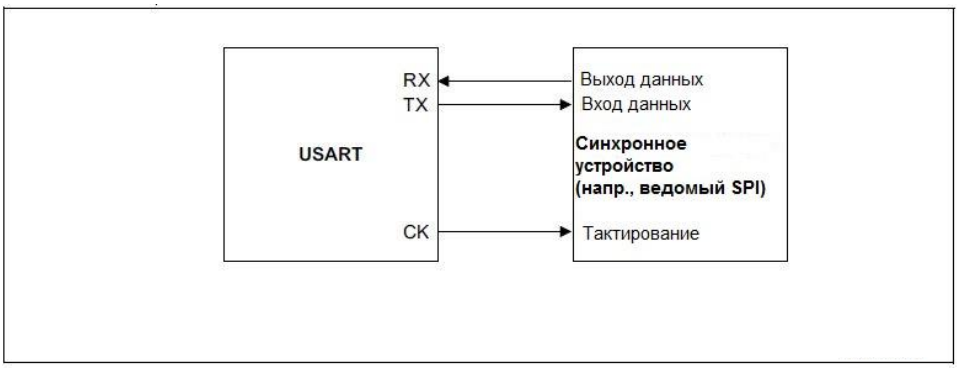



Рисунок 214 – Пример синхронной передачи USART

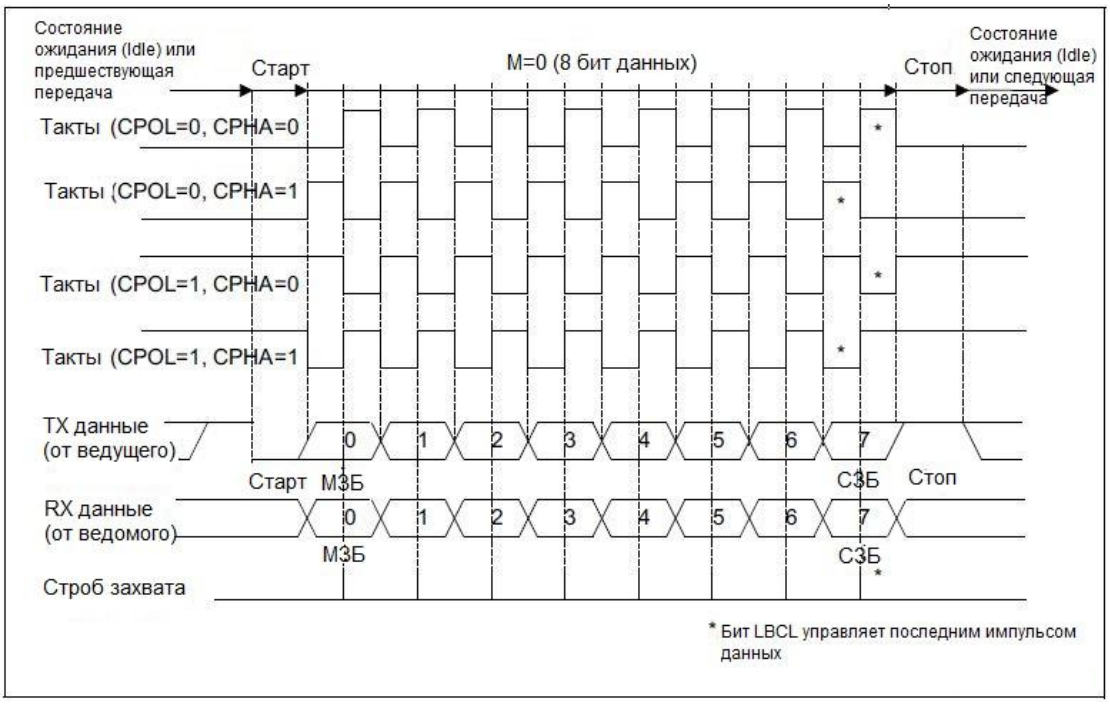



Рисунок 215 – Диаграмма тактирования данных USART (M=0)

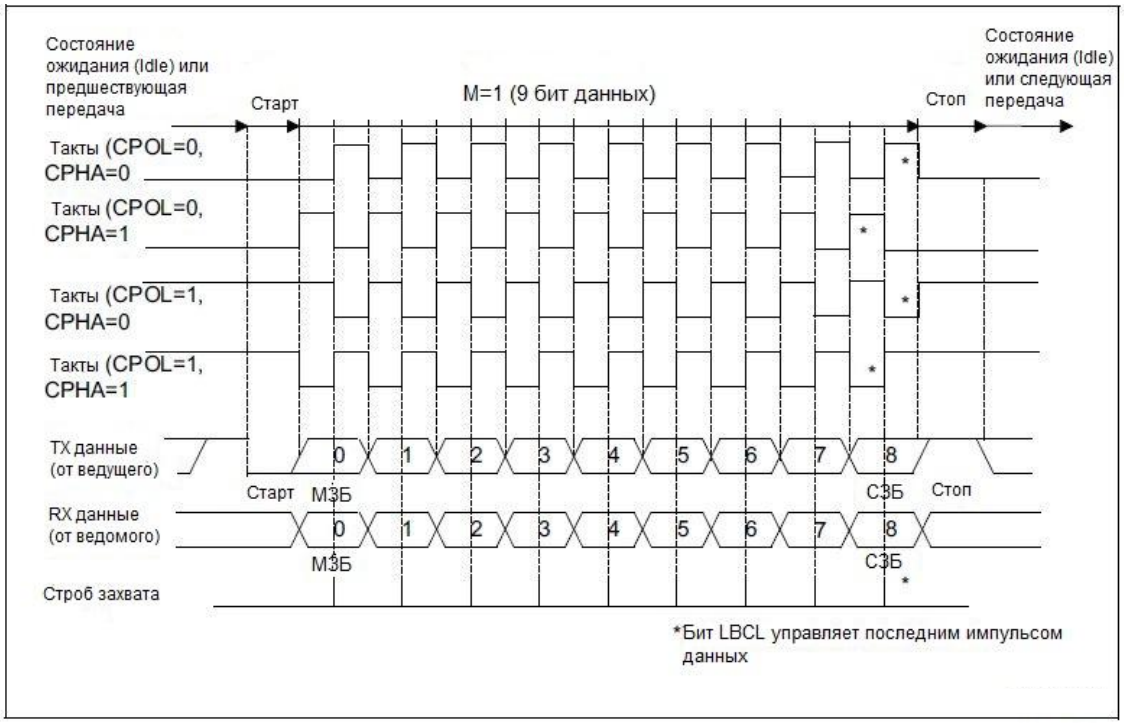



Рисунок 216 – Диаграмма тактирования данных USART (M=1)

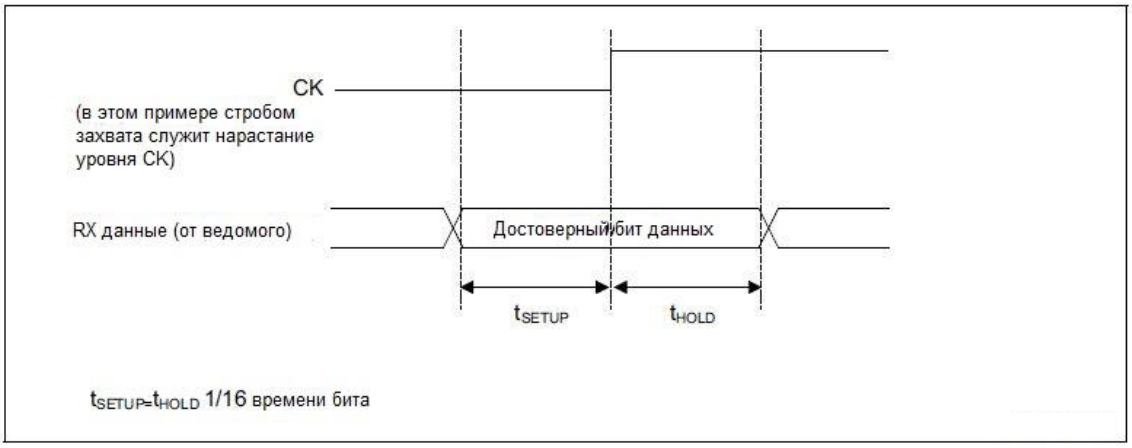



Рисунок 217 – Время установки/удержания RX

Примечание – Функция тактирования CK отличается в режиме Smartcard (См. далее описание режима Smartcard).

#### 18.12.2.9 Однопроводной полудуплексный обмен данными

Однопроводной полудуплексный режим обмена данными (single-wire half-duplex mode) выбирается путем установки бита HDSEL в регистре USART_CR3. В данном режиме должны быть сброшены следующие биты:

- SCEN в регистре USART_CR3.

USART может быть сконфигурирован для соответствия однопроводному полудуплексному протоколу, где линии сигналов TX и RX имеют внутреннее соединение. Выбор между полудуплексным или полнодуплексным режимом передачи данных осуществляется управляющим битом ‘HALF DUPLEX SEL’ (HDSEL в регистре USART_CR3).

Как только бит HDSEL устанавливается в «1»:

- линии сигналов TX и RX соединяются внутри;
- вывод RX больше не используется;
- вывод TX всегда свободен, если данные не передаются. Таким образом, он используется как стандартный ввод/вывод в режиме ожидания или при приеме данных. Это означает, что данный ввод/вывод должен быть сконфигурирован так, чтобы он был высокоомным входом (или выходом с открытым стоком), когда он не управляется USART.

Кроме этого, обмен данными происходит так же, как и обычном режиме передачи данных USART.

Конфликты на линии связи должны контролироваться программно (например, с использованием централизованного арбитра шины). В частности, передача данных никогда не блокируется аппаратной частью и будет продолжаться, как только данные будут записаны в регистр данных, пока установлен бит TE.

#### 18.12.2.10 Режим Smartcard

Режим смарт-карты (Smartcard mode) выбирается путем установки бита SCEN в регистре USART_CR3. В режиме Smartcard следующие биты должны быть сброшены:

- бит HDSEL в регистре USART_CR3.

Более того, бит CLKEN может быть установлен с целью тактирования смарт-карты.

Интерфейс режима Smartcard разработан для поддержки асинхронного протокола в соответствии со стандартом ISO 7816-3. USART должен быть сконфигурирован следующим образом:

- 8 бит + бит четности: где M=1 и PCE=1 в регистре USART_CR1
- 1,5 стоп бита при передаче и приеме: где STOP=11 в регистре USART_CR2.

Примечание – Также можно выбрать 0,5 стоп бит для приема, но рекомендуется использовать 1,5 стоп бита для передачи и приема данных, чтобы избежать переключения между двумя конфигурациями.

Рисунок 218 показывает пример, как выглядит сигнал данных с ошибкой четности и без ошибки четности.

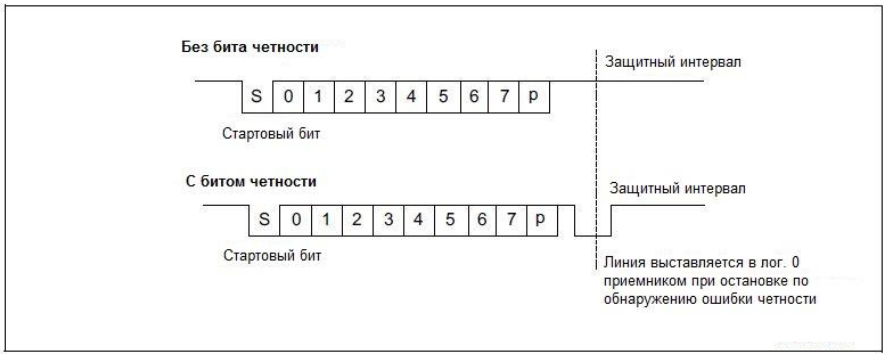



Рисунок 218 – Асинхронный протокол ISO 7816-3

При подключении к смарт-карте, выход TX приемопередатчика USART управляет двунаправленной линией, которая также управляется и смарт-картой. Вывод TX должен быть сконфигурирован как открытый сток.

Smartcard – это однопроводный полудуплексный протокол передачи данных.

- Передача данных из регистра сдвига передатчика осуществляется с гарантированной задержкой минимум ½ такта скорости. При нормальном режиме работы полная передача из регистра сдвига начнется сдвигом на следующем

перепаде тактов скорости. В режиме Smartcard эта передача дополнительно задерживается на ½ такта скорости.

- Если обнаружена ошибка четности при приеме фрейма, запрограммированного на 0,5 или 1,5 стоп-бита, линия передачи подтягивается в 0 на период такта скорости после завершения приема фрейма. Это показывает смарт-карте, что данные, переданные на приемопередатчик USART, не были корректно приняты. Данный сигнал NACK (подтягивание линии передачи в 0 на период 1 такта скорости) сгенерирует ошибку фрейма на стороне передатчика (конфигурация 1,5 стоп бита). Приложение должно обработать эту ситуацию повторной отправкой данных в соответствии с протоколом. Ошибка четности не подтверждается приемником (сигнал NACK), если установлен управляющий бит NACK, иначе сигнал NACK не передается.
- Установка флага TC может быть задержана путем программирования регистра защитного интервала времени (Guard Time). При нормальной работе флаг TC устанавливается, когда регистр сдвига передатчика пуст и нет никаких запросов на передачу. В режиме смарт-карты пустой регистр сдвига передатчика запускает счетчик защитного интервала для счета запрограммированного значения в регистре защитного интервала времени (Guard Time register). Флаг TC подтягивается в 0. Когда счетчик достигает запрограммированного значения, устанавливается флаг TC.
- На снятие флага TC режим смарт-карты не влияет.
- Если обнаружена ошибка фрейма на стороне передатчика (ответ NACK от приемника), сигнал NACK не будет обнаружен как стартовый бит блоком приема передатчика. В соответствии со стандартом ISO, длительность принятого сигнала NACK может составлять 1 или 2 периода тактов скорости.
- На стороне приемника, если была обнаружена ошибка четности, и был передан сигнал NACK, приемник не обработает сигнал NACK как стартовый бит.

Примечание – Символ break не имеет значения в режиме Smartcard. Данные 0x00 с ошибкой фрейма будут обработаны как данные, а не как символ break.

Фрейм Idle не передается при переключении бита TE. Фрейм Idle (как определено для других конфигураций) не определен в протоколе ISO.

Рисунок 219 показывает, как обрабатывается сигнал NACK. В данном примере USART передает данные и сконфигурирован на 1.5 стоп бита. Блок приемника USART разрешен для проверки целостности данных и сигнала NACK.

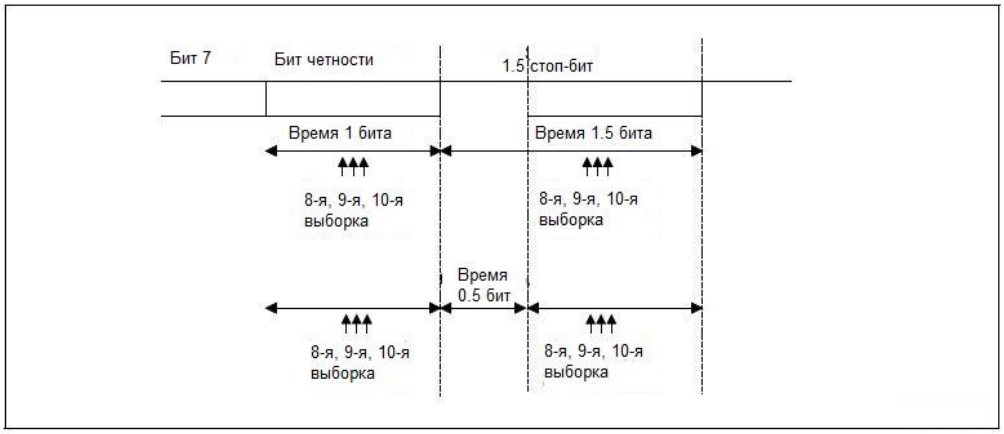



Рисунок 219 – Обнаружение ошибки четности с использованием1,5 стоп-бит

USART может тактировать смарт-карту при помощи выхода CK. В режиме смарт-карты, выход CK не связан с обменом данными, он просто генерирует тактовую частоту, полученную из внутреннего входа тактирования периферии при помощи 5-битного предделителя. Коэффициент деления конфигурируется в регистре предделителя USART_GTPR. Частота CK программируется от fCK/2 до fCK/62, где fCK это частота тактов входа периферии.

#### 18.12.2.11 Аппаратное управление потоком

Можно управлять потоком последовательных данных между двумя устройствами, используя вход CTS и выход RTS. Рисунок 220 показывает схему соединения двух устройств для данного режима:

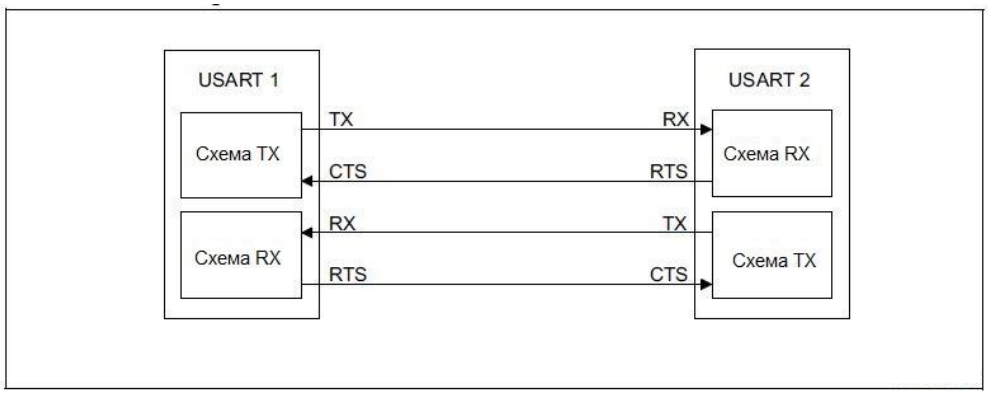



Рисунок 220 – Аппаратное управление потоком данных между двумя устройствами USART

Управление потоком RTS и CTS можно разрешить независимо записью битов RTSE и CTSE соответственно в 1 (в регистре USART_CR3).

***Управление потоком RTS***

Если разрешено управление потоком RTS (RTSE=1), выставляется сигнал RTS (лог. 0), когда приемник USART готов принимать новые данные. Если регистр приема заполнен, то сигнал RTS снимается, тем самым показывая, что ожидается остановка передачи в конце текущего фрейма.

Рисунок 221 показывает пример обмена с разрешенным управлением потоком RTS.

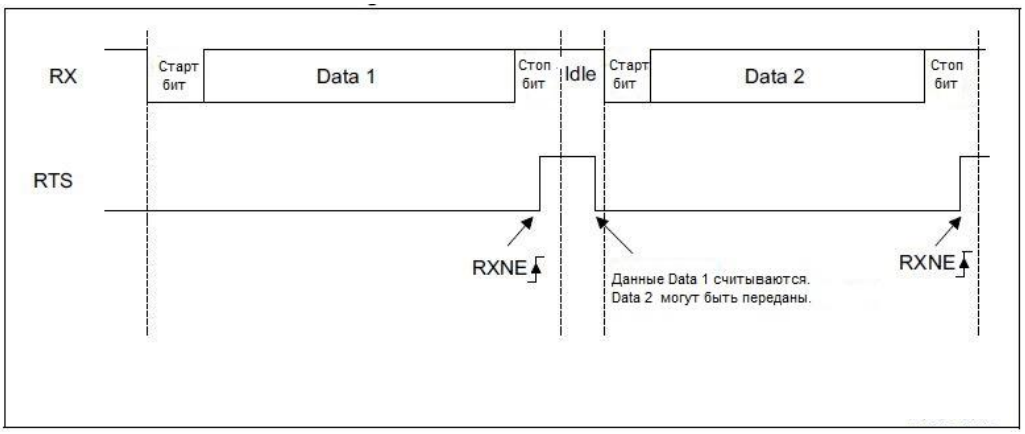



Рисунок 221 – Управление потоком RTS

***Управление потоком CTS***

Если управление потоком CTS разрешено (CTSE=1), передатчик проверяет вход CTS перед передачей следующего фрейма. Если устанавливается сигнал CTS (логический «0»), тогда начинается передача следующих данных (подразумевается, что есть данные для передачи, то есть другими словами TXE=0), в ином случае передача не произойдет. Когда снимается сигнал CTS во время передачи, текущая передача завершится перед остановкой передатчика.

Если CTSE=1, статус CTSIF будет установлен автоматически аппаратурой, как только вход CTS переключится. Это показывает, когда приемник будет готов к обмену данными. Прерывание генерируется, если установлен бит CTSIE в регистре USART_CR3. Рисунок 222 показывает пример обмена с разрешенным управлением потоком CTS.

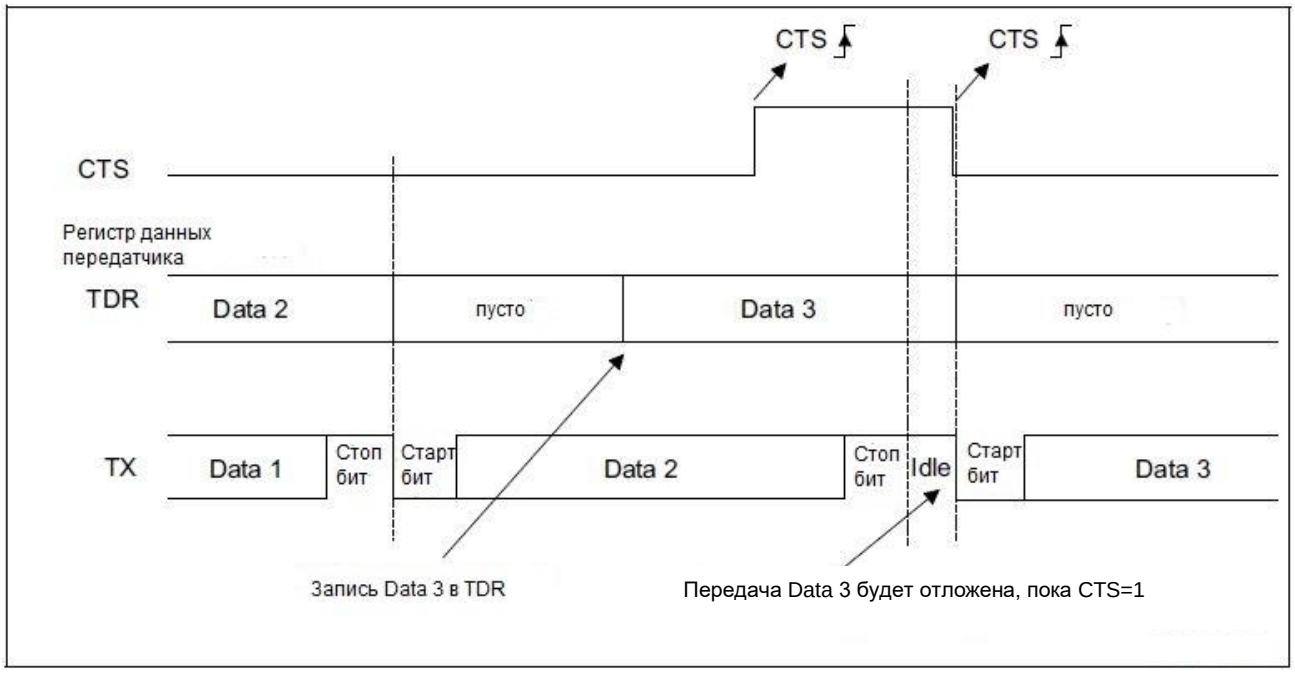



Рисунок 222 – Управление потоком CTS

Примечание – Особое поведение фреймов break: при разрешенном управлении потоком CTS, передатчик не проверяет статус входа CTS для отправки сигнала break.

### 18.12.3 *Прерывания USART*

Таблица 120 – Запрос на прерывание USART

| Событие | Флаг события | Бит разрешения |
|---|---|---|
| Регистр данных передачи пуст | TXE | TXEIE |
| Флаг CTS | CTS | CTSIE |
| Передача завершена | TC | TCIE |
| Принятые данные готовы для чтения | RXNE | RXNEIE |
| Обнаружена ошибка переполнения | ORE | |
| Обнаружена линия Idle | IDLE | IDLEIE |
| Ошибка четности | PE | PEIE |
| Флаг Break | LBD | LBDIE |
| Флаг шума, ошибки переполнения или ошибки фрейма в случае мультибуферного обмена | NF или ORE или FE | EIE |

События прерываний подсоединены к одному и тому же вектору прерывания (см. Рисунок 223).

- Во время передачи: Передача завершена (Transmission Complete), Линия пуста для начала передачи (Clear to Send) или Регистр данных передачи пуст (Transmit Data Register Empty).
- Во время приема: Обнаружение линии Idle, ошибка переполнения, Регистр данных приема не пуст (Receive Data register not empty), Ошибка четности, флаг шума (только при мультибуферном обмене) и ошибка фрейма (только при мультибуферном обмене).

Все эти события генерируют прерывания, если соответствующий бит разрешения прерывания установлен.

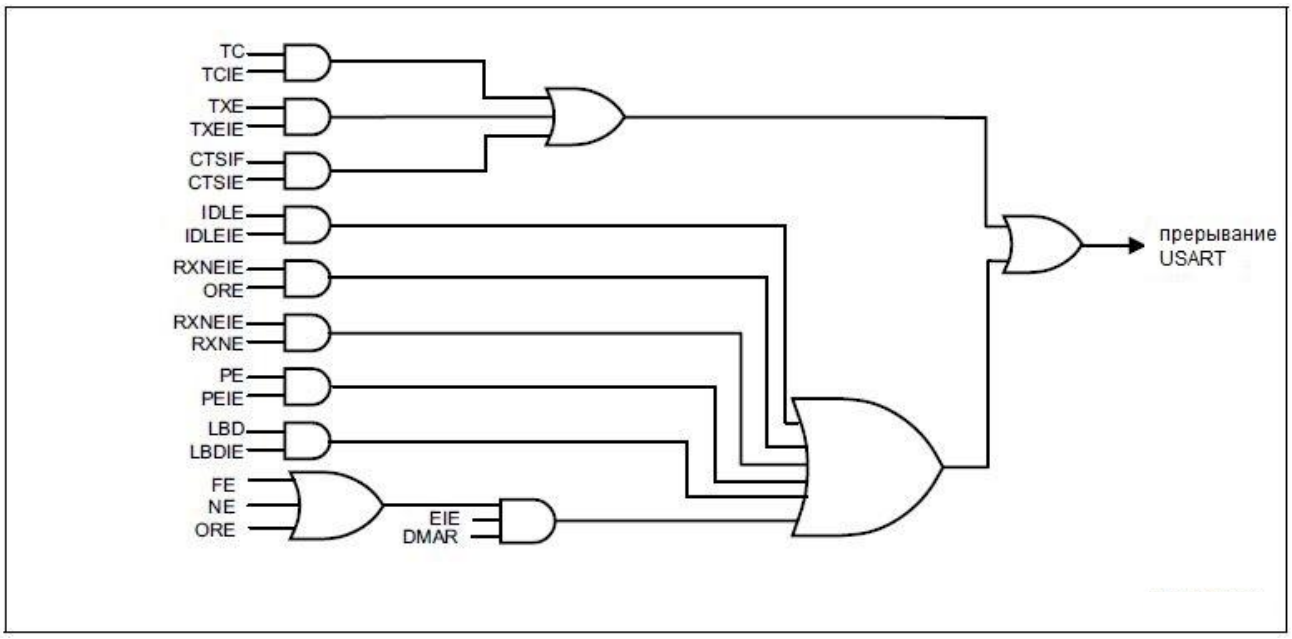



Рисунок 223 – Диаграмма отображений событий USART на прерывание

## 18.13 Контроллер OTP (OTP_CNTR)

Контроллер предназначен для управления чтением и записью в однократно программируемую память OTP, которая имеет четыре блока объёмом 128 Кбит (16 Кбайт). Общий объём OTP памяти 64 Кбайт.

Контроллер позволяет:

- читать данные из адресного пространства памяти по AHB, формируя необходимую задержку сигналом готовности;
- записывать (программировать) данные через регистровый интерфейс, доступный по AHB;
- осуществлять аппаратную блокировку отладочных и тестовых функций по данным из специального адреса OTP.

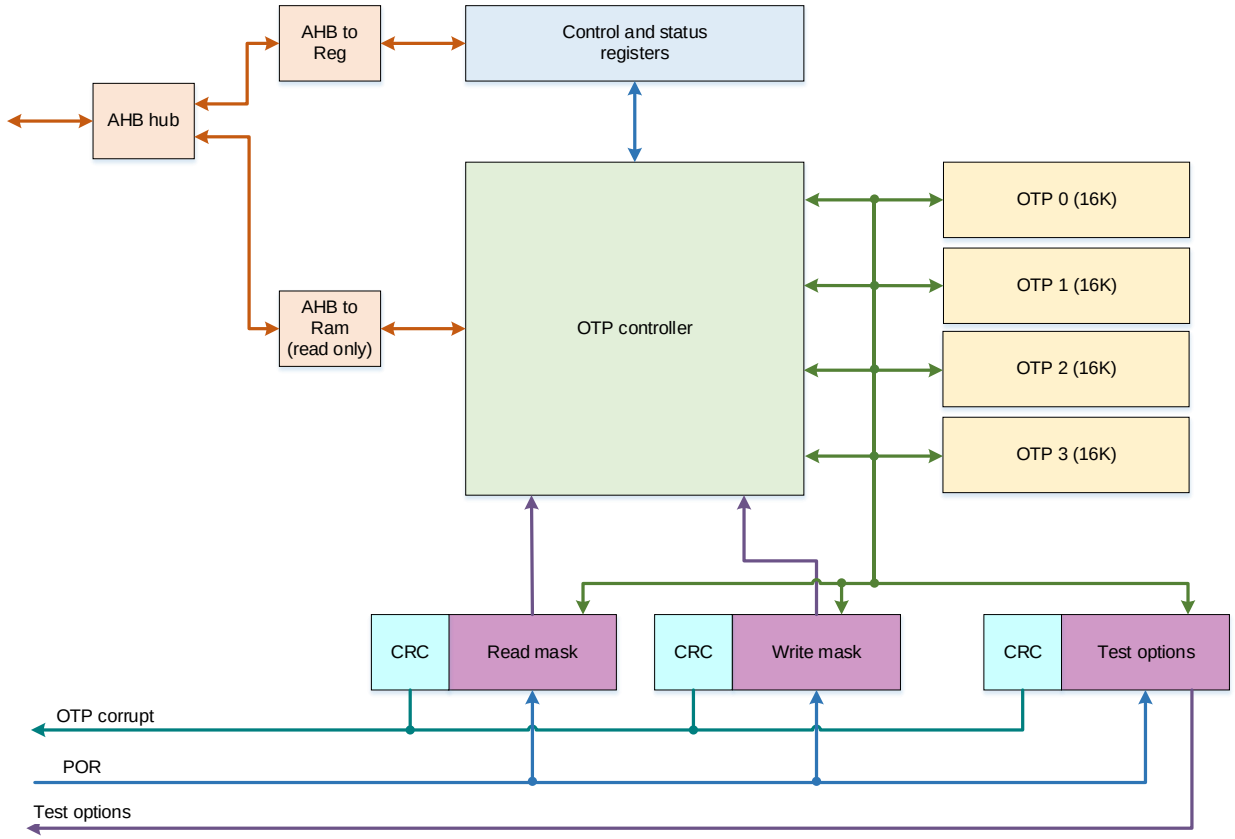



Рисунок 224 – Структурная схема

Для правильной работы операций записи/чтения через регистровый интерфейс необходимо рассчитать и записать значения задержек в регистрах DELAY_0_REG и DELAY_1_REG. Готовность принять новый запрос на чтение или запись можно осуществлять через проверку флага BUSY регистра STAT_CNTR для всех блоков OTP или через флаги OTP_3_BUSY - OTP_0_BUSY регистра OTP_STAT_REG для отдельных блоков.

При работе через регистровый интерфейс адрес записи и чтения задается в поле ADDR регистра RW_CMD_REG и устанавливается для всех блоков OTP. Запись в блок OTP-памяти осуществляется по одному биту для каждого блока. Выбор блоков памяти для программирования осуществляется с помощью полей WDATA_0 - WDATA_3 регистра RW_CMD_REG. При регистровом чтении регистр READ_DATA_REG содержит 32-битное слово, прочитанное из четырех блоков памяти с битового адреса ADDR. При чтении младшие 3 бита в поле адреса ADDR игнорируются и считаются равными 0, так как чтение осуществляется по одному байту из каждого блока памяти. В случае ошибки чтения в регистр READ_DATA_REG заносится значение 0xDEADBEEF.

Отображение блоков памяти OTP в адресное пространство защищенной подсистемы и соответствующая битовая адресация ADDR блоков памяти при регистровом доступе приведены в таблице 121.

Таблица 121 – Адресация памяти OTP при операциях записи и чтения

| Битовая адресация в блоке памяти OTP при записи/чтении | Байтовая адресация в пространстве защищенной подсистемы при чтении | | | |
|---|---|---|---|---|
| | Блок памяти OTP 3 (MSB) | Блок памяти OTP 2 | Блок памяти OTP 1 | Блок памяти OTP 0 (LSB) |
| ADDR = 7(msb)…0(lsb) | 0x1010_0003 | 0x1010_0002 | 0x1010_0001 | 0x1010_0000 |
| ADDR = 15…8 | 0x1010_0007 | 0x1010_0006 | 0x1010_0005 | 0x1010_0004 |

| | | | | |
|---|---|---|---|---|
| ADDR = 16391… 16384 | 0x1010_2003 | 0x1010_2002 | 0x1010_2001 | 0x1010_2000 |
| | | | | |
| ADDR = 131071 …131064 | 0x1010_FFFF | 0x1010_FFFE | 0x1010_FFFD | 0x1010_FFFC |

Модулем предусмотрена защита от записи/чтения в регистрах WRITE_PROTECT_REG и READ_PROTECT_REG для отдельных областей памяти в 2048 байт - защиту можно только установить, сброс осуществляется только общим сбросом по питанию. При попытке писать в защищенную от записи область или читать из защищенной от чтения области, а также при попытке чтения или записи по время установленного сигнала BUSY, выставляется флаг ошибки RW_ERROR в регистрах STAT_CNTR и RW_CMD_REG. Для защиты от модификации регистров WRITE_PROTECT_REG, READ_PROTECT_REG и TEST_OPT сторонними методами при штатной записи данных в регистры рассчитывается контрольная сумма для нового значения, по которой контролируется целостность данных при чтении. Состояние сигнала целостности данных регистров при чтении заносится в биты READ_PCRC, WRITE_PCRC, TEST_CRC регистра STAT_CNTR.

# 19 Системный таймер SysTick

Процессор имеет 24-разрядный системный таймер, SysTick, который считает вниз от загруженного в него значения до нуля; перезагрузка (возврат в начало) значения в регистр LOAD происходит по следующему фронту синхросигнала, затем счет продолжается по последующему фронту.

Когда процессор остановлен для отладки, таймер не декрементируется.

## 19.1 Описание регистров системного таймера SysTick

Таблица 122 – 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* | SysTick->CAL |
| * Калибровочное значение системного таймера | | | | | |

### 19.1.1 *SysTick->CTRL*

Регистр CTRL разрешает основные функции системного таймера.

Назначение бит:

Таблица 123 – Регистр контроля и статуса CTRL

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

#### 19.1.1.1 COUNTFLAG

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

#### 19.1.1.2 CLKSOURCE

Указывает источник синхросигнала:
0 – LSI;
1 – HCLK

#### 19.1.1.3 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.

#### 19.1.1.4 ENABLE

Разрешает работу таймера:
0 – работа таймера запрещена;
1 – работа таймера разрешена.

Когда ENABLE установлен в единицу, таймер загружает значение RELOAD из регистра LOAD и затем начинает декрементироваться. По достижению значения 0 таймер устанавливает бит COUNTFLAG и в зависимости от TCKINT генерирует запрос на прерывание. Затем загружается значение RELOAD и продолжается счет.

### 19.1.2 *SysTick->LOAD*

Регистр LOAD устанавливает стартовое значение, загружаемое в регистр VAL.

Таблица 124 – Регистр перегружаемого значения LOAD

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

#### 19.1.2.1 RELOAD

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

**Расчет значения RELOAD**

Значение RELOAD может быть любым в диапазоне 0x00000001-0x00FFFFFF. Значение 0 допустимо, но не оказывает эффекта, потому что запрос на прерывание и активизация бита COUNTFLAG происходит только при переходе таймера из состояния 1 в 0.

Расчет значения RELOAD происходит в соответствии с использованием таймера:

- Для формирования мультикороткого таймера с периодом N процессорных циклов применяется значение RELOAD, равное N-1. Например, если требуется прерывание каждые 100 циклов, то устанавливается значение RELOAD, равное 99;
- Для формирования одиночного прерывания после задержки в N тактов процессора используется значение N. Например, если требуется прерывание после 400 тактов процессора, то устанавливается RELOAD, равное 400.

### 19.1.3 *SysTick->VAL*

Регистр VAL содержит текущее значение системного таймера.

Таблица 125 – Регистр текущего значения таймера VAL

| Номер | 31…24 | 23…0 |
|---|---|---|
| Доступ | | |
| Сброс | | |
| | - | CURRENT |

#### 19.1.3.1 CURRENT

Чтение возвращает текущее значение системного таймера.

Запись любого значения очищает регистр в ноль, и также очищает бит COUNTFLAG регистра CTRL.

### 19.1.4 *SysTick->CAL*

Регистр CALIB показывает калибровочное значение системного таймера.

Таблица 126 – Регистр калибровочного значения таймера CAL

| Номер | 31 | 30 | 29…24 | 23…0 |
|---|---|---|---|---|
| Доступ | | | | |
| Сброс | | | | |
| | NOREF | SKEW | - | TENMS |

#### 19.1.4.1 NOREF

Читается как ноль.

#### 19.1.4.2 SKEW

Читается как ноль.

#### 19.1.4.3 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).

## 19.2 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.

# 20 Программная модель микроконтроллера

## 20.1 Таблица векторов прерываний

Таблица 127 – Распределение прерываний микроконтроллера

| Номер | Название | Описание |
|---|---|---|
| | | |
| -2 | - | - |
| -1 | - | - |
| 0 | - | - |
| 1 | - | - |
| … | - | - |
| 31 | - | - |
| 32 | FT_IF0 | Прерывание контроллера обработки отказов уровня 0 |
| 33 | FT_IF1 | Прерывание контроллера обработки отказов уровня 1 |
| 34 | FT_IF2 | Прерывание контроллера обработки отказов уровня 2 |
| 35 | FT_IF3 | Прерывание контроллера обработки отказов уровня 3 |
| 36 | CLK_IF | Прерывание блока контроля частоты |
| 37 | Зарезервировано | |
| 38 | RTC_IF | Прерывание блока таймера реального времени |
| 39 | BKP_IF | Прерывание блока батарейного домена |
| 40 | EXT_INTERROR0 | Внешние прерывания |
| 41 | EXT_INTERROR1 | |
| 42 | EXT_INTERROR2 | |
| 43 | EXT_INTERROR3 | |
| 44 | EXT_INTERROR4 | |
| 45 | EXT_INTERROR5 | |
| 46 | EXT_INTERROR6 | |
| 47 | EXT_INTERROR7 | |
| 48 | IF_ERR_SCR | Прерывания блока SCRUBBER |
| 49 | IF_FIN_SCR | |
| 50 | DMA_ERR | Прерывания блока завершения транзакций DMA |
| 51 | DMA_DONE0 | |
| 52 | DMA_DONE1 | |
| | … | |
| 82 | DMA_DONE31 | |
| 83 | IRQ_PORTA | Прерывание порта A |
| 84 | IRQ_PORTB | Прерывание порта B |
| 85 | IRQ_PORTC | Прерывание порта C |
| 86 | IRQ_PORTD | Прерывание порта D |
| 87 | - | - |
| 88 | - | - |
| 89 | INT_ETH0 | Прерывания контроллера Ethernet |
| 90 | INT_CAN0 | Прерывания контроллера CAN0 |
| 91 | INT_CAN1 | Прерывания контроллера CAN1 |
| 92 | INT_SSP0 | Прерывания контроллера SSP0 |
| 93 | RXINTR_SSP0 | |
| 94 | TXINTR_SSP0 | |
| 95 | RORINTR_SSP0 | |
| 96 | RTINTR_SSP0 | |
| 97 | SSPRNEINTR_SSP0 | |
| 98 | SSPTFEINTR_SSP0 | |
| 99 | SSPTNBSYINTR_SSP0 | |
| 100 | INT_SSP1 | Прерывания контроллера SSP1 |
| 101 | RXINTR_SSP1 | |
| 102 | TXINTR_SSP1 | |
| 103 | RORINTR_SSP1 | |
| 104 | RTINTR_SSP1 | |

| Номер | Название | Описание |
|---|---|---|
| 105 | SSPRNEINTR_SSP1 | |
| 106 | SSPTFEINTR_SSP1 | |
| 107 | SSPTNBSYINTR_SSP1 | |
| 108 | INT_UART0 | Прерывания контроллера UART0 |
| 109 | TNBSYINTR_UART0 | |
| 110 | TFEINTR_UART0 | |
| 111 | RNEINTR_UART0 | |
| 112 | EINTR_UART0 | |
| 113 | RTINTR_UART0 | |
| 114 | TXINTR_UART0 | |
| 115 | RXINTR_UART0 | |
| 116 | MSINTR_UART0 | |
| 117 | INT_UART1 | Прерывания контроллера UART1 |
| 118 | TNBSYINTR_UART1 | |
| 119 | TFEINTR_UART1 | |
| 120 | RNEINTR_UART1 | |
| 121 | EINTR_UART1 | |
| 122 | RTINTR_UART1 | |
| 123 | TXINTR_UART1 | |
| 124 | RXINTR_UART1 | |
| 125 | MSINTR_UART1 | |
| 126 | INT_UART2 | Прерывания контроллера UART2 |
| 127 | TNBSYINTR_UART2 | |
| 128 | TFEINTR_UART2 | |
| 129 | RNEINTR_UART2 | |
| 130 | EINTR_UART2 | |
| 131 | RTINTR_UART2 | |
| 132 | TXINTR_UART2 | |
| 133 | RXINTR_UART2 | |
| 134 | MSINTR_UART2 | |
| 135 | INT_UART3 | Прерывания контроллера UART3 |
| 136 | TNBSYINTR_UART3 | |
| 137 | TFEINTR_UART3 | |
| 138 | RNEINTR_UART3 | |
| 139 | EINTR_UART3 | |
| 140 | RTINTR_UART3 | |
| 141 | TXINTR_UART3 | |
| 142 | RXINTR_UART3 | |
| 143 | MSINTR_UART3 | |
| 144 | INT_USB | Прерывания контроллера USB |
| 145 | INT_MIL0 | Прерывания контроллера MIL0 |
| 146 | INT_DAC0 | Прерывания контроллеров DAC3, DAC2, DAC1, DAC0 |
| 147 | INT_DAC1 | |
| 148 | INT_DAC2 | |
| 149 | INT_DAC3 | |
| 150 | INT_FIFO_DAC0 | |
| 151 | INT_FIFO_DAC1 | |
| 152 | INT_FIFO_DAC2 | |
| 153 | INT_FIFO_DAC3 | |
| 154 | INT_TMR0 | Прерывания блоков таймеров TMR3, TMR2, TMR1, TMR0 |
| 155 | INT_TMR1 | |
| 156 | INT_TMR2 | |
| 157 | INT_TMR3 | |
| 158 | INT_QEP0 | Прерывание блоков квадратурных энкодеров |
| 159 | INT_QEP1 | |
| 160 | - | |
| 161 | - | |

| Номер | Название | Описание |
|---|---|---|
| 162 | INT_CAP0 | Прерывания блоков захвата |
| 163 | INT_CAP1 | |
| 164 | INT_CAP2 | |
| 165 | INT_CAP3 | |
| 166 | EPWM_INT_8 | Прерывания блоков ШИМ |
| 167 | EPWM_INT_7 | |
| 168 | EPWM_INT_6 | |
| 169 | EPWM_INT_5 | |
| 170 | EPWM_INT_4 | |
| 171 | EPWM_INT_3 | |
| 172 | EPWM_INT_2 | |
| 173 | EPWM_INT_1 | |
| 174 | EPWM_INT_0 | |
| 175 | EPWM_TZINT_8 | |
| 176 | EPWM_TZINT_7 | |
| 177 | EPWM_TZINT_6 | |
| 178 | EPWM_TZINT_5 | |
| 179 | EPWM_TZINT_4 | |
| 180 | EPWM_TZINT_3 | |
| 181 | EPWM_TZINT_2 | |
| 182 | EPWM_TZINT_1 | |
| 183 | EPWM_TZINT_0 | |
| 184 | EPWM_FIFOINT_8 | |
| 185 | EPWM_FIFOINT_7 | |
| 186 | EPWM_FIFOINT_6 | |
| 187 | EPWM_FIFOINT_5 | |
| 188 | EPWM_FIFOINT_4 | |
| 189 | EPWM_FIFOINT_3 | |
| 190 | EPWM_FIFOINT_2 | |
| 191 | EPWM_FIFOINT_1 | |
| 192 | EPWM_FIFOINT_0 | |
| 193 | CMP_INT0 | Прерывания блоков компараторов CMP3 - CMP0 |
| 194 | CMP_INT1 | |
| 195 | CMP_INT2 | |
| 196 | CMP_INT3 | |
| 197 | ADC_INT00 | Прерывания блоков АЦП |
| 198 | ADC_FIFOINT00 | |
| 199 | ADC_INT01 | |
| 200 | ADC_FIFOINT01 | |
| 201 | ADC_INT02 | |
| 202 | ADC_FIFOINT02 | |
| 203 | ADC_INT10 | |
| 204 | ADC_FIFOINT10 | |
| 205 | ADC_INT11 | |
| 206 | ADC_FIFOINT11 | |
| 207 | ADC_INT12 | |
| 208 | ADC_FIFOINT12 | |
| 209 | CRDC_INT | Прерывание блока CORDIC |
| 210 | CANFD_INT | Прерыывание блока CANFD |
| 211 | INT_MIL1 | Прерывание блока MIL1 |
| 212 | CRPT_FIFO_INT0 | Прерывание от шлюза канала 0 |
| 213 | CRPT_REG_INT0 | Прерывание от шлюза канала 0 |
| 214 | CRPT_FIFO_INT1 | Прерывание от шлюза канала 1 |
| 215 | CRPT_REG_INT1 | Прерывание от шлюза канала 1 |
| 216 | SDIO_INT | Прерывание блока SDIO |
| 217 | CRPT_LOAD_DONE_INT | Прерывание от криптографического сопроцессора (готов к работе) |

| Номер | Название | Описание |
|---|---|---|
| 218 | CRPT_LOCKED_UP_INT | Прерывание от криптографического сопроцессора (NMI или HardFault) |
| 239:219 | - | Резерв. |

## 20.2 Адресное пространство микроконтроллера

Таблица 128 – Адресное пространство микроконтроллера

| Базовый адрес | Размер, байт | Обозначение | Доступ по шинам | Описание |
|---|---|---|---|---|
| 0x0000_0000 | 8K | BOOT ROM | I,D,C | Область памяти загрузочного ПЗУ, из этой памяти микроконтроллер начинает работу, определяет дальнейший режим функционирования и выполняет необходимые настройки |
| 0x0100_0000 | 1M | FLASH | I,D,C | Память программ. Основная память программ для пользовательской программы |
| 0x0200_0000 | 128K | RAMC | I,D,C | Статическое ОЗУ в CODESECTION, более быстрая память для выполнения кода из ОЗУ. Физически разделена на 2 банка |
| 0x1000_0000 | 256M | EXT BUS | I,D,S,M,C | Область памяти для отображения внешней системной шины |
| 0x2000_0000 | 128K | RAMD | S,M,C | Статическое ОЗУ в DATASECTION, предназначено для данных. Физически разделена на 2 банка |
| 0x2100_0000 | | E0 | S,M | Регистры блока контроллера ETHERNETMAC |
| 0x2101_0000 | | USB | S,M | Регистры блока контроллера USB |
| 0x2200_0000 | 32M | BIT BAND REGION RAMD | * | Функционально выделенная область памяти на уровне процессорного ядра, позволяющая упростить битовые манипуляции в RAMD |
| 0x4000_0000 | 1M | PERIPHERAL | S,M | Область отображения регистров периферии шины APB. Подробнее см. таблицу «Распределение областей памяти для отображения регистров периферии» |

| Базовый адрес | Размер, байт | Обозначение | Доступ по шинам | Описание |
|---|---|---|---|---|
| 0x4200_0000 | 32M | BITBANDREGIONPERIPHERAL | * | Функционально выделенная область памяти на уровне процессорного ядра, позволяющая упростить битовые манипуляции в PERIPHERAL |
| 0x5000_0000 | 256M | EXT BUS | S,M,C | Область памяти для отображения внешней системной шины |
| 0x6000_0000 | 1G | EXT BUS | S,M,C | Область памяти для отображения внешней системной шины |
| 0x9000_0000 | 1G | EXTBUS | S,M,C | Область памяти для отображения внешней системной шины |
| 0xE000_0000 | 512M | SYSTEM | - | Отображение внутренних регистров процессорного ядра. Подробнее см. таблицу «Распределение областей памяти для отображения регистров периферии ядра» |

## 20.3 Адресное пространство периферийных блоков микроконтроллера

Таблица 129 – Адресное пространство сектора периферийных блоков микроконтроллера

| | Базовый адрес | Размер, байт | Обозначение | Доступ по шинам | Описание |
|---|---|---|---|---|---|
| | | | | | |
| 0 | 0x4000_0000 | - | CLK_CNTR | S,M | Контроллер тактовых частот |
| 1 | 0x4000_1000 | | BKP_CNTR | | Контроллер батарейного домена и часов реального времени |
| 2 | - | | - | | - |
| 3 | 0x4000_3000 | | FTMODE_CNTR | | Контроллер блока диагностики ошибок |
| 4 | 0x4000_4000 | | WDG_CNTR | | Контроллер сторожевых таймеров |
| 5 | 0x4000_5000 | | ROM_CNTR | | Контроллер ПЗУ |
| 6 | 0x4000_6000 | | FLASH_CNTR | | Контроллер FLASH 0 памяти |
| 7 | | | | | |
| 8 | 0x4000_8000 | | RAMC0_CNTR | | Контроллер ОЗУ0 в области CODE |
| 9 | 0x4000_9000 | | RAMC1_CNTR | | Контроллер ОЗУ1 в области CODE |
| 10 | 0x4000_A000 | | RAMD0_CNTR | | Контроллер ОЗУ0 в области DATA |
| 11 | 0x4000_B000 | | RAMD1_CNTR | | Контроллер ОЗУ1 в области DATA |
| 12 | 0x4000_C000 | | EBC_CNTR | | Контроллер внешней шины |
| | | | | | |
| 13 | 0x4008_0000 | | PORTA_CNTR | | Контроллер порта A |
| 14 | 0x4008_1000 | | PORTB_CNTR | | Контроллер порта B |
| 15 | 0x4008_2000 | | PORTC_CNTR | | Контроллер порта C |
| 16 | 0x4008_3000 | | PORTD_CNTR | | Контроллер порта D |
| 17 | 0x4008_4000 | | - | | Зарезервировано |
| 18 | 0x4008_5000 | | - | | Зарезервировано |
| 19 | 0x4008_6000 | | - | | Зарезервировано |
| 20 | 0x4008_7000 | | MILCNR1_CNTR | | Контроллер MIL1 управление |

| | Базовый адрес | Размер, байт | Обозначение | Доступ по шинам | Описание |
|---|---|---|---|---|---|
| 21 | 0x4008_9000 | | SSP0_CNTR | | Контроллер SSP0 |
| 22 | 0x4008_A000 | | SSP1_CNTR | | Контроллер SSP1 |
| 23 | 0x4008_B000 | | CAN0_CNTR | | Контроллер CAN0 |
| 24 | 0x4008_C000 | | CAN1_CNTR | | Контроллер CAN1 |
| 25 | 0x4008_D000 | | UART0_CNTR | | Контроллер UART0 |
| 26 | 0x4008_E000 | | UART1_CNTR | | Контроллер UART1 |
| 27 | 0x4008_F000 | | UART2_CNTR | | Контроллер UART2 |
| 28 | 0x4009_0000 | | UART3_CNTR | | Контроллер UART3 |
| 29 | 0x4009_1000 | | MILCNR0_CNTR | | Контроллер MIL0 управление |
| 30 | 0x4009_3000 | | USB_CNTR | | Контроллер USB (регистры управления специальными функциями USB и настройка RAM USB) |
| 31 | 0x4009_4000 | | TIMER0_CNTR | | Контроллер TIMER0 |
| 32 | 0x4009_5000 | | TIMER1_CNTR | | Контроллер TIMER1 |
| 33 | 0x4009_6000 | | TIMER2_CNTR | | Контроллер TIMER2 |
| 34 | 0x4009_7000 | | TIMER3_CNTR | | Контроллер TIMER3 |
| 35 | 0x4009_8000 | | CAP0_CNTR | | Контроллер CAP0 |
| 36 | 0x4009_9000 | | CAP1_CNTR | | Контроллер CAP1 |
| 37 | 0x4009_A000 | | CAP2_CNTR | | Контроллер CAP2 |
| 38 | 0x4009_B000 | | CAP3_CNTR | | Контроллер CAP3 |
| 39 | 0x4009_C000 | | QEP0_CNTR | | Контроллер QEP0 |
| 40 | 0x4009_D000 | | QEP1_CNTR | | Контроллер QEP1 |
| 41 | 0x4009_E000 | | PWM0_CNTR | | Контроллер PWM0 |
| 42 | 0x4009_F000 | | PWM1_CNTR | | Контроллер PWM1 |
| 43 | 0x400A_0000 | | PWM2_CNTR | | Контроллер PWM2 |
| 44 | 0x400A_1000 | | PWM3_CNTR | | Контроллер PWM3 |
| 45 | 0x400A_2000 | | PWM4_CNTR | | Контроллер PWM4 |
| 46 | 0x400A_3000 | | PWM5_CNTR | | Контроллер PWM5 |
| 47 | 0x400A_4000 | | PWM6_CNTR | | Контроллер PWM6 |
| 48 | 0x400A_5000 | | PWM7_CNTR | | Контроллер PWM7 |
| 49 | 0x400A_6000 | | PWM8_CNTR | | Контроллер PWM8 |
| 50 | 0x400A_7000 | | ADC0_CNTR | | Контроллер ADC0 |
| 51 | 0x400A_8000 | | ADC1_CNTR | | Контроллер ADC1 |
| 52 | 0x400A_9000 | | ADC2_CNTR | | Контроллер ADC2 |
| 53 | 0x400A_A000 | | DAC0_CNTR | | Контроллер DAC0 |
| 54 | 0x400A_B000 | | DAC1_CNTR | | Контроллер DAC1 |
| 55 | 0x400A_C000 | | DAC2_CNTR | | Контроллер DAC2 |
| 56 | 0x400A_D000 | | DAC3_CNTR | | Контроллер DAC3 |
| 57 | 0x400A_E000 | | COMP0_CNTR | | Контроллер COMP0 |
| 58 | 0x400A_F000 | | COMP1_CNTR | | Контроллер COMP1 |
| 59 | 0x400B_0000 | | COMP2_CNTR | | Контроллер COMP2 |
| 60 | 0x400B_1000 | | COMP3_CNTR | | Контроллер COMP3 |
| 61 | 0x400B_2000 | | I2C_CNTR | | Контроллер I2C_CNTR |
| 62 | 0x400B_3000 | | CORDIC_CNTR | | Контроллер CORDIC_CNTR |
| 63 | 0x400B_4000 | | - | | |
| 64 | 0x400B_5000 | | SDIO | | Контроллер SDIO |
| 65 | 0x400B_6000 | | CRC | | Контроллер CRC |

## 20.4 Адресное пространство сектора регистров ядра

В режиме DUALCORE дублируется для каждого из ядер. Подробное описание процессорных ядер Cortex-M4F приведено в документе Cortex-M4 Devices Generic User Guide.

Таблица 130 – Адресное пространство сектора регистров ядра

| Базовый адрес | Размер, байт | Обозначение | Описание |
|---|---|---|---|
| 0xE000_E008 | - | System Control Block | Системные регистры |
| 0xE000_E010 | | System Timer | Регистры системного таймера |
| 0xE000_E100 | | NVIC | Регистры контроллера прерываний |
| 0xE000_ED00 | | System Control Block | Системные регистры |
| 0xE000_ED90 | | MPU | Регистры блока защиты памяти |
| 0xE000_EF00 | | NVIC | Регистры контроллера прерываний |
| 0xE000_EF30 | | FPU | Регистры блока вычислений с плавающей запятой |
| 0xE004_0000 | | TPIU | Блок управления портом трассировки TPIU |
| 0xE004_1000 | | ETM | Блок формирования трассировки ядра ETM |
| 0xE004_2000 | | DMA_CNTR | Блок прямого доступа в память |
| 0xE004_3000 | | SCR_CNTR | Блок диагностики памяти |
| 0xE004_4000 | | ICACHE_CNTR | Блок управления I-кэш памятью |
| 0xE004_5000 | | DCACHE_CNTR | Блок управления D-кэш памятью |
| 0xE00F_E000 | | SMP_REG | Регистр идентификации ядра<br>0 – первое ядро<br>1 – второе ядро<br>Запрещено чтение данного регистра в режиме LOCKSTEP |
| 0xE00F_E000 | | ROM TABLE | Таблица описания периферии ядра |

## 20.5 Регистры ядра (SCB) и контроллер прерываний (ISR)

Таблица 131 – Регистры ядра микроконтроллера

| Базовый адрес | Смещение | Название | Состояние после сброса | Описание |
|---|---|---|---|---|
| 0xE000_0000 | | ITM | | |
| | | | | |
| 0xE000_1000 | | DWT | | |
| | 0x000 | | | |
| | | | | |
| 0xE000_E000 | | SC | | |
| | 0x004 | ICTR | | Регистр описания контроллера прерываний |
| | 0x008 | ACTLR | | Регистр настройки ядра |
| | 0x010 | STCSR | | Регистр настройки и состояния SysTick |
| | 0x014 | STRVR | | Регистр основания счета SysTick |
| | 0x018 | STCVR | | Регистр текущего значения SysTick |
| | 0x01C | STCR | | Регистр калибровки SysTick |
| | | | | |
| | 0x100-0x11C | | | |
| | | | | |
| | 0x180-0x19C | | | |
| | | | | |
| | 0x200-0x21C | | | |
| | | | | |
| | 0x300-0x31C | | | |
| | | | | |
| | 0x400-0x41C | | | |

| Базовый адрес | Смещение | Название | Состояние после сброса | Описание |
|---|---|---|---|---|
| | | | | |
| | 0xD00 | CPUID | | Регистр ID процессорного ядра |
| | 0xD04 | ICSR | | Регистр настройки и состояния прерываний |
| | 0xD08 | VTOR | | Регистр смещения таблицы векторов прерывания |
| | 0xD0C | AIRCR | | Регистр прерываний приложений и сброса |
| | 0xD10 | SCR | | Регистр настройки системы |
| | 0xD14 | CCR | | Регистр настройки и конфигурации |
| | 0xD18 | SHPR1 | | Регистр настройки приоритета 1 обработчиков |
| | 0xD1C | SHPR2 | | Регистр настройки приоритета 2 обработчиков |
| | 0xD20 | SHPR3 | | Регистр настройки приоритета 3 обработчиков |
| | 0xD24 | SHCSR | | Регистр настройки и состояния обработчиков |
| | 0xD28 | CFSR | | Регистр конфигурации состояния ошибки |
| | 0xD2C | HFSR | | Регистр состояния HardFault |
| | 0xD30 | DFSR | | Регистр состояния ошибки отладки |
| | 0xD34 | MMFAR | | Регистр адреса ошибки управления памяти |
| | 0xD38 | BFAR | | Регистр адреса ошибки BusFault |
| | 0xD3C | AFSR | | Регистр состояния ошибки настройки ядра |
| | 0xD40 | ID_PFR0 | | |
| | 0xD44 | ID_PFR1 | | |
| | 0xD48 | ID_DFR0 | | |
| | 0xD4C | ID_AFR0 | | |
| | 0xD50 | ID_MMFR0 | | |
| | 0xD54 | ID_MMFR1 | | |
| | 0xD58 | ID_MMFR2 | | |
| | 0xD5C | ID_MMFR3 | | |
| | 0xD60 | ID_IASR0 | | |
| | 0xD64 | ID_IASR1 | | |
| | 0xD68 | ID_IASR2 | | |
| | 0xD6C | ID_IASR3 | | |
| | 0xD70 | ID_IASR4 | | |
| | | | | |
| | 0xD88 | CPACR | | |
| | | | | |
| | 0xD90 | MPU_TYPE | | |
| | 0xD94 | MPU_CNTR | | |
| | 0xD98 | MPU_RNR | | |
| | 0xD9C | MPU_RBAR | | |
| | 0xDA0 | MPU_RASR | | |
| | 0xDA4 | MPU_RBAR_A1 | | |
| | 0xDA8 | MPU_RASR_A1 | | |
| | 0xDAC | MPU_RBAR_A2 | | |
| | 0xDB0 | MPU_RASR_A2 | | |
| | 0xDB4 | MPU_RBAR_A3 | | |
| | 0xDB8 | MPU_RASR_A3 | | |
| | | | | |
| | 0xDF0 | DHCSR | | |
| | 0xDF4 | DCRSR | | |

| Базовый адрес | Смещение | Название | Состояние после сброса | Описание |
|---|---|---|---|---|
| | 0xDF8 | DCRDR | | |
| | 0xDFC | DEMCR | | |
| | | | | |
| | 0xF00 | STIR | | |
| | | | | |
| | 0xF34 | FPCCR | | |
| | 0xF38 | FPCAR | | |
| | 0xF3C | FPDSCR | | |
| | 0xF40 | MVFR0 | | |
| | 0xF44 | MVFR1 | | |
| | | | | |
| | 0xFD0 | Peripheral ID4 | | |
| | | | | |
| | 0xFE0 | Peripheral ID0 | | |
| | 0xFE4 | Peripheral ID1 | | |
| | 0xFE8 | Peripheral ID2 | | |
| | 0xFEC | Peripheral ID3 | | |
| | 0xFF0 | Component ID0 | | |
| | 0xFF4 | Component ID1 | | |
| | 0xFF8 | Component ID2 | | |
| | 0xFFC | Component ID3 | | |
| | | | | |
| 0xE004_0000 | | TPUI | | |
| | | | | |
| 0xE00F_F000 | | | | |
| | 0x000 | SCS | | |
| | 0x004 | DWT | | |
| | 0x008 | FPB | | |
| | 0x00C | ITM | | |
| | 0x010 | TPUI | | |
| | 0x014 | ETM | | |
| | 0x018 | End Marker | | |
| | 0xFFC | SYSTEM ACCESS | | |

## 20.6 Описание регистров контроллера кэш-памяти (CACHE_CNTR)

Таблица 132 – Регистры контроллера кэш-памяти микроконтроллера

| Базовый адрес | Смещение | Название | Состояние после сброса | Описание |
|---|---|---|---|---|
| 0xE0044000 | | | | Базовый адрес контроллера кэша шины I-Bus |
| 0xE0045000 | | | | Базовый адрес контроллера кэша шины D-Bus |
| | 0x0000_0000 | KEY | | Регистр ключа, разрешающего модификацию остальных регистров |
| | 0x0000_0004 | CNTR | | Регистр управления контроллером кэш-памяти |
| | 0x0000_0008 | HIT_CNT | | Регистр счетчика числа попаданий Кеш |
| | 0x0000_000C | MISS_CNT | | Регистр счетчика числа промахов Кэш |
| | 0x0000_0020 | ECCCS | | Регистр статуса ошибок ECCCACHE |
| | 0x0000_0024 | ECCADR | | Регистр адреса последней ошибки |
| | 0x0000_0028 | ECCDATA | | Регистр данных последней ошибки |

| Базовый адрес | Смещение | Название | Состояние после сброса | Описание |
|---|---|---|---|---|
| | 0x0000_002C | ECCECC | | Регистр контрольной суммы последней ошибки |

### 20.6.1 *KEY*

| Base ADDR= | 0xE0044000<br>0xE0045000 | Offset= | 0x0000_0000 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| REG Name: | KEY | | | | | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | KEY[31:0] | 0000_0000 | При записи в регистр значения 0x8555AAA1 открывается возможность записи в другие регистры блока CACHE_CNTR |

### 20.6.2 *CNTR*

| Base ADDR= | 0xE0044000<br>0xE0045000 | Offset= | 0x0000_0004 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | CLR_MIS | CLR_HIT | EN_CACHE_BUS | EN_CACHE_FLASH | CLR_CACHE | EN_CACHE |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…6 | | | Зарезервировано |
| 5 | CLR_MIS | | Бит сброса регистра счетчика промахов<br>Сброс выполняется записью 1 |
| 4 | CLR_HIT | | Бит сброса регистра счетчика попаданий<br>Сброс выполняется записью 1 |
| 3 | EN_CACHE_BUS | | Бит разрешения кэширования данных от EXT_BUS<br>Имеет значение только при EN_CACHE=1<br>0 –данные не кэшируются<br>1 – данные кэшируются<br>Кэширование осуществляется пословно |
| 2 | EN_CACHE_ FLASH | | Бит разрешения кэширования данных от FLASH<br>Имеет значение только при EN_CACHE=1<br>0 –данные не кэшируются<br>1 – данные кэшируются<br>Кэширование осуществляется строками целиком |

| | | | |
|---|---|---|---|
| 1 | CLR_ CACHE | | Бит сброса кэша.<br>Сброс выполняется записью 1 |
| 0 | EN_CACHE | | Бит общего разрешения работы кэша<br>0 – выключен<br>1 – включен |

### 20.6.3 *HIT_CNT*

| Base ADDR= | 0xE0044000<br>0xE0045000 | Offset= | 0x0000_0008 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| CNT[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CNT[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | CNT[31:0] | | Счетчик числа попаданий в кэш |

### 20.6.4 *MISS_CNT*

| Base ADDR= | 0xE0044000<br>0xE0045000 | Offset= | 0x0000_000C |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CNT[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CNT[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | CNT[31:0] | | Счетчик числа промахов в кэш |

### 20.6.5 *ECCCS*

| Base ADDR= | 0xE0044000<br>0xE0045000 | Offset= | 0x0000_0020 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SECC_CNT[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DECC_CNT[7:0] | | | | | | | | CLR_DCNT | CLR_SCNT | FIX_DECC | FIX_SECC | DECC_IE | SECC_IE | DECC | SECC |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…16 | SECC_CNT[15:0] | | Счетчик числа одиночных ошибок<br>0 – нет ошибок<br>1 – одна ошибка<br>…<br>65535 – 65535 или более ошибок |
| 15:8 | DECC_CNT[7:0] | | Счетчик числа двойных ошибок<br>0 – нет ошибок<br>1 – одна ошибка<br>…<br>255 – 255 или более ошибок |
| 7 | CLR_DCNT | | Бит сброса счетчика двойных ошибок<br>Запись 1 сбрасывает счетчик DECC_CNT<br>Всегда читается как 0 |
| 6 | CLR_SCNT | | Бит сброса счетчика двойных ошибок<br>Запись 1 сбрасывает счетчик DECC_CNT<br>Всегда читается как 0 |
| 5 | FIX_DECC | | Бит разрешения фиксации в регистрах адреса и данных адреса последней ошибки при двойной ошибке<br>0 – запрещено<br>1 – разрешено |
| 4 | FIX_SECC | | Бит разрешения фиксации в регистрах адреса и данных адреса последней ошибки при одиночной ошибке<br>0 – запрещено<br>1 – разрешено |
| 3 | DECC_IE | | Бит разрешения прерывания при возникновении двойной ошибки ECC<br>0 – прерывание запрещено<br>1 – прерывание разрешено |
| 2 | SECC_IE | | Бит разрешения прерывания при возникновении одиночной ошибки ECC<br>0 – прерывание запрещено<br>1 – прерывание разрешено |
| 1 | DECC | | Флаг возникновения двойной ошибки<br>0 – ошибок не было<br>1 – ошибка была<br>Сбрасывается записью 1 |
| 0 | SECC | | Флаг возникновения одиночной ошибки<br>0 – ошибок не было<br>1 – ошибка была<br>Сбрасывается записью 1 |

### 20.6.6 *ECCADR*

| Base ADDR= | | 0xE0044000<br>0xE0045000 | | Offset= | | 0x0000_0024 | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |

| ECCADR[31:16] | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECCADR[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | ECCADR[31:0] | | Адрес последней двойной или одиночной ошибки |

### 20.6.7 *ECCDATA*

| Base ADDR= | | 0xE0044000<br>0xE0045000 | | | Offset= | | 0x0000_0028 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| ECCDATA[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECCDATA[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | ECCDATA[31:0] | | Считанные данные при последней двойной или одиночной ошибке, без корректировки ECC |

### 20.6.8 *ECCECC*

| Base ADDR= | 0xE0044000<br>0xE0045000 | Offset= | 0x0000_002C |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| . | | | | | | | | ECC[7:0] | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…8 | - | | Зарезервировано |
| 7…0 | ECC[7:0] | | Считанные ECC-биты при последней двойной или одиночной ошибке, без корректировки ECC |

## 20.7 Описание регистров контроллера диагностики памяти SCR_CNTR

Таблица 133 – Регистры контроллера диагностики памяти

| **Базовый Адрес** | | **Название** | **Описание** |
|---|---|---|---|
| 0xE0043000 | | SCRCNTR | |
| **Смещение** | | | |
| 0x0000 | 0 | SCR_CNTRL | Регистр задания режима работы блока SCRUBBER |
| 0x0004 | 1 | SCR_SADDR | Регистр начального адреса блока SCRUBBER |
| 0x0008 | 2 | SCR_FADDR | Регистр конечного адреса блока SCRUBBER |
| 0x000c | 3 | SCR_DATA | Регистр данных блока SCRUBBER |
| 0x0010 | 4 | SCR_ECC | Регистр ECC блока SCRUBBER |

### 20.7.1 *SCR_CNTRL*

| Base ADDR= | **0xE004_3000** | Offset= | **0x0000_0000** | Reset= | 0x0004_0000 |
|---|---|---|---|---|---|
| REG Name: | SCR_CNTRL | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | | | | | | | | | | SCR_DREADY | SCR_ERROR_IF | SCR_FINISH_IF |
| | | | | | | | | | | | | | r | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SCR_TIME [7:0] | | | | | | | | SCR_CYCLE | SCR_SEC | SCR_ERROR_IE | SCR_FINISH_IE | SCR_MODE[2:0] | | | SCR_EN |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…8 | Зарезервировано | | Резерв |
| 18 | SCR_DREADY | 1 | Флаг завершения операции чтения или записи 32 разрядного слова при SCR_MODE=3 или SCR_MODE=4. |
| 17 | SCR_ERROR_IF | 0 | Бит события обнаружения ошибки:<br>0 – ошибки не обнаружено<br>1 – обнаружена ошибка<br>Сбрасывается записью 1. При совпадении момента программного сброса и взведения бита при обнаружении новой ошибки, более приоритетным является повторное взведение бита |
| 16 | SCR_FINISH_IF | 0 | Бит события окончания выполнения операции:<br>0 – операция не закончена<br>1 – операция закончена<br>Сбрасывается записью 1. При совпадении момента программного сброса и взведения бита при повторном выполнении операции в циклическом режиме более приоритетным является повторное взведение бита |
| 15…8 | SCR_TIME [7:0] | 0 | Биты задания периода выполнения чтения и чтения-записи в рамках выполняемой операции:<br>0 – без дополнительных тактов паузы (непрерывно)<br>1 – один дополнительный такт HCLK<br>2 – два дополнительных такта HCLK<br>…<br>255 – двести пятьдесят пять дополнительных тактов HCLK<br>Обращения со стороны процессорного ядра и DMA более приоритетны и могут увеличивать паузы между чтениями |
| 7 | SCR_CYCLE | 0 | Бит разрешения циклического повторения выполнения операции циклического чтения и циклического связанного чтения-записи:<br>0 – операция выполняется единожды<br>1 – операция будет автоматически перезапущена после окончания |
| 6 | SCR_SEC | 0 | Бит разрешения исправления одиночных ошибок (SEC) в режиме циклического связанного чтения и записи SCR_MODE = 110. |
| 5 | SCR_ERROR_IE | | Бит разрешения генерации прерывания от блока по событию обнаружению ошибки SCR_ERROR_IF<br>0 – прерывание не генерируется<br>1 – прерывание генерируется |
| 4 | SCR_FINISH_IE | 0 | Бит разрешения генерации прерывания от блока по событию окончания выполнения операции SCR_FINISH_IF:<br>0 – прерывание не генерируется<br>1 – прерывание генерируется |
| 3…1 | SCR_MODE[2:0] | 0 | Режим работы блока SCRUBBER:<br>000 – нет операции<br>001 – первоначальная инициализация<br>010 – проверка первоначальной инициализации<br>011 – режим прямого чтения (реализуем только в LockStep режиме для внешней памяти)<br>100 – режим прямой записи (реализуем только в LockStep режиме для внешней памяти)<br>101 – режим циклического чтения<br>110 – режим циклического связанного чтения и записи |

| | | | |
|---|---|---|---|
| | | | 111 - зарезервировано<br><br>Запись кода команды инициализирует начало выполнения операции, запись команды 000 останавливает выполнение операции. |
| 0 | SCR_EN | 0 | Бит разрешения работы блока SCRUBBER:<br>0 – блок не работает<br>1 – блок работает |

### 20.7.2 *SCR_SADDR*

| Base ADDR= | **0xE004_3000** | Offset= | **0x0000_0004** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | SCR_SADDR | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SCR_SADDR [31:16] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SCR_SADDR [15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | SCR_SADDR [31:0] | 0 | Адрес начала выполнения операции.<br>В режиме прямого чтения и прямой записи значение регистра при чтении/записи автоматически увеличивается на значение 0x04, при достижении значения SCR_FADDR формируется событие окончания выполнения операции SCR_FINISH_IF |

### 20.7.3 *SCR_FADDR*

| Base ADDR= | **0xE004_3000** | Offset= | **0x0000_0008** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | SCR_FADDR | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SCR_FADDR [31:16] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SCR_FADDR [15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | SCR_FADDR [31:0] | 0 | Адрес окончания выполнения операции. |

### 20.7.4 *SCR_DATA*

| Base ADDR= | **0xE004_3000** | Offset= | **0x0000_000C** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | SCR_DATA | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| SCR_DATA [31:16] | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SCR_DATA [15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | SCR_DATA [31:0] | 0 | Данные для выполнения операции прямой записи или чтения.<br>При выполнении функции прямой записи SCR_MODE=100 запись в регистр SCR_DATA инициализирует транзакцию. |

### 20.7.5 *SCR_ECC*

| Base ADDR= | | | **0xE004_3000** | | | Offset= | | **0x0000_0010** | | | Reset= | | 0x0000_0000 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | SCR_ECC | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Зарезервировано | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | | | | | SCR_ECC [7:0] | | | | | | | |
| | | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…8 | Зарезервировано | | Резерв. |
| 7…0 | SCR_ECC [7:0] | 0 | ECC для выполнения операции прямой записи или чтения. |

## 20.8 Описание регистров контроллера тактовых частот CLK_CNTR

Описание регистров синхронного типа – в «Синхронный регистр управления частотой периферийного блока»; описание регистров асинхронного типа – в «Асинхронный регистр управления частотой периферийного блока».

Таблица 134 – Регистры контроллера тактовых частот

| **Базовый Адрес** | | **Название** | **Описание** |
|---|---|---|---|
| 0x4000_0000 | | CLK_CNTR | |
| **Смещение** | | | |
| 0x0000 | 0 | KEY_ADDR | Регистр ключа доступа к регистрам. Для разрешения доступа к остальным регистрам блока на запись в регистр необходимо записать значение 0x8555AAA1 |

| Базовый Адрес | | Название | Описание |
|---|---|---|---|
| 0x4000_0000 | | CLK_CNTR | |
| **Смещение** | | | |
| 0x0004 | 1 | MAX_CLK | Регистр управления системной частотой микроконтроллера MAX CLK |
| 0x0008 | 2 | CPU_CLK | Регистр управления частотой системных шин и ядра микроконтроллера CPU CLK |
| 0x000c | 3 | PER0_CLK | Регистр разрешения тактирования шин APB периферийных блоков 0 |
| 0x0010 | 4 | PER1_CLK | Регистр разрешения тактирования шин APB периферийных блоков 1 |
| 0x0014 | 5 | CPU_CHK0 | Регистр управления блоком контроля частоты системных шин и ядра микроконтроллера CPU CLK |
| 0x0018 | 6 | CPU_CHK1 | Регистр управления блоком контроля частоты системных шин и ядра микроконтроллера CPU CLK |
| 0x001c | 7 | CPU_CHK2 | Регистр управления блоком контроля частоты системных шин и ядра микроконтроллера CPU CLK |
| 0x0020 | 8 | CPU_STAT | Регистр статуса частоты системных шин и ядра микроконтроллера CPU CLK |
| 0x0024 | 9 | LSI_CLK | Регистр управления LSI |
| 0x0028 | 10 | LSI_CHK0 | Регистр управления блоком контроля синхросигнала LSI |
| 0x002c | 11 | LSI_CHK1 | Регистр управления блоком контроля синхросигнала LSI |
| 0x0030 | 12 | LSI_CHK2 | Регистр управления блоком контроля синхросигнала LSI |
| 0x0034 | 13 | LSI_STAT | Регистр статуса синхросигнала LSI |
| 0x0038 | 14 | HSI_STAT | Регистр статуса синхросигнала HSI |
| 0x003c | 15 | LSE_CLK | Регистр управления LSE |
| 0x0040 | 16 | LSE_CHK0 | Регистр управления блоком контроля синхросигнала LSE |
| 0x0044 | 17 | LSE_CHK1 | Регистр управления блоком контроля синхросигнала LSE |
| 0x0048 | 18 | LSE_CHK2 | Регистр управления блоком контроля синхросигнала LSE |
| 0x004c | 19 | LSE_STAT | Регистр статуса синхросигнала LSE |
| 0x0050 | 20 | HSE0_CLK | Регистр управления HSE0 |
| 0x0054 | 21 | HSE0_CHK0 | Регистр управления блоком контроля синхросигнала HSE0 |
| 0x0058 | 22 | HSE0_CHK1 | Регистр управления блоком контроля синхросигнала HSE0 |
| 0x005c | 23 | HSE0_CHK2 | Регистр управления блоком контроля синхросигнала HSE0 |
| 0x0060 | 24 | HSE0_STAT | Регистр статуса синхросигнала HSE0 |
| 0x0064 | 25 | HSE1_CLK | Регистр управления HSE1 |
| 0x0068 | 26 | HSE1_CHK0 | Регистр управления блоком контроля синхросигнала HSE1 |
| 0x006c | 27 | HSE1_CHK1 | Регистр управления блоком контроля синхросигнала HSE1 |
| 0x0070 | 28 | HSE1_CHK2 | Регистр управления блоком контроля синхросигнала HSE1 |
| 0x0074 | 29 | HSE1_STAT | Регистр статуса синхросигнала HSE1 |
| 0x0078 | 30 | PLL0_CLK | Регистр управления блоком PLL0 |

| Базовый Адрес | | Название | Описание |
|---|---|---|---|
| 0x4000_0000 | | CLK_CNTR | |
| **Смещение** | | | |
| 0x007c | 31 | PLL0_CHK0 | Регистр управления блоком контроля синхросигнала PLL 0 |
| 0x0080 | 32 | PLL0_CHK1 | Регистр управления блоком контроля синхросигнала PLL 0 |
| 0x0084 | 33 | PLL0_CHK2 | Регистр управления блоком контроля синхросигнала PLL0 |
| 0x0088 | 34 | PLL0_STAT | Регистр статуса синхросигнала PLL0 |
| 0x008c | 35 | PLL1_CLK | Регистр управления блоком PLL0 |
| 0x0090 | 36 | PLL1_CHK0 | Регистр управления блоком контроля синхросигнала PLL 0 |
| 0x0094 | 37 | PLL1_CHK1 | Регистр управления блоком контроля синхросигнала PLL 0 |
| 0x0098 | 38 | PLL1_CHK2 | Регистр управления блоком контроля синхросигнала PLL0 |
| 0x009c | 39 | PLL1_STAT | Регистр статуса синхросигнала PLL0 |
| 0x00a0 | 40 | PLL2_CLK | Регистр управления блоком PLL0 |
| 0x00a4 | 41 | PLL2_CHK0 | Регистр управления блоком контроля синхросигнала PLL 0 |
| 0x00a8 | 42 | PLL2_CHK1 | Регистр управления блоком контроля синхросигнала PLL 0 |
| 0x00ac | 43 | PLL2_CHK2 | Регистр управления блоком контроля синхросигнала PLL0 |
| 0x00b0 | 44 | PLL2_STAT | Регистр статуса синхросигнала PLL0 |
| 0x00b4 | 45 | PLL3_CLK | Регистр управления блоком PLL3 |
| 0x00b8 | 46 | PLL3_CHK0 | Регистр управления блоком контроля синхросигнала PLL3 |
| 0x00bc | 47 | PLL3_CHK1 | Регистр управления блоком контроля синхросигнала PLL3 |
| 0x00c0 | 48 | PLL3_CHK2 | Регистр управления блоком контроля синхросигнала PLL3 |
| 0x00c4 | 49 | PLL3_STAT | Регистр статуса синхросигнала PLL3 |
| 0x00c8 | 50 | EMAC_CLK | Регистр задания частоты RMII.<br>Асинхронный тип регистра (Используются все источники синхросигналов)<br>(Rev 1.0 – синхронный) |
| 0x00cc | 51 | USB_CLK | Регистр задания частоты USB PHY<br>Асинхронный тип регистра (Используются все источники синхросигналов) |
| 0x00d0 | 52 | USBMAC_CLK | Регистр задания частоты USBMAC. Синхронный тип регистра |
| 0x00d4 | 53 | RTC0_CLK | Регистр задания частоты часов реального времени<br>Асинхронный тип регистра |
| 0x00d8 | 54 | SSP0_CLK | Регистр задания частоты блока SSP0<br>Асинхронный тип регистра |
| 0x00dc | 55 | SSP1_CLK | Регистр задания частоты блока SSP1<br>Асинхронный тип регистра |
| 0x00e0 | 56 | CAN0_CLK | Регистр задания частоты блока CAN0<br>Синхронный тип регистра задания частоты периферийного блока (Используется только синхросигнал CPU деленный в заданное количество раз) |
| 0x00e4 | 57 | CAN1_CLK | Регистр задания частоты блока CAN1<br>Синхронный тип регистра |

| Базовый Адрес | | Название | Описание |
|---|---|---|---|
| 0x4000_0000 | | CLK_CNTR | |
| **Смещение** | | | |
| 0x00e8 | 58 | UART0_CLK | Регистр задания частоты блока UART0<br>Асинхронный тип регистра |
| 0x00ec | 59 | UART1_CLK | Регистр задания частоты блока UART1<br>Асинхронный тип регистра |
| 0x00f0 | 60 | UART2_CLK | Регистр задания частоты блока UART2<br>Асинхронный тип регистра |
| 0x00f4 | 61 | UART3_CLK | Регистр задания частоты блока UART3<br>Асинхронный тип регистра |
| 0x00f8 | 62 | MIL_CLK | Зарезервировано |
| 0x00fc | 63 | TIMER0_CLK | Регистр задания частоты таймера TIMER0<br>Синхронный тип регистра |
| 0x0100 | 64 | TIMER1_CLK | Регистр задания частоты таймера TIMER1<br>Синхронный тип регистра |
| 0x0104 | 65 | TIMER2_CLK | Регистр задания частоты таймера TIMER2<br>Синхронный тип регистра |
| 0x010c | 66 | TIMER3_CLK | Регистр задания частоты таймера TIMER3<br>Синхронный тип регистра |
| 0x0110 | 67 | CAP0_CLK | Регистр задания частоты блока захвата CAP0<br>Синхронный тип регистра |
| 0x0114 | 68 | CAP1_CLK | Регистр задания частоты блока захвата CAP1<br>Синхронный тип регистра |
| 0x0118 | 69 | CAP2_CLK | Регистр задания частоты блока захвата CAP2<br>Синхронный тип регистра |
| 0x011c | 70 | CAP3_CLK | Регистр задания частоты блока захвата CAP3<br>Синхронный тип регистра |
| 0x0120 | 71 | QEP0_CLK | Регистр задания частоты блока квадратурного энкодера QEP0<br>Синхронный тип регистра |
| 0x0124 | 72 | QEP1_CLK | Регистр задания частоты блока квадратурного энкодера QEP1<br>Синхронный тип регистра |
| 0x0128 | 73 | PWM0_CLK | Регистр задания частоты блока ШИМ PWM0<br>Синхронный тип регистра |
| 0x012c | 74 | PWM1_CLK | Регистр задания частоты блока ШИМ PWM1<br>Синхронный тип регистра |
| 0x0130 | 75 | PWM2_CLK | Регистр задания частоты блока ШИМ PWM2<br>Синхронный тип регистра |
| 0x0134 | 76 | PWM3_CLK | Регистр задания частоты блока ШИМ PWM3<br>Синхронный тип регистра |
| 0x0138 | 77 | PWM4_CLK | Регистр задания частоты блока ШИМ PWM4<br>Синхронный тип регистра |
| 0x013c | 78 | PWM5_CLK | Регистр задания частоты блока ШИМ PWM5<br>Синхронный тип регистра |
| 0x0140 | 79 | PWM6_CLK | Регистр задания частоты блока ШИМ PWM6<br>Синхронный тип регистра |
| 0x0144 | 80 | PWM7_CLK | Регистр задания частоты блока ШИМ PWM7<br>Синхронный тип регистра |
| 0x0148 | 81 | PWM8_CLK | Регистр задания частоты блока ШИМ PWM8<br>Синхронный тип регистра |
| 0x014c | 82 | ADC0_CLK | Регистр задания частоты блока АЦП ADC0<br>Асинхронный тип регистра |
| 0x01d0 | 83 | ADC1_CLK | Регистр задания частоты блока АЦП ADC1<br>Асинхронный тип регистра |
| 0x01d4 | 84 | ADC2_CLK | Регистр задания частоты блока АЦП ADC2<br>Асинхронный тип регистра |

| Базовый Адрес | | Название | Описание |
|---|---|---|---|
| 0x4000_0000 | | CLK_CNTR | |
| **Смещение** | | | |
| 0x01d8 | 85 | CRDC_CLK | Регистр задания частоты блока тригонометрических вычислений CRDC Асинхронный тип регистра |
| 0x01dc | 86 | CANFD_CLK | Регистр задания частоты блока CANFD Асинхронный тип регистра |
| 0x01e0 | 87 | CRPT_CLK | Регистр задания частоты блока CRPT Асинхронный тип регистра |
| 0x01e4 | 88 | SDIO_CLK | Регистр задания частоты блока SDIO Асинхронный тип регистра |

### 20.8.1 *KEY*

| Base ADDR= | **0x4000_0000** | Offset= | **0x0000_0000** |
|---|---|---|---|
| REG Name: | KEY | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | KEY[31:0] | 0000_0000 | При записи в регистр значения 0x8555AAA1 открывается возможность записи в другие регистры блока CLK_CNTR |

### 20.8.2 *MAX_CLK*

| Base ADDR= | **0x4000_0000** | Offset= | **0x0000_0004** |
|---|---|---|---|
| REG Name: | MAX_CLK | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | | | | | | | | | SELECT[3:0] | | | |
| | | | | | | | | | | | | RW | RW | RW | RW |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…4 | зарезервировано | - | Зарезервировано |
| 3..0 | SELECT[3:0] | - | Номер источника тактовой частоты для формирования частоты SELCLK<br>0000 – Генератор HSI<br>0001 – Генератор HSI/2<br>0010 – Генератор HSE0<br>0011 – Генератор HSE0/2<br>0100 – Генератор HSE1<br>0101 – Генератор HSE1/2<br>0110 – Генератор LSI<br>0111 – Генератор LSE<br>1000 – Генератор PLL0<br>1001 – Генератор PLL1<br>1010 – Генератор PLL2<br>1011 – Генератор PLL3<br>1100 – Зарезервировано<br>1101 – Зарезервировано<br>1110 – Зарезервировано<br>1111 – Зарезервировано |

### 20.8.3 *CPU_CLK*

| Base ADDR= | | **0x4000_0000** | | | Offset= | | **0x0000_0008** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | CPU_CLK | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | CLR_CHK_EVENT3 | CLR_CHK_EVENT2 | CLR_CHK_EVENT1 | CLR_CHK_EVENT0 | CLR_CHK_SHIFT_REG1 | CLR_CHK_SHIFT_REG0 | EN_CHK | EN_CHK_EVENT3 | EN_CHK_EVENT2 | EN_CHK_EVENT1 | EN_CHK_EVENT0 |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DIV[15:0] | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…27 | - | - | Зарезервировано |
| 26 | CLR_CHK_EVENT3 | - | Бит сброса флага EVENT3<br>1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |
| 25 | CLR_CHK_EVENT2 | - | Бит сброса флага EVENT2<br>1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |
| 24 | CLR_CHK_EVENT1 | - | Бит сброса флага EVENT1<br>1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |
| 23 | CLR_CHK_EVENT0 | - | Бит сброса флага EVENT0<br>1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |
| 22 | CLR_CHK_SHIFT_REG1 | | Бит сброса теневого регистра максимального значения регистра SHIFT_REG1<br>1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |
| 21 | CLR_CHK_SHIFT_REG0 | | Бит сброса теневого регистра максимального значения регистра SHIFT_REG0<br>1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |
| 20 | EN_CHK | | Бит разрешения контроля частоты CPUCLK блоком CLK_CHECKER<br>0 – частота не контролируется<br>1 – монитор работает |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 19 | EN_CHK_EVENT3 | | Бит разрешения аварийного переключения на частоту HSI (очень высокая частота)<br>0 – ничего не делать при событии EVENT3<br>1 – перейти на частоту HSI при событии EVENT3<br>Имеет смысл только при EN_CHK = 1<br>Для CPUCLK аварийное переключение не предусмотрено |
| 18 | EN_CHK_EVENT2 | | Бит разрешения аварийного переключения на частоту HSI (высокая частота)<br>0 – ничего не делать при событии EVENT2<br>1 – перейти на частоту HSI при событии EVENT2<br>Имеет смысл только при EN_CHK = 1<br>Для CPUCLK аварийное переключение не предусмотрено |
| 17 | EN_CHK_EVENT1 | | Бит разрешения аварийного переключения на частоту HSI (низкая частота)<br>0 – ничего не делать при событии EVENT1<br>1 – перейти на частоту HSI при событии EVENT1<br>Имеет смысл только при EN_CHK = 1<br>Для CPUCLK аварийное переключение не предусмотрено |
| 16 | EN_CHK_EVENT0 | | Бит разрешения аварийного переключения на частоту HSI (нет частоты)<br>0 – ничего не делать при событии EVENT0<br>1 – перейти на частоту HSI при событии EVENT0<br>Имеет смысл только при EN_CHK = 1<br>Для CPUCLK аварийное переключение не предусмотрено |
| 15…0 | DIV[15:0] | | Делитель частоты MAX_CLK для формирования частоты CPUCLK<br>CPUCLK = MAX_CLK/(DIV+1) |

### 20.8.4 *CPU_CHK0*

| Base ADDR= | **0x4000_0000** | Offset= | **0x0000_0014** |
|---|---|---|---|
| REG Name: | CPU_CHK0 | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PRES_REG0[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PRES_REG2[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…16 | PRES_REG0[15:0] | | Значение регистра делителя частоты сброса для счетчика SHIFT_REG1<br>RST(SHIFT_REG1) = HSI / (PRES_REG0+1) |
| 15…0 | PRES_REG2[15:0] | | Значение регистра делителя частоты HSI для счетчика на SHIFT_REG0<br>CLK(SHIFT_REG0) = HSI / (PRES_REG2+1) |

### 20.8.5 *CPU_CHK1*

| Base ADDR= | | **0x4000_0000** | | | Offset= | | **0x0000_0018** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | CPU_CHK1 | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PRES_REG3[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PRES_REG1[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…16 | PRES_REG3[15:0] | | Значение регистра делителя частоты сброса для счетчика SHIFT_REG0<br>RST(SHIFT_REG0) = CPUCLK / (PRES_REG3+1) |
| 15…0 | PRES_REG1[15:0] | | Значение регистра делителя частоты HSI для счетчика на SHIFT_REG1<br>CLK(SHIFT_REG1) = CPUCLK / (PRES_REG1+1) |

### 20.8.6 *CPU_CHK2*

| Base ADDR= | | **0x4000_0000** | | | Offset= | | **0x0000_001C** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | CPU_CHK2 | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| BASE_REG3[7:0] | | | | | | | | BASE_REG2[7:0] | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BASE_REG1[7:0] | | | | | | | | BASE_REG0[7:0] | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…24 | BASE_REG3[7:0] | | Регистр сравнения частоты для определения сильного увеличения частоты (EVENT3) |
| 23…16 | BASE_REG2[7:0] | | Регистр сравнения частоты для определения увеличения частоты (EVENT2) |
| 15…8 | BASE_REG1[7:0] | | Регистр сравнения частоты для определения снижения частоты (EVENT1) |
| 7…0 | BASE_REG0[7:0] | | Регистр сравнения частоты для определения исчезновения частоты (EVENT0) |

### 20.8.7 *CPU_STAT*

| Base ADDR= | | **0x4000_0000** | | | Offset= | | **0x0000_0020** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | CPU_STAT | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | EVENT3 | EVENT2 | EVENT1 | EVENT0 |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MAX_CHK_SHIFT_REG1[7:0] | | | | | | | | MAX_CHK_SHIFT_REG0[7:0] | | | | | | | |
| | | | | | | | | | | | | | | | |

| **Бит** | **Имя** | **Значение** | **Описание** |
|---|---|---|---|
| 31…20 | - | | Зарезервировано |
| 19 | EVENT3 | | Флаг возникновения события EVENT3(очень высокая частота)<br>0 – нет события<br>1 – есть событие |
| 18 | EVENT2 | | Флаг возникновения события EVENT2(высокая частота)<br>0 – нет события<br>1 – есть событие |
| 17 | EVENT1 | | Флаг возникновения события EVENT1 (низкая частота)<br>0 – нет события<br>1 – есть событие |
| 16 | EVENT0 | | Флаг возникновения события EVENT0 (нет частоты)<br>0 – нет события<br>1 – есть событие |
| 15…8 | MAX_CHK_SHIFT_REG1[7:0] | | Максимальное значение регистра SHIFT_REG1 с момента последнего сброса теневого регистра |
| 7…0 | MAX_CHK_SHIFT_REG0[7:0] | | Максимальное значение регистра SHIFT_REG0 с момента последнего сброса теневого регистра |

### 20.8.8 *LSI_CLK*[2]

| Base ADDR= | | **0x4000_0000** | | | Offset= | | **0x0000_0024** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | LSI_CLK | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | CLR_CHK_EVENT3 | CLR_CHK_EVENT2 | CLR_CHK_EVENT1 | CLR_CHK_EVENT0 | CLR_CHK_SHIFT_REG1 | CLR_CHK_SHIFT_REG0 | EN_CHK | EN_CHK_EVENT3 | EN_CHK_EVENT2 | EN_CHK_EVENT1 | EN_CHK_EVENT0 |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…27 | - | - | Зарезервировано |
| 26 | CLR_CHK_EVENT3 | - | Бит сброса флага EVENT3<br>1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |
| 25 | CLR_CHK_EVENT2 | - | Бит сброса флага EVENT2<br>1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |
| 24 | CLR_CHK_EVENT1 | - | Бит сброса флага EVENT1<br>1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |
| 23 | CLR_CHK_EVENT0 | - | Бит сброса флага EVENT0<br>1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |
| 22 | CLR_CHK_SHIFT_REG1 | | Бит сброса теневого регистра максимального значения регистра SHIFT_REG1<br>1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |
| 21 | CLR_CHK_SHIFT_REG0 | | Бит сброса теневого регистра максимального значения регистра SHIFT_REG0<br>1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |
| 20 | EN_CHK | | Бит разрешения контроля частоты LSI<br>0 – частота не контролируется<br>1 – монитор работает |

[2] Управление генератором LSI осуществляется через регистры батарейного домена.

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 19 | EN_CHK_EVENT3 | | Бит разрешения аварийного переключения на частоту HSI (очень высокая частота)<br>0 – ничего не делать при событии EVENT3<br>1 – перейти на частоту HSI при событии EVENT3<br>Имеет смысл только при EN_CHK = 1 |
| 18 | EN_CHK_EVENT2 | | Бит разрешения аварийного переключения на частоту HSI (высокая частота)<br>0 – ничего не делать при событии EVENT2<br>1 – перейти на частоту HSI при событии EVENT2<br>Имеет смысл только при EN_CHK = 1 |
| 17 | EN_CHK_EVENT1 | | Бит разрешения аварийного переключения на частоту HSI (низкая частота)<br>0 – ничего не делать при событии EVENT1<br>1 – перейти на частоту HSI при событии EVENT1<br>Имеет смысл только при EN_CHK = 1 |
| 16 | EN_CHK_EVENT0 | | Бит разрешения аварийного переключения на частоту HSI (нет частоты)<br>0 – ничего не делать при событии EVENT0<br>1 – перейти на частоту HSI при событии EVENT0<br>Имеет смысл только при EN_CHK = 1 |
| 15…0 | - | | Зарезервировано |

### 20.8.9 *LSI_CHK0*

| Base ADDR= | **0x4000_0000** | Offset= | **0x0000_0028** |
|---|---|---|---|
| REG Name: | LSI_CHK0 | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PRES_REG0[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PRES_REG2[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…16 | PRES_REG0[15:0] | | Значение регистра делителя частоты сброса для счетчика SHIFT_REG1<br>RST(SHIFT_REG1) = HSI / (PRES_REG0+1) |
| 15…0 | PRES_REG2[15:0] | | Значение регистра делителя частоты HSI для счетчика на SHIFT_REG0<br>CLK(SHIFT_REG0) = HSI / (PRES_REG2+1) |

### 20.8.10 *LSI_CHK1*

| Base ADDR= | | **0x4000_0000** | | | Offset= | | **0x0000_002C** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | LSI_CHK1 | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PRES_REG3[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PRES_REG1[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…16 | PRES_REG3[15:0] | | Значение регистра делителя частоты сброса для счетчика SHIFT_REG0<br>RST(SHIFT_REG0) = LSI / (PRES_REG3+1) |
| 15…0 | PRES_REG1[15:0] | | Значение регистра делителя частоты HSI для счетчика на SHIFT_REG1<br>CLK(SHIFT_REG1) = LSI / (PRES_REG1+1) |

### 20.8.11 *LSI_CHK2*

| Base ADDR= | | **0x4000_0000** | | | Offset= | | **0x0000_0030** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | LSI_CHK2 | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| BASE_REG3[7:0] | | | | | | | | BASE_REG2[7:0] | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BASE_REG1[7:0] | | | | | | | | BASE_REG0[7:0] | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…24 | BASE_REG3[7:0] | | Регистр сравнения частоты для определения сильного увеличения частоты (EVENT3) |
| 23…16 | BASE_REG2[7:0] | | Регистр сравнения частоты для определения увеличения частоты (EVENT2) |
| 15…8 | BASE_REG1[7:0] | | Регистр сравнения частоты для определения снижения частоты (EVENT1) |
| 7…0 | BASE_REG0[7:0] | | Регистр сравнения частоты для определения исчезновения частоты (EVENT0) |

### 20.8.12 *LSI_STAT*

| Base ADDR= | | **0x4000_0000** | | | Offset= | | **0x0000_0034** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | LSI_STAT | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | LSI_RDY | EVENT3 | EVENT2 | EVENT1 | EVENT0 |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MAX_CHK_SHIFT_REG1[7:0] | | | | | | | | MAX_CHK_SHIFT_REG0[7:0] | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…21 | - | | Зарезервировано |
| 20 | LSI_READY | - | Флаг готовности генератора LSI<br>0 – генератор не готов или выключен<br>1 – генератор готов |
| 19 | EVENT3 | | Флаг возникновения события EVENT3(очень высокая частота)<br>0 – нет события<br>1 – есть событие |
| 18 | EVENT2 | | Флаг возникновения события EVENT2(высокая частота)<br>0 – нет события<br>1 – есть событие |
| 17 | EVENT1 | | Флаг возникновения события EVENT1 (низкая частота)<br>0 – нет события<br>1 – есть событие |
| 16 | EVENT0 | | Флаг возникновения события EVENT0 (нет частоты)<br>0 – нет события<br>1 – есть событие |
| 15…8 | MAX_CHK_SHIFT_REG1[7:0] | | Максимальное значение регистра SHIFT_REG1 с момента последнего сброса теневого регистра |
| 7…0 | MAX_CHK_SHIFT_REG0[7:0] | | Максимальное значение регистра SHIFT_REG0 с момента последнего сброса теневого регистра |

### 20.8.13 *HSI_STAT*[3]

| Base ADDR= | | **0x4000_0000** | | | Offset= | | **0x0000_0038** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | HSI_STAT | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | HSI_RDY | - | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…21 | - | | Зарезервировано |
| 20 | HSI_READY | - | Флаг готовности генератора HSI<br>0 – генератор не готов или выключен<br>1 – генератор готов |
| 19…0 | - | - | Зарезервировано |

### 20.8.14 *LSE_CLK*[4]

| Base ADDR= | | **0x4000_0000** | | | Offset= | | **0x0000_003C** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | LSE_CLK | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | CLR_CHK_EVENT3 | CLR_CHK_EVENT2 | CLR_CHK_EVENT1 | CLR_CHK_EVENT0 | CLR_CHK_SHIFT_REG1 | CLR_CHK_SHIFT_REG0 | EN_CHK | EN_CHK EVENT3 | EN_CHK EVENT2 | EN_CHK EVENT1 | EN_CHK EVENT0 |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…27 | - | - | Зарезервировано |

[3] Управление генератором HSI осуществляется через регистры батарейного домена.

[4] Управление генератором LSE осуществляется через регистры батарейного домена.

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 26 | CLR_CHK_EVENT3 | - | Бит сброса флага EVENT3<br>1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |
| 25 | CLR_CHK_EVENT2 | - | Бит сброса флага EVENT2<br>1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |
| 24 | CLR_CHK_EVENT1 | - | Бит сброса флага EVENT1<br>1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |
| 23 | CLR_CHK_EVENT0 | - | Бит сброса флага EVENT0<br>1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |
| 22 | CLR_CHK_SHIFT_REG1 | | Бит сброса теневого регистра максимального значения регистра SHIFT_REG1<br>1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |
| 21 | CLR_CHK_SHIFT_REG0 | | Бит сброса теневого регистра максимального значения регистра SHIFT_REG0<br>Запись 1 сбрасывает теневой регистр |
| 20 | EN_CHK | | Бит разрешения контроля частоты LSE<br>0 – частота не контролируется<br>1 – монитор работает |
| 19 | EN_CHK_EVENT3 | | Бит разрешения аварийного переключения на частоту HSI (очень высокая частота)<br>0 – ничего не делать при событии EVENT3<br>1 – перейти на частоту HSI при событии EVENT3<br>Имеет смысл только при EN_CHK = 1 |
| 18 | EN_CHK_EVENT2 | | Бит разрешения аварийного переключения на частоту HSI (высокая частота)<br>0 – ничего не делать при событии EVENT2<br>1 – перейти на частоту HSI при событии EVENT2<br>Имеет смысл только при EN_CHK = 1 |
| 17 | EN_CHK_EVENT1 | | Бит разрешения аварийного переключения на частоту HSI (низкая частота)<br>0 – ничего не делать при событии EVENT1<br>1 – перейти на частоту HSI при событии EVENT1<br>Имеет смысл только при EN_CHK = 1 |
| 16 | EN_CHK_EVENT0 | | Бит разрешения аварийного переключения на частоту HSI (нет частоты)<br>0 – ничего не делать при событии EVENT0<br>1 – перейти на частоту HSI при событии EVENT0<br>Имеет смысл только при EN_CHK = 1 |
| 15…0 | - | | Зарезервировано |

### 20.8.15 *LSE_CHK0*

| Base ADDR= | **0x4000_0000** | Offset= | **0x0000_0040** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|

| REG Name: | | LSE_CHK0 | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PRES_REG0[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PRES_REG2[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…16 | PRES_REG0[15:0] | | Значение регистра делителя частоты сброса для счетчика SHIFT_REG1<br>RST(SHIFT_REG1) = HSI / (PRES_REG0+1) |
| 15…0 | PRES_REG2[15:0] | | Значение регистра делителя частоты HSI для счетчика на SHIFT_REG0<br>CLK(SHIFT_REG0) = HSI / (PRES_REG2+1) |

### 20.8.16 *LSE_CHK1*

| Base ADDR= | **0x4000_0000** | | | | Offset= | **0x0000_0044** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | LSE_CHK1 | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PRES_REG3[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PRES_REG1[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…16 | PRES_REG3[15:0] | | Значение регистра делителя частоты сброса для счетчика SHIFT_REG0<br>RST(SHIFT_REG0) = LSE / (PRES_REG3+1) |
| 15…0 | PRES_REG1[15:0] | | Значение регистра делителя частоты HSI для счетчика на SHIFT_REG1<br>CLK(SHIFT_REG1) = LSE / (PRES_REG1+1) |

### 20.8.17 *LSE_CHK2*

| Base ADDR= | **0x4000_0000** | Offset= | **0x0000_0048** |
|---|---|---|---|
| REG Name: | LSE_CHK2 | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BASE_REG3[7:0] | | | | | | | | BASE_REG2[7:0] | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BASE_REG1[7:0] | | | | | | | | BASE_REG0[7:0] | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…24 | BASE_REG3[7:0] | | Регистр сравнения частоты для определения сильного увеличения частоты (EVENT3) |
| 23…16 | BASE_REG2[7:0] | | Регистр сравнения частоты для определения увеличения частоты (EVENT2) |
| 15…8 | BASE_REG1[7:0] | | Регистр сравнения частоты для определения снижения частоты (EVENT1) |
| 7…0 | BASE_REG0[7:0] | | Регистр сравнения частоты для определения исчезновения частоты (EVENT0) |

### 20.8.18 *LSE_STAT*

| Base ADDR= | **0x4000_0000** | Offset= | **0x0000_004C** |
|---|---|---|---|
| REG Name: | LSE_STAT | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | LSE_RDY | EVENT3 | EVENT2 | EVENT1 | EVENT0 |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MAX_CHK_SHIFT_REG1[7:0] | | | | | | | | MAX_CHK_SHIFT_REG0[7:0] | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…20 | - | | Зарезервировано |
| 20 | LSE_READY | - | Флаг готовности генератора LSE<br>0 – генератор не готов или выключен<br>1 – генератор готов |
| 19 | EVENT3 | | Флаг возникновения события EVENT3(очень высокая частота)<br>0 – нет события<br>1 – есть событие |
| 18 | EVENT2 | | Флаг возникновения события EVENT2(высокая частота)<br>0 – нет события<br>1 – есть событие |
| 17 | EVENT1 | | Флаг возникновения события EVENT1 (низкая частота)<br>0 – нет события<br>1 – есть событие |
| 16 | EVENT0 | | Флаг возникновения события EVENT0 (нет частоты) |

| | | | |
|---|---|---|---|
| | | | 0 – нет события<br>1 – есть событие |
| 15…8 | MAX_CHK_SHIFT_REG1[7:0] | | Максимальное значение регистра SHIFT_REG1 с момента последнего сброса теневого регистра |
| 7…0 | MAX_CHK_SHIFT_REG0[7:0] | | Максимальное значение регистра SHIFT_REG0 с момента последнего сброса теневого регистра |

### 20.8.19 *HSEn_CLK*

| Base ADDR= | **0x4000_0000** | Offset= | **0x0000_0050**<br>**0x0000_0064** |
|---|---|---|---|

| REG Name: | | HSEn_CLK | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | - | | | | CLR_CHK_EVENT3 | CLR_CHK_EVENT2 | CLR_CHK_EVENT1 | CLR_CHK_EVENT0 | CLR_CHK_SHIFT_REG1 | CLR_CHK_SHIFT_REG0 | EN_CHK | EN_CHK EVENT3 | EN_CHK EVENT2 | EN_CHK EVENT1 | EN_CHK EVENT0 |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…30 | - | - | Зарезервировано |
| 29 | - | - | Зарезервировано |
| 28 | - | - | Зарезервировано |
| 27 | - | - | Зарезервировано |
| 26 | CLR_CHK_EVENT3 | - | Бит сброса флага EVENT3<br>1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |
| 25 | CLR_CHK_EVENT2 | - | Бит сброса флага EVENT2<br>1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |
| 24 | CLR_CHK_EVENT1 | - | Бит сброса флага EVENT1<br>1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |
| 23 | CLR_CHK_EVENT0 | - | Бит сброса флага EVENT0<br>1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |
| 22 | CLR_CHK_SHIFT_REG1 | | Бит сброса теневого регистра максимального значения регистра SHIFT_REG1 |

| | | | |
|---|---|---|---|
| | | | 1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |
| 21 | CLR_CHK_SHIFT_REG0 | | Бит сброса теневого регистра максимального значения регистра SHIFT_REG0<br>1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |
| 20 | EN_CHK | | Бит разрешения контроля частоты HSEn<br>0 – частота не контролируется<br>1 – монитор работает |
| 19 | EN_CHK_EVENT3 | | Бит разрешения аварийного переключения на частоту HSI (очень высокая частота)<br>0 – ничего не делать при событии EVENT3<br>1 – перейти на частоту HSI при событии EVENT3<br>Имеет смысл только при EN_CHK = 1 |
| 18 | EN_CHK_EVENT2 | | Бит разрешения аварийного переключения на частоту HSI (высокая частота)<br>0 – ничего не делать при событии EVENT2<br>1 – перейти на частоту HSI при событии EVENT2<br>Имеет смысл только при EN_CHK = 1 |
| 17 | EN_CHK_EVENT1 | | Бит разрешения аварийного переключения на частоту HSI (низкая частота)<br>0 – ничего не делать при событии EVENT1<br>1 – перейти на частоту HSI при событии EVENT1<br>Имеет смысл только при EN_CHK = 1 |
| 16 | EN_CHK_EVENT0 | | Бит разрешения аварийного переключения на частоту HSI (нет частоты)<br>0 – ничего не делать при событии EVENT0<br>1 – перейти на частоту HSI при событии EVENT0<br>Имеет смысл только при EN_CHK = 1 |
| 15…0 | - | | Зарезервировано |

### 20.8.20 *HSEn_CHK0*

| Base ADDR= | 0x4000_0000 | Offset= | **0x0000_0054**<br>**0x0000_0068** |
|---|---|---|---|
| REG Name: | HSEn_CHK0 | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PRES_REG0[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PRES_REG2[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…16 | PRES_REG0[15:0] | | Значение регистра делителя частоты сброса для счетчика SHIFT_REG1<br>RST(SHIFT_REG1) = HSI / (PRES_REG0+1) |
| 15…0 | PRES_REG2[15:0] | | Значение регистра делителя частоты HSI для счетчика на SHIFT_REG0<br>CLK(SHIFT_REG0) = HSI / (PRES_REG2+1) |

### 20.8.21 *HSEn_CHK1*

| Base ADDR= | **0x4000_0000** | Offset= | **0x0000_0058**<br>**0x0000_006C** |
|---|---|---|---|
| REG Name: | HSEn_CHK1 | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PRES_REG3[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PRES_REG1[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…16 | PRES_REG3[15:0] | | Значение регистра делителя частоты сброса для счетчика SHIFT_REG0<br>RST(SHIFT_REG0) = HSEn / (PRES_REG3+1) |
| 15…0 | PRES_REG1[15:0] | | Значение регистра делителя частоты HSI для счетчика на SHIFT_REG1<br>CLK(SHIFT_REG1) = HSEn / (PRES_REG1+1) |

### 20.8.22 *HSEn_CHK2*

| Base ADDR= | **0x4000_0000** | Offset= | **0x0000_005C**<br>**0x0000_0070** |
|---|---|---|---|
| REG Name: | HSEn_CHK2 | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BASE_REG3[7:0] | | | | | | | | BASE_REG2[7:0] | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BASE_REG1[7:0] | | | | | | | | BASE_REG0[7:0] | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…24 | BASE_REG3[7:0] | | Регистр сравнения частоты для определения сильного увеличения частоты (EVENT3) |
| 23…16 | BASE_REG2[7:0] | | Регистр сравнения частоты для определения увеличения частоты (EVENT2) |
| 15…8 | BASE_REG1[7:0] | | Регистр сравнения частоты для определения снижения частоты (EVENT1) |
| 7…0 | BASE_REG0[7:0] | | Регистр сравнения частоты для определения исчезновения частоты (EVENT0) |

### 20.8.23 *HSEn_STAT*

| Base ADDR= | **0x4000_0000** | Offset= | **0x0000_0060**<br>**0x0000_0074** |
|---|---|---|---|
| REG Name: | HSEn_STAT | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | HSEn_RDY | EVENT3 | EVENT2 | EVENT1 | EVENT0 |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MAX_CHK_SHIFT_REG1[7:0] | | | | | | | | MAX_CHK_SHIFT_REG0[7:0] | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…21 | - | | Зарезервировано |
| 20 | HSEn_RDY | - | Флаг готовности генератора HSEn<br>0 – генератор не готов или выключен<br>1 – генератор готов |
| 19 | EVENT3 | | Флаг возникновения события EVENT3(очень высокая частота)<br>0 – нет события<br>1 – есть событие |

| | | | |
|---|---|---|---|
| 18 | EVENT2 | | Флаг возникновения события EVENT2(высокая частота)<br>0 – нет события<br>1 – есть событие |
| 17 | EVENT1 | | Флаг возникновения события EVENT1 (низкая частота)<br>0 – нет события<br>1 – есть событие |
| 16 | EVENT0 | | Флаг возникновения события EVENT0 (нет частоты)<br>0 – нет события<br>1 – есть событие |
| 15…8 | MAX_CHK_SHIFT_REG1[7:0] | | Максимальное значение регистра SHIFT_REG1 с момента последнего сброса теневого регистра |
| 7…0 | MAX_CHK_SHIFT_REG0[7:0] | | Максимальное значение регистра SHIFT_REG0 с момента последнего сброса теневого регистра |

### 20.8.24 *PLLn_CLK*

| Base ADDR= | | **0x4000_0000** | | | Offset= | | **0x0000_0078<br>0x0000_008C<br>0x0000_00A0<br>0x0000_00B4** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | PLLn_CLK | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CLR_CHK_EVENT3 | CLR_CHK_EVENT2 | CLR_CHK_EVENT1 | CLR_CHK_EVENT0 | CLR_CHK_SHIFT_REG1 | CLR_CHK_SHIFT_REG0 | EN_CHK | EN_CHK_EVENT0 | EN_CHK_EVENT1 | EN_CHK_EVENT2 | EN_CHK_EVENT3 | SELECT | | | PLLn_ON | PLLn_RDYMODE |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PLLn_DV | | PLLn_N | | | | | | | | | PLLn_Q | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31 | CLR_CHK_EVENT3 | - | Бит сброса флага EVENT3<br>1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |
| 30 | CLR_CHK_EVENT2 | - | Бит сброса флага EVENT2<br>1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |

| | | | |
|---|---|---|---|
| 29 | CLR_CHK_EVENT1 | - | Бит сброса флага EVENT1<br>1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |
| 28 | CLR_CHK_EVENT0 | - | Бит сброса флага EVENT0<br>1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |
| 27 | CLR_CHK_SHIFT_REG1 | | Бит сброса теневого регистра максимального значения регистра SHIFT_REG1<br>1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |
| 26 | CLR_CHK_SHIFT_REG0 | | Бит сброса теневого регистра максимального значения регистра SHIFT_REG0<br>1 – сброс регистра<br>0 – нет сброса<br>Устанавливается в 0 автоматически, запись 1 имеет смысл только при 0 значении бита |
| 25 | EN_CHK | | Бит разрешения контроля частоты PLLn<br>0 – частота не контролируется<br>1 – монитор работает |
| 24 | EN_CHK_EVENT0 | | Бит разрешения аварийного переключения на частоту HSI (нет частоты)<br>0 – ничего не делать при событии EVENT0<br>1 – перейти на частоту HS при событии EVENT0<br>Имеет смысл только при EN_CHK = 1 |
| 23 | EN_CHK_EVENT1 | | Бит разрешения аварийного переключения на частоту HSI (низкая частота)<br>0 – ничего не делать при событии EVENT1<br>1 – перейти на частоту HSI при событии EVENT1<br>Имеет смысл только при EN_CHK = 1 |
| 22 | EN_CHK_EVENT2 | | Бит разрешения аварийного переключения на частоту HSI (высокая частота)<br>0 – ничего не делать при событии EVENT2<br>1 – перейти на частоту HSI при событии EVENT2<br>Имеет смысл только при EN_CHK = 1 |
| 21 | EN_CHK_EVENT3 | | Бит разрешения аварийного переключения на частоту HSI (очень высокая частота)<br>0 – ничего не делать при событии EVENT3<br>1 – перейти на частоту HSI при событии EVENT3<br>Имеет смысл только при EN_CHK = 1 |
| 20…18 | SELECT[2:0] | | Номер источника тактовой частоты<br>000 – Генератор HSI<br>001 – Генератор HSI/2<br>010 – Генератор HSE0<br>011 – Генератор HSE0/2<br>100 – Генератор HSE1<br>101 – Генератор HSE1/2<br>110 и111 – Зарезервировано<br>Переключение частоты необходимо выполнять с включенного источника |
| 17 | PLLn_ON | - | Бит разрешения работы генератора PLLn<br>0 – PLLn выключена<br>1 – PLLn включена |

| | | | |
|---|---|---|---|
| 16 | PLLn_RDYMODE | | Режим работы выходной частоты в неустановившемся состоянии<br>0 – частота не выдается пока PLL_RDY = 0<br>1 – частота выдается не зависимо от PLL_RDY |
| 15…14 | PLLn_DV | - | Выбор диапазона выходной частоты $F_{O_PLL}$<br><br>$F_{O_PLL} = F_{INT_PLL} / (2^{DV})$<br><br><00> – $F_{O_PLL} = F_{INT_PLL}$<br><01> – $F_{O_PLL} = F_{INT_PLL} / 2$<br><10> – $F_{O_PLL} = F_{INT_PLL} /4$<br><11> – недопустимое значение |
| 13…5 | PLLn_N[8:0] | | Коэффициент умножения для получения внутренней тактовой частоты PLL<br>$F_{INT_PLL} = F_{C_PLL} • (N + 1) / (Q + 1)$<br>N = 0….511, но допустимыми являются только значения от 3 до 299 |
| 4…0 | PLLn_Q[4:0] | | Коэффициент деления для получения внутренней тактовой частоты PLL<br>$F_{INT_PLL} = F_{C_PLL} • (N + 1) / (Q + 1)$<br>Q = 0….31 |

### 20.8.25 *PLLn_CHK0*

| Base ADDR= | **0x4000_0000** | Offset= | **0x0000_007C**<br>**0x0000_0090**<br>**0x0000_00A4**<br>**0x0000_00B8** |
|---|---|---|---|
| REG Name: | PLLn_CHK0 | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PRES_REG0[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PRES_REG2[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…16 | PRES_REG0[15:0] | | Значение регистра делителя частоты HSI для счетчика на SHIFT_REG0<br>CLK(SHIFT_REG0) = HSI / (PRES_REG0+1) |
| 15…0 | PRES_REG2[15:0] | | Значение регистра делителя частоты PLLndля счетчика на SHIFT_REG1<br>CLK(SHIFT_REG1) = PLLn / (PRES_REG2+1) |

### 20.8.26 *PLLn_CHK1*

| Base ADDR= | **0x4000_0000** | Offset= | **0x0000_0080**<br>**0x0000_0094**<br>**0x0000_00A8**<br>**0x0000_00BC** |
|---|---|---|---|
| REG Name: | PLLn_CHK1 | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PRES_REG3[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PRES_REG1[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…16 | PRES_REG3[15:0] | | Значение регистра делителя частоты PLLn сброса для счетчика SHIFT_REG0<br>RST(SHIFT_REG0) = PLLn / (PRES_REG3+1) |
| 15…0 | PRES_REG1[15:0] | | Значение регистра делителя частоты сброса для счетчика SHIFT_REG1<br>RST(SHIFT_REG1) = HSI / (PRES_REG1+1) |

### 20.8.27 *PLLn_CHK2*

| Base ADDR= | **0x4000_0000** | Offset= | **0x0000_0084**<br>**0x0000_0098**<br>**0x0000_00AC**<br>**0x0000_00C0** |
|---|---|---|---|
| REG Name: | PLLn_CHK2 | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BASE_REG3[7:0] | | | | | | | | BASE_REG2[7:0] | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BASE_REG1[7:0] | | | | | | | | BASE_REG0[7:0] | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…24 | BASE_REG3[7:0] | | Регистр сравнения частоты для определения сильного увеличения частоты (EVENT3) |
| 23…16 | BASE_REG2[7:0] | | Регистр сравнения частоты для определения увеличения частоты (EVENT2) |
| 15…8 | BASE_REG1[7:0] | | Регистр сравнения частоты для определения снижения частоты (EVENT1) |
| 7…0 | BASE_REG0[7:0] | | Регистр сравнения частоты для определения исчезновения частоты (EVENT0) |

### 20.8.28 *PLLn_STAT*

| Base ADDR= | **0x4000_0000** | Offset= | **0x0000_0088<br>0x0000_009C<br>0x0000_00B0<br>0x0000_00C4** |
|---|---|---|---|
| REG Name: | PLLn_STAT | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | PLLn_RDY | EVENT3 | EVENT2 | EVENT1 | EVENT0 |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MAX_CHK_SHIFT_REG1[7:0] | | | | | | | | MAX_CHK_SHIFT_REG0[7:0] | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…21 | - | | Зарезервировано |
| 20 | PLLn_READY | - | Флаг готовности генератора PLLn<br>0 – генератор не готов или выключен<br>1 – генератор готов |
| 19 | EVENT3 | | Флаг возникновения события EVENT3(очень высокая частота)<br>0 – нет события<br>1 – есть событие |
| 18 | EVENT2 | | Флаг возникновения события EVENT2(высокая частота)<br>0 – нет события<br>1 – есть событие |
| 17 | EVENT1 | | Флаг возникновения события EVENT1(низкая частота)<br>0 – нет события<br>1 – есть событие |
| 16 | EVENT0 | | Флаг возникновения события EVENT0 (нет частоты)<br>0 – нет события<br>1 – есть событие |
| 15…8 | MAX_CHK_SHIFT_REG1[7:0] | | Максимальное значение регистра SHIFT_REG1 с момента последнего сброса теневого регистра |
| 7…0 | MAX_CHK_SHIFT_REG0[7:0] | | Максимальное значение регистра SHIFT_REG0 с момента последнего сброса теневого регистра |

### 20.8.29 *Регистры синхросигналов периферийных блоков*

Синхросигналы периферийных блоков подразделяются на:

- равные системной тактовой частоте CPU. Источником синхросигналов для таких блоков служит системная частота процессорного ядра CPU_CLK. В этом случае отдельный регистр для управления частотой блока не реализован. Для организации тактирования таких блоков достаточно включить соответствующий синхросигнал PCLK в одном из регистров PER_CLK. Частота таких блоков равна частоте процессорного ядра и управляется регистром CPU_CLK;
- Синхронный относительно системной тактовой частоты, имеющий с системной частотой общий источник, но с возможностью управления. В этом случае системный синхросигнал CPU и синхросигнал блока имеют общую опорную частоту MAX_CLK, настройка которой производится в регистре MAX_CLK. Для блока реализован отдельный регистр управления, позволяющий делить опорную тактовую частоту. В таблице регистров такие регистры помечены как «синхронные». Расположение бит управления для таких регистров приведено в таблице «Синхронный регистр управления частотой периферийного блока».
- Асинхронный относительно процессорного ядра. В этом случае частота формируется независимо от процессорной частоты на основе одного их доступных внутренних или внешних источников частоты. Для передачи данных между ядром микроконтроллера и такими блоками реализованы соответствующие схемы пересинхронизации. В таблице регистров такие регистры помечены как «асинхронные». Расположение бит управления для таких регистров приведено в таблице «Асинхронный регистр управления частотой периферийного блока».

#### 20.8.29.1 PER0_CLK

Таблица 135 - Регистр PER0_CLK

| Base ADDR= | | | **0x4000_0000** | | | | Offset= | | **0x0000_000C** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | PER0_CLK | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31 | USB_CLK_EN | - | |
| 30 | MIL_CLK_EN | - | |
| 29 | UART3_CLK_EN | - | |
| 28 | UART2_CLK_EN | - | |
| 27 | UART1_CLK_EN | - | |
| 26 | UART0_CLK_EN | - | |
| 25 | CAN1_CLK_EN | - | |
| 24 | CAN0_CLK_EN | - | |
| 23 | SSP1_CLK_EN | - | |
| 22 | SSP0_CLK_EN | - | |
| 21 | SD_MMC_CLK_EN | - | |

| | | | |
|---|---|---|---|
| 20 | CRC_CLK_EN | - | |
| 19 | - | - | |
| 18 | - | - | |
| 17 | - | - | |
| 16 | PORTD_CLK_EN | - | |
| 15 | PORTC_CLK_EN | - | |
| 14 | PORTB_CLK_EN | - | |
| 13 | PORTA_CLK_EN | - | |
| 12 | - | - | |
| 11 | - | - | |
| 10 | - | - | |
| 9 | - | - | |
| 8 | - | - | |
| 7 | - | - | |
| 6 | - | - | |
| 5 | - | - | |
| 4 | - | - | |
| 3 | - | - | |
| 2 | - | - | |
| 1 | - | - | |
| 0 | - | - | |

#### 20.8.29.2 PER1_CLK

Таблица 136 – Регистр PER1_CLK

| Base ADDR= | | | **0x4000_0000** | | | | Offset= | | **0x0000_0010** | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | PER1_CLK | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31 | CORDIC_CLK_EN | - | - |
| 30 | I2C_CLK_EN | - | - |
| 29 | CMP3_CLK_EN | - | - |
| 28 | CMP2_CLK_EN | - | - |
| 27 | CMP1_CLK_EN | - | - |
| 26 | CMP0_CLK_EN | - | - |
| 25 | DAC3_CLK_EN | - | - |
| 24 | DAC2_CLK_EN | - | - |
| 23 | DAC1_CLK_EN | - | - |
| 22 | DAC0_CLK_EN | - | - |
| 21 | ADC2_CLK_EN | - | - |
| 20 | ADC1_CLK_EN | - | - |

| | | | |
|---|---|---|---|
| 19 | ADC0_CLK_EN | - | - |
| 18 | PWM8_CLK_EN | - | - |
| 17 | PWM7_CLK_EN | - | - |
| 16 | PWM6_CLK_EN | - | - |
| 15 | PWM5_CLK_EN | - | - |
| 14 | PWM4_CLK_EN | - | - |
| 13 | PWM3_CLK_EN | - | - |
| 12 | PWM2_CLK_EN | - | - |
| 11 | PWM1_CLK_EN | - | - |
| 10 | PWM0_CLK_EN | - | - |
| 9 | QEP1_CLK_EN | - | - |
| 8 | QEP0_CLK_EN | - | - |
| 7 | CAP3_CLK_EN | - | - |
| 6 | CAP2_CLK_EN | - | - |
| 5 | CAP1_CLK_EN | - | - |
| 4 | CAP0_CLK_EN | - | - |
| 3 | TIMER3_CLK_EN | - | - |
| 2 | TIMER2_CLK_EN | - | - |
| 1 | TIMER1_CLK_EN | - | - |
| 0 | TIMER0_CLK_EN | - | - |

### 20.8.30 *Синхронный регистр управления частотой периферийного блока*

Таблица 137 – Синхронный регистр управления частотой периферийного блока

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31..17 | - | - | Зарезервировано |
| 16 | EN_CLK | | Разрешение формирования тактовой частоты для блока<br>0 – нет частоты;<br>1 – разрешена выдача частоты |
| 15..0 | DIV[15:0] | | Делитель частоты MAX_CLK для формирования частоты<br>CLK = MAX_CLK/(DIV+1) |

### 20.8.31 *Асинхронный регистр управления частотой периферийного блока*

Таблица 138 – Асинхронный регистр управления частотой периферийного блока

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…28 | SELECT[3:0] | | Номер источника тактовой частоты для формирования частоты SELCLK<br>00000 – Генератор HSI<br>00001 – Генератор HSE0<br>00010 – Генератор HSE1<br>00011 – Генератор LSI<br>00100 – Генератор LSE<br>00101 – Генератор PLL0<br>00110 – Генератор PLL1<br>00111 – Генератор PLL2<br>01000 – Генератор PLL3<br>01101 – MAX_CLK<br>Переключение частоты необходимо выполнять с включенного источника |
| 27…17 | - | - | Зарезервировано |
| 16 | EN_CLK | | Разрешение формирования тактовой частоты<br>0 – нет частоты;<br>1 – разрешена выдача частоты |
| 15…0 | DIV[15:0] | | Делитель частоты SELCLK для формирования частоты<br>CLK = SELCLK/(DIV+1) |

## 20.9 Описание регистров контроллера обработки событий отказов, сбоев и ошибок FT_CNTR

Таблица 139 – Регистры контроллера обработки событий отказов, сбоев и ошибок

| Базовый адрес | Название | Описание |
|---|---|---|
| 0x4000_3000 | FT_CNTR | Блок обработки сбоев, ошибок и отказов |
| **Смещение** | | |
| 0x0000_0000 | KEY | Регистр ключа |
| 0x0000_0004 | CONTROL | Регистр управления блоком |
| 0x0000_0008 | STATUS | Регистр статуса блока |
| 0x0000_000C | TIMEOUT | Регистр времени отложенного сброса |
| 0x0000_0010 | TICKCNT | Общий счетчик HSI-генератора |
| 0x0000_0014 | FIRSTEVENT | Значение TICKCOUNT на момент возникновения первого разрешенного события сбоя |
| 0x0000_0018 | LASTEVENT | Значение TICKCOUNT на момент возникновения последнего разрешенного события сбоя |

| | | |
|---|---|---|
| 0x0000_001C | TIMEOUTCNT | Текущее значение счетчика отложенного сброса |
| 0x0000_0020 | EVENT0 | Флаги событий сбоев |
| 0x0000_0024 | EVENT1 | Флаги событий сбоев |
| 0x0000_0028 | EVENT2 | Флаги событий сбоев |
| 0x0000_002C | - | Зарезервировано |
| 0x0000_0030 | EVENT4 | Флаги событий сбоев |
| 0x0000_0034 | EVENT5 | Флаги событий сбоев |
| 0x0000_0038 | EVENT6 | Флаги событий сбоев |
| 0x0000_003C | EVENT7 | Флаги событий сбоев |
| 0x0000_0040 | - | Зарезервировано |
| 0x0000_0044 | EVENT9 | Флаги событий сбоев |
| 0x0000_0048 | EVENT10 | Флаги событий сбоев |
| 0x0000_004C | EVENT11 | Флаги событий сбоев |
| 0x0000_0050 | EVENT12 | Флаги событий сбоев |
| 0x0000_0054 | RESET_EVENT0 | Биты разрешения отложенного сброса от событий сбоев |
| 0x0000_0058 | RESET_EVENT1 | Биты разрешения отложенного сброса от событий сбоев |
| 0x0000_005C | RESET_EVENT2 | Биты разрешения отложенного сброса от событий сбоев |
| 0x0000_0060 | - | Зарезервировано |
| 0x0000_0064 | RESET_EVENT4 | Биты разрешения отложенного сброса от событий сбоев |
| 0x0000_0068 | IE_EVENT5 | Биты разрешения прерывания от событий сбоев |
| 0x0000_006C | IE_EVENT6 | Биты разрешения прерывания от событий сбоев |
| 0x0000_0070 | IE_EVENT7 | Биты разрешения прерывания от событий сбоев |
| 0x0000_0074 | - | Зарезервировано |
| 0x0000_0078 | IE_EVENT9 | Биты разрешения прерывания от событий сбоев |
| 0x0000_007C | IE_EVENT10 | Биты разрешения прерывания от событий сбоев |
| 0x0000_0080 | IE_EVENT11 | Биты разрешения прерывания от событий сбоев |
| 0x0000_0084 | IE_EVENT12 | Биты разрешения прерывания от событий сбоев |

### 20.9.1 *KEY*

| Base ADDR= | 0x4000_3000 | Offset= | 0x0000_0000 |
|---|---|---|---|
| REG Name: | KEY | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | KEY[31:0] | 0000_0000 | При записи в регистр значения 0x8555AAA1 открывается возможность записи в другие регистры блока FT_CNTR |

### 20.9.2 *CONTROL*

| Base ADDR= | 0x4000_3000 | Offset= | 0x0000_0004 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | RESET_TIMEOUT_CNT | TICK_OVER_IE | - |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…3 | - | | Зарезервировано |
| 2 | RESET_TIMEOUT_CNT | 0 | Бит сброса регистра TIMEOUTCNT в состояние ожидания<br>Запись 0 – ничего<br>Запись 1 – сброс, при условии отсутствия каких-либо флагов события, для которых разрешен вызов отложенного сброса |
| 1 | TICKOVER_IE | 0 | Бит разрешения формирования флага TICKOVER, который устанавливается при переполнении счетчика TICKCNT.<br>Счетчик TICKCOUNT переполняется раз в ~9 минут. |
| 0 | - | 1 | Зарезервировано |

### 20.9.3 *STATUS*

| Base ADDR= | 0x4000_3000 | Offset= | 0x0000_0008 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | FT_IF3 | FT_IF2 | FT_IF1 | FT_IF0 | | | TICKOVER |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…7 | - | | Зарезервировано |
| 6 | FT_IF3 | 0 | Флаг запроса прерывания от возникновения разрешенных событий с низкой важностью (EVENT9-EVENT12)<br>0 – нет запроса прерывания<br>1 – есть запрос прерывания |
| 5 | FT_IF2 | 0 | Флаг запроса прерывания от возникновения разрешенных событий со средней важностью (EVENT5-EVENT8)<br>0 – нет запроса прерывания<br>1 – есть запрос прерывания |
| 4 | FT_IF1 | 0 | Флаг запроса прерывания от возникновения любых событий с высокой важностью (EVENT0-EVENT4)<br>0 – нет запроса прерывания<br>1 – есть запрос прерывания |
| 3 | FT_IF0 | 0 | Флаг запроса прерывания от возникновения событий с высокой важностью, для которых разрешен отложенный сброс (EVENT0-EVENT4)<br>0 – нет запроса прерывания при запуске отложенного сброса<br>1 – есть запрос прерывания при запуске отложенного сброса |
| 2, 1 | – | | Зарезервировано |
| 0 | TICKOVER | 0 | Флаг переполнения счетчика TICKCNT<br>0 – нет переполнения<br>1 – было переполнение<br>Флаг активен при TICKOVER_IE = 1:<br>- устанавливается в момент переполнения счетчика TICKCNT;<br>- сбрасывается записью 1, если в момент сброса происходит очередное переполнение, флаг не сбрасывается.<br>При TICKOVER_IE = 0 флаг удерживается в 0 |

### 20.9.4 *TIMEOUT*

| Base ADDR= | | 0x4000_3000 | | | Offset= | | 0x0000_000C | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | TIMEOUT | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TIMEOUT[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…16 | - | | Зарезервировано |
| 15…0 | TIMEOUT[15:0] | 16'HFFFF | Число тактов генератора HSI от момента возникновения сбоя до генерации сигнала сброса.<br>При возникновении события сброса данное значение перезаписывается в регистр TIMEOUTCNT, который после этого начинает отсчет данного времени. |

### 20.9.5 *TICKCNT*

| BaseADDR= | | 0x4000_3000 | | | Offset= | | 0x0000_0010 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REGName: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| TICKCNT[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TICKCNT[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | TICKCNT[31:0] | | Счетчик тактов HSI-генератора с момента последнего сброса. При переполнении данного счетчика взводится бит TICKOVER. |

### 20.9.6 *FIRSTEVENT*

| BaseADDR= | | 0x4000_3000 | | | Offset= | | 0x0000_0014 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REGName: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| FIRSTEVENT[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FIRSTEVENT[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | FIRSTEVENT[31:0] | | Значение счетчика TICKCNT на момент возникновения первого события сбоя, для которого разрешен отложенный сброс (регистр RESET_EVENTx) или формирование прерывания (регистр IE_EVENTx). Сбрасывается только сигналом сброса |

### 20.9.7 *LASTEVENT*

| Base ADDR= | | 0x4000_3000 | | | Offset= | | 0x0000_0018 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| LASTEVENT[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LASTEVENT[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | LASTEVENT[31:0] | | Значение счетчика TICKCNT на момент возникновения последнего события сбоя, для которого разрешен отложенный сброс (регистр RESET_EVENTx) или формирование прерывания (регистр IE_EVENTx). Сбрасывается только сигналом сброса |

### 20.9.8 *TIMEOUTCNT*

| Base ADDR= | 0x4000_3000 | Offset= | 0x0000_001C |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RESERVED | | | | | | | | | | | | | | | TIMEOUTCNT[16] |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TIMEOUTCNT[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…17 | - | | Зарезервировано |
| 16…0 | TIMEOUTCNT | 17`H1FFFF | Значение счетчика времени отложенного сброса. |

В момент возникновения события, для которого разрешено формирование отложенного сброса, в счетчик TIMEOUTCNT переписывается значение регистра TIMEOUT. Счетчик начинает считать вниз до нуля и при достижении нулевого значения формируется сигнал сброса FT_RSTn.

Счетчик TIMEOUTCNT имеет 17 разрядов, при сбросе устанавливается в состояние всех единиц. При возникновении события устанавливается в состояние 16-ти битного регистра TIMEOUT, при этом 17-й разряд сбрасывается в 0. Это условие начала счета для данного счетчика. Счетчик выполнен по троированной логике. Для прекращения счета отложенного сброса необходимо:

- снять все флаги событий, разрешающих формирование сигнала запроса сброса FT_RSTn;
- через запись 1 в бит RESET_TIMEOUTCNT установить счетчик в исходное состояние.

Если в момент сброса счетчика TIMEOUTCNT сохраняются условия для его счета, то сброс счетчика не произойдет.

### 20.9.9 *EVENT0 (HIGH)*

| BaseADDR= | 0x4000_3000 | Offset= | 0x0000_0020 |
|---|---|---|---|
| REGName: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | AHB_SRAMC_COMP_ERR_1 | SRAMC_COMP_REG_ERR_1 |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AHB_SRAMC_COMP_ERR_0 | SRAMC_COMP_REG_ERR_0 | AHB_SRAMD_COMP_ERR_1 | SRAMD_COMP_REG_ERR_1 | AHB_SRAMD_COMP_ERR_0 | SRAMD_COMP_REG_ERR_0 | AHB_FLASH_COMP_ERR_1 | FLASH_COMP_REG_ERR_1 | AHB_FLASH_COMP_ERR_0 | FLASH_COMP_REG_ERR_0 | AHB_C_COMP_ERR | AHB_M_COMP_ERR | AHB_S_COMP_ERR | AHB_D_COMP_ERR | AHB_I_COMP_ERR | COMP_REG_ERR |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…18 | Зарезервировано | | Зарезервировано |
| 17 | AHB_SRAMC_COMP_ERR_1 | 0 | Флаг ошибки сравнения сигналов шины AHBLOCKSTEP контроллера SRAMC_1<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 16 | SRAMC_COMP_REG_ERR_1 | 0 | Флаг ошибки сравнения сигналов LOCKSTEP контроллера SRAMC_1<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается. |
| 15 | AHB_SRAMC_COMP_ERR_0 | 0 | Флаг ошибки сравнения сигналов шины AHBLOCKSTEP контроллера SRAMC_0<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |

| | | | |
|---|---|---|---|
| 14 | SRAMC_COMP_REG_ERR_0 | 0 | Флаг ошибки сравнения сигналов LOCKSTEP контроллера SRAMC_0<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 13 | AHB_SRAMD_COMP_ERR_1 | 0 | Флаг ошибки сравнения сигналов шины AHBLOCKSTEP контроллера SRAMD1<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается. |
| 12 | SRAMD_COMP_REG_ERR_1 | 0 | Флаг ошибки сравнения сигналов LOCKSTEP контроллера SRAMD1<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 11 | AHB_SRAMD_COMP_ERR_0 | 0 | Флаг ошибки сравнения сигналов шины AHBLOCKSTEP контроллера SRAMD0<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 10 | SRAMD_COMP_REG_ERR_0 | 0 | Флаг ошибки сравнения сигналов LOCKSTEP контроллера SRAMD0<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 9 | AHB_FLASH_COMP_ERR_1 | 0 | Флаг ошибки сравнения сигналов шины AHBLOCKSTEP контроллера FLASH 1<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 8 | FLASH_COMP_REG_ERR_1 | 0 | Флаг ошибки сравнения сигналов LOCKSTEP контроллера FLASH 1<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 7 | AHB_FLASH_COMP_ERR_0 | 0 | Флаг ошибки сравнения сигналов шины AHBLOCKSTEP контроллера FLASH 0<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |

| | | | |
|---|---|---|---|
| 6 | FLASH_COMP_REG_ERR_0 | 0 | Флаг ошибки сравнения сигналов LOCKSTEP контроллера FLASH 0<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 5 | AHB_C_COMP_ERR | 0 | Флаг ошибки сравнения сигналов системной шины SCRUBBER процессора CPU 0 и процессора CPU 1<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 4 | AHB_M_COMP_ERR | 0 | Флаг ошибки сравнения сигналов системной шины DMA процессорного ядра CPU 0 и процессорного ядра CPU 1<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 3 | AHB_S_COMP_ERR | 0 | Флаг ошибки сравнения сигналов системной шины S процессорного ядра CPU 0 и процессорного ядра CPU1<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 2 | AHB_D_COMP_ERR | 0 | Флаг ошибки сравнения сигналов системной шины D процессорного ядра CPU 0 и процессорного ядра CPU 1<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 1 | AHB_I_COMP_ERR | 0 | Флаг ошибки сравнения сигналов системной шины I процессорного ядра CPU 0 и процессорного ядра CPU 1<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 0 | COMP_REG_ERR | 0 | Ошибка сравнения вспомогательных сигналов процессорного ядра CPU 0 и процессорного ядра CPU 1<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |

### 20.9.10 *EVENT1 (HIGH)*

| Base ADDR= | 0x4000_3000 | Offset= | 0x0000_0024 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | ECC_EXTBUS_DE_TE | ECC_EXTBUS_SE | - | ECCERR0_RAMC1 | - | ECCERR0_RAMC0 | - | ECCERR0_RAMD1 | - | ECCERR0_RAMD0 |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | - | ECCERR1_FLASH0 | ECCERR0_FLASH0 | - | ECCERR_ROM | ECCERRC_1 | ECCERRM_1 | ECCERRS_1 | ECCERRD_1 | ECCERRI_1 | ECCERRC_0 | ECCERRM_0 | ECCERRS_0 | ECCERRD_0 | ECCERRI_0 |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…26 | - | | Зарезервировано |
| 25 | ECC_EXTBUS_DE_TE[5] | 0 | Двойная или тройная ошибка ECC на шине AHB на EXT_BUS<br>0 – нет ошибки<br>1 – есть ошибка |
| 24 | ECC_EXTBUS_SE[5] | 0 | Одиночная ошибка ECC на шине AHB на EXT_BUS<br>0 – нет ошибки<br>1 – есть ошибка |
| 23 | - | - | Зарезервировано |
| 22 | ECCERR0_RAMC1[5] | 0 | Ошибка ECC0 на шине AHB на RAMC1<br>0 – нет ошибки<br>1 – есть ошибка |
| 21 | - | - | Зарезервировано |
| 20 | ECCERR0_RAMC0[5] | 0 | Ошибка ECC0 на шине AHB на RAMC0<br>0 – нет ошибки<br>1 – есть ошибка |
| 19 | - | - | Зарезервировано |
| 18 | ECCERR0_RAMD1[5] | 0 | Ошибка ECC0 на шине AHB на RAMD1<br>0 – нет ошибки<br>1 – есть ошибка |
| 17 | - | - | Зарезервировано |
| 16 | ECCERR0_RAMD0[5] | 0 | Ошибка ECC0 на шине AHB на RAMD0<br>0 – нет ошибки<br>1 – есть ошибка |

[5] Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается.

| | | | |
|---|---|---|---|
| 15 | - | - | Зарезервировано |
| 14 | - | - | Зарезервировано |
| 13 | ECCERR1_FLASH0[5] | 0 | Ошибка ECC 1 на шине AHB на FLASH0<br>0 – нет ошибки<br>1 – есть ошибка |
| 12 | ECCERR0_FLASH0[5] | 0 | Ошибка ECC 0 на шине AHB на FLASH0<br>0 – нет ошибки<br>1 – есть ошибка. |
| 11 | - | - | Зарезервировано |
| 10 | ECCERR_ROM[5] | 0 | Ошибка ECC на шине AHB на ROM<br>0 – нет ошибки<br>1 – есть ошибка |
| 9 | ECCERRC_1[5] | 0 | Ошибка ECC на шине C на SCRUBBER1<br>0 – нет ошибки<br>1 – есть ошибка |
| 8 | ECCERRM_1[5] | 0 | Ошибка ECC на шине M на DMA1<br>0 – нет ошибки<br>1 – есть ошибка |
| 7 | ECCERRS_1[5] | 0 | Ошибка ECC на шине S на CPU1<br>0 – нет ошибки<br>1 – есть ошибка |
| 6 | ECCERRD_1[5] | 0 | Ошибка ECC на шине D на CPU1<br>0 – нет ошибки<br>1 – есть ошибка |
| 5 | ECCERRI_1[5] | 0 | Ошибка ECC на шине I на CPU1<br>0 – нет ошибки<br>1 – есть ошибка |
| 4 | ECCERRC_0[5] | 0 | Ошибка ECC на шине C на SCRUBBER0<br>0 – нет ошибки<br>1 – есть ошибка |
| 3 | ECCERRM_0[5] | 0 | Ошибка ECC на шине M на DMA0<br>0 – нет ошибки<br>1 – есть ошибка |
| 2 | ECCERRS_0[5] | 0 | Ошибка ECC на шине S на CPU0<br>0 – нет ошибки<br>1 – есть ошибка |
| 1 | ECCERRD_0[5] | 0 | Ошибка ECC на шине D на CPU0<br>0 – нет ошибки<br>1 – есть ошибка |
| 0 | ECCERRI_0[5] | 0 | Ошибка ECC на шине I на CPU0<br>0 – нет ошибки<br>1 – есть ошибка |

### 20.9.11 *EVENT2 (HIGH)*

| Base ADDR= | 0x4000_3000 | Offset= | 0x0000_0028 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | ECC_CACHE_D_1_DE | ECC_CACHE_D_0_DE | ECC_CACHE_I_1_DE | ECC_CACHE_I_0_DE | ECCERR_EXTBUS | ECC_RAMC1_DE | ECC_RAMC0_DE | ECC_RAMD1_DE | ECC_RAMD0_DE | - | ECC_FLASH0_DE | ECC_ROM_DE |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…9 | - | | Зарезервировано |
| 13 | Зарезервировано | 0 | Зарезервировано |
| 12 | Зарезервировано | 0 | Зарезервировано |
| 11 | ECC_CACHE_D_1_DE | 0 | Двойная неисправимая ошибка ECC в массиве памяти CACHE_D в CPU1<br>0 – нет ошибки<br>1 – есть ошибка |
| 10 | ECC_CACHE_D_0_DE | 0 | Двойная неисправимая ошибка ECC в массиве памяти CACHE_D в CPU0<br>0 – нет ошибки<br>1 – есть ошибка |
| 9 | ECC_CACHE_I_1_DE | 0 | Двойная неисправимая ошибка ECC в массиве памяти CACHE_I в CPU1<br>0 – нет ошибки<br>1 – есть ошибка |
| 8 | ECC_CACHE_I_0_DE | 0 | Двойная неисправимая ошибка ECC в массиве памяти CACHE_I в CPU0<br>0 – нет ошибки<br>1 – есть ошибка |
| 7 | ECCERR_EXTBUS | 0 | Одиночная, двойная или тройная неисправимая ошибка ECC в массиве памяти EXT_BUS<br>0 – нет ошибки<br>1 – есть ошибка |
| 6 | ECC_RAMC1_DE | 0 | Двойная неисправимая ошибка ECC в массиве памяти RAMC1<br>0 – нет ошибки<br>1 – есть ошибка |
| 5 | ECC_RAMC0_DE | 0 | Двойная неисправимая ошибка ECC в массиве памяти RAMC0<br>0 – нет ошибки<br>1 – есть ошибка |

| | | | |
|---|---|---|---|
| 4 | ECC_RAMD1_DE | 0 | Двойная неисправимая ошибка ECC в массиве памяти RAMD1<br>0 – нет ошибки<br>1 – есть ошибка |
| 3 | ECC_RAMD0_DE | 0 | Двойная неисправимая ошибка ECC в массиве памяти RAMD0<br>0 – нет ошибки<br>1 – есть ошибка |
| 2 | - | - | Зарезервировано |
| 1 | ECC_FLASH0_DE | 0 | Двойная неисправимая ошибка ECC в массиве памяти FLASH0<br>0 – нет ошибки<br>1 – есть ошибка |
| 0 | ECC_ROM_DE | 0 | Двойная неисправимая ошибка ECC в массиве памяти ROM<br>0 – нет ошибки<br>1 – есть ошибка |

### 20.9.12 *EVENT4 (HIGH)*

| Base ADDR= | 0x4000_3000 | Offset= | 0x0000_0030 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | ROM_REG_DE | RAMC1_REG_DE | RAMC0_REG_DE | RAMD1_REG_DE | RAMD0_REG_DE | - | FLASH_REG_DE | FT_REG_DE | FT_CNT_ERR | BKP_MEM_DE | BKP_ERR | RTC_ERR |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…9 | - | | Зарезервировано |
| 11 | ROM_REG_DE[6] | | Двойная неисправимая ошибка в регистрах управления ROM_CNTR<br>0 – нет ошибки<br>1 – есть ошибка |
| 10 | RAMC1_REG_DE[6] | | Двойная неисправимая ошибка в регистрах управления RAMC1_CNTR<br>0 – нет ошибки<br>1 – есть ошибка |
| 9 | RAMC0_REG_DE[6] | | Двойная неисправимая ошибка в регистрах управления RAMC0_CNTR<br>0 – нет ошибки<br>1 – есть ошибка |
| 8 | RAMD1_REG_DE[6] | | Двойная неисправимая ошибка в регистрах управления RAMD1_CNTR<br>0 – нет ошибки<br>1 – есть ошибка |

[6] Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается.

| | | | |
|---|---|---|---|
| 7 | RAMD0_REG_DE[6] | | Двойная неисправимая ошибка в регистрах управления RAMD0_CNTR<br>0 – нет ошибки<br>1 – есть ошибка |
| 6 | - | | Зарезервировано |
| 5 | FLASH_REG_DE[6] | 0 | Двойная неисправимая ошибка в регистрах управления FLASH_CNTR<br>0 – нет ошибки<br>1 – есть ошибка |
| 4 | FT_REG_DE[6] | 0 | Двойная неисправимая ошибка в регистре управления FT_CNTR->CONTROL<br>0 – нет ошибки<br>1 – есть ошибка |
| 3 | FT_CNT_ERR[6] | 0 | Ошибка в счетчике TIMEOUTCNT отложенного сброса FT_CNTR<br>0 – нет ошибки<br>1 – есть ошибка |
| 2 | BKP_MEM_DE[6] | 0 | Двойная ошибка в массиве памяти BKP (00-59)<br>0 – нет ошибки<br>1 – есть ошибка |
| 1 | BKP_ERR[6] | 0 | Ошибка в часах конфигурационных регистрах BKP (60-63, BLDO_CNTRx)<br>0 – нет ошибки<br>1 – есть ошибка |
| 0 | RTC_ERR[6] | 0 | Ошибка в часах реального времени RTC<br>0 – нет ошибки<br>1 – есть ошибка |

### 20.9.13 *EVENT5 (MIDDLE)*

| Base ADDR= | | 0x4000_3000 | | | Offset= | | 0x0000_0034 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | ADC_CLK_DE | PWM_CLK_DE | QEP_CLK_DE | CAP_CLK_DE | TIMER_CLK_DE | MIL_CLK_DE | UART_CLK_DE | CAN_CLK_DE | SSP_CLK_DE | RTC_CLK_DE | USBMAC_CLK_DE |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| USB_CLK_DE | EMAC_CLK_DE | PER_CLK_DE | CPU_CLK_DE | PLL3_CLK_DE | PLL2_CLK_DE | PLL1_CLK_DE | PLL0_CLK_DE | HSE1_CLK_DE | HSE0_CLK_DE | LSE_CLK_DE | LSI_CLK_DE | KEY_DE | MAX_CLK_DE | - | - |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…27 | - | | Зарезервировано |

| | | | |
|---|---|---|---|
| 26 | ADC_CLK_DE | | Двойная неисправимая ошибка в регистрах ADC0_CLK – ADC2_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 25 | PWM_CLK_DE | | Двойная неисправимая ошибка в регистрах PWM0_CLK – PWM8_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 24 | QEP_CLK_DE | | Двойная неисправимая ошибка в регистрах QEP0_CLK – QEP1_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 23 | CAP_CLK_DE | | Двойная неисправимая ошибка в регистрах CAP0_CLK – CAP3_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 22 | TIMER_CLK_DE | | Двойная неисправимая ошибка в регистрах TIMER0_CLK – TIMER3_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 21 | MIL_CLK_DE | | Двойная неисправимая ошибка в регистре MIL_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 20 | UART_CLK_DE | | Двойная неисправимая ошибка в регистрах UART0_CLK – UART3_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 19 | CAN_CLK_DE | | Двойная неисправимая ошибка в регистрах CAN0_CLK, CAN1_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 18 | SSP_CLK_DE | | Двойная неисправимая ошибка в регистрах SSP0_CLK, SSP1_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 17 | RTC_CLK_DE | | Двойная неисправимая ошибка в регистре RTC_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |

| | | | |
|---|---|---|---|
| 16 | USBMAC_CLK_DE | | Двойная неисправимая ошибка в регистре USBMAC_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 15 | USB_CLK_DE | | Двойная неисправимая ошибка в регистре USB_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 14 | EMAC_CLK_DE | | Двойная неисправимая ошибка в регистре EMAC_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 13 | PER_CLK_DE | | Двойная неисправимая ошибка в регистрах PER0_CLK, PER1_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 12 | CPU_CLK_DE | | Двойная неисправимая ошибка в регистрах CPU_CLK, CPU_CHK0, CPU_CHK1, CPU_CHK2 блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 11 | PLL3_CLK_DE | | Двойная неисправимая ошибка в регистрах PLL3_CLK, PLL3_CHK0, PLL3_CHK1, PLL3_CHK2 блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 10 | PLL2_CLK_DE | | Двойная неисправимая ошибка в регистрах PLL2_CLK, PLL2_CHK0, PLL2_CHK1, PLL2_CHK2 блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 9 | PLL1_CLK_DE | | Двойная неисправимая ошибка в регистрах PLL1_CLK, PLL1_CHK0, PLL1_CHK1, PLL1_CHK2 блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 8 | PLL0_CLK_DE | | Двойная неисправимая ошибка в регистрах PLL0_CLK, PLL0_CHK0, PLL0_CHK1, PLL0_CHK2 блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 7 | HSE1_CLK_DE | | Двойная неисправимая ошибка в регистрах HSE1_CLK, HSE1_CHK0, HSE1_CHK1, HSE1_CHK2 блока CLK_CNTR |

| | | | |
|---|---|---|---|
| | | | 0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 6 | HSE0_CLK_DE | | Двойная неисправимая ошибка в регистрах HSE0_CLK, HSE0_CHK0, HSE0_CHK1, HSE0_CHK2 блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 5 | LSE_CLK_DE | | Двойная неисправимая ошибка в регистрах LSE_CLK, LSE_CHK0, LSE_CHK1, LSE_CHK2 блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 4 | LSI_CLK_DE | | Двойная неисправимая ошибка в регистрах LSI_CLK, LSI_CHK0, LSI_CHK1, LSI_CHK2 блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 3 | KEY_DE | | Двойная неисправимая ошибка в регистре KEY блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 2 | MAX_CLK_DE | | Двойная неисправимая ошибка в регистре MAX_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 1…0 | - | | Зарезервировано |

### 20.9.14 *EVENT6 (MIDDLE)*

| Base ADDR= | | 0x4000_3000 | | | Offset= | | 0x0000_0038 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | PLL3_CLOCK_ERROR | PLL2_CLOCK_ERROR | PLL1_CLOCK_ERROR | PLL0_CLOCK_ERROR | LSE_CLOCK_ERROR | LSI_CLOCK_ERROR | HSE1_CLOCK_ERROR | HSE0_CLOCK_ERROR | CPU_CLOCK_ERROR |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|

| | | | |
|---|---|---|---|
| 31…9 | - | | Зарезервировано |
| 8 | PLL3_CLOCK_ERROR | | Ошибка при формировании частоты PLL3_CLK в блоке CLK_CHECKER в блоке CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 7 | PLL2_CLOCK_ERROR | | Ошибка при формировании частоты PLL2_CLK в блоке CLK_CHECKER в блоке CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 6 | PLL1_CLOCK_ERROR | | Ошибка при формировании частоты PLL1_CLK в блоке CLK_CHECKER в блоке CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка в мо<br>Флаг сбрасывается записью 1, если мент сброса происходит очередное событие, флаг не сбрасывается |
| 5 | PLL0_CLOCK_ERROR | | Ошибка при формировании частоты PLL0_CLK в блоке CLK_CHECKER в блоке CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 4 | LSE_CLOCK_ERROR | | Ошибка при формировании частоты LSE_CLK в блоке CLK_CHECKER в блоке CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 3 | LSI_CLOCK_ERROR | | Ошибка при формировании частоты LSI_CLK в блоке CLK_CHECKER в блоке CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 2 | HSE1_CLOCK_ERROR | | Ошибка при формировании частоты HSE1_CLK в блоке CLK_CHECKER в блоке CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 1 | HSE0_CLOCK_ERROR | | Ошибка при формировании частоты HSE0_CLK в блоке CLK_CHECKER в блоке CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 0 | CPU_CLOCK_ERROR | | Ошибка при формировании частоты CPU_CLK в блоке CLK_CHECKER в блоке CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |

### 20.9.15 *EVENT7 (MIDDLE)*

| Base ADDR= | | 0x4000_3000 | | | Offset= | | 0x0000_003C | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |

| | | | | | | | | | | | | ROM_REG_SE | RAMD1_REG_SE | RAMD0_REG_SE | RAMC1_REG_SE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RAMC0_REG_SE | - | FLASH_REG_SE | FT_REG_SE | BKP_MEM_SE | ECC_CACHE_D_1_SE | ECC_CACHE_D_0_SE | ECC_CACHE_I_1_SE | ECC_CACHE_I_0_SE | ROM_SE | SRAMC1_SE | SRAMC0_SE | SRAMD1_SE | SRAMD0_SE | - | FLASH0_SE |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…17 | - | | Зарезервировано |
| 19 | ROM_REG_SE | | Одиночная ошибка в регистрах управления блока ROM<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 18 | RAMD1_REG_SE | | Одиночная ошибка в регистрах управления блока RAMD1<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 17 | RAMD0_REG_SE | | Одиночная ошибка в регистрах управления блока RAMD0<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 16 | RAMC1_REG_SE | | Одиночная ошибка в регистрах управления блока RAMC1<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 15 | RAMC0_REG_SE | | Одиночная ошибка в регистрах управления блока RAMC0<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 14 | - | | Зарезервировано |
| 13 | FLASH_REG_SE | 0 | Одиночная ошибка в регистрах управления блока FLASH_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |

| | | | |
|---|---|---|---|
| 12 | FT_REG_SE | 0 | Одиночная ошибка в регистре управления блока FT_CNTR->CONTROL<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 11 | BKP_MEM_SE | 0 | Одиночная ошибка в массиве памяти BKP (00-59)<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 10 | ECC_CACHE_D_1_SE | 0 | Одиночная ошибка в массиве памяти CACHED в CPU1<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 9 | ECC_CACHE_D_0_SE | 0 | Одиночная ошибка в массиве памяти CACHED в CPU0<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 8 | ECC_CACHE_I_1_SE | 0 | Одиночная ошибка в массиве памяти CACHEI в CPU1<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 7 | ECC_CACHE_I_0_SE | 0 | Одиночная ошибка в массиве памяти CACHEI в CPU0<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 6 | ROM_SE | 0 | Одиночная ошибка в массиве памяти ROM<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается. |
| 5 | SRAMC1_SE | 0 | Одиночная ошибка в массиве памяти RAMC1<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 4 | SRAMC0_SE | 0 | Одиночная ошибка в массиве памяти RAMC0<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 3 | SRAMD1_SE | 0 | Одиночная ошибка в массиве памяти RAMD1<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 2 | SRAMD0_SE | 0 | Одиночная ошибка в массиве памяти RAMD0<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 1 | - | | Зарезервировано |
| 0 | FLASH0_SE | 0 | Одиночная ошибка в массиве памяти FLASH0<br>0 – нет ошибки<br>1 – есть ошибка |

| | | | |
|---|---|---|---|
| | | | Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |

### 20.9.16 *EVENT9 (LOW)*

| Base ADDR= | 0x4000_3000 | | | | Offset= | | 0x0000_0044 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | ADC_CLK_SE | PWM_CLK_SE | QEP_CLK_SE | CAP_CLK_SE | TIMER_CLK_SE | MIL_CLK_SE | UART_CLK_SE | CAN_CLK_SE | SSP_CLK_DE | RTC_CLK_SE | USBMAC_CLK_SE |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| USB_CLK_SE | EMAC_CLK_SE | PER_CLK_SE | CPU_CLK_SE | PLL3_CLK_SE | PLL2_CLK_SE | PLL1_CLK_SE | PLL0_CLK_SE | HSE1_CLK_SE | HSE0_CLK_SE | LSE_CLK_SE | LSI_CLK_SE | KEY_SE | MAX_CLK_SE | - | - |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…27 | - | | Зарезервировано |
| 26 | ADC_CLK_SE | | Одиночная неисправимая ошибка в регистрах ADC0_CLK – ADC2_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 25 | PWM_CLK_SE | | Одиночная неисправимая ошибка в регистрах PWM0_CLK – PWM8_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 24 | QEP_CLK_SE | | Одиночная неисправимая ошибка в регистрах QEP0_CLK – QEP1_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 23 | CAP_CLK_SE | | Одиночная неисправимая ошибка в регистрах CAP0_CLK – CAP3_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 22 | TIMER_CLK_SE | | Одиночная неисправимая ошибка в регистрах TIMER0_CLK – TIMER3_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 21 | MIL_CLK_SE | | Одиночная неисправимая ошибка в регистре MIL_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |

| | | | |
|---|---|---|---|
| 20 | UART_CLK_SE | | Одиночная неисправимая ошибка в регистрах UART0_CLK – UART3_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 19 | CAN_CLK_SE | | Одиночная неисправимая ошибка в регистрах CAN0_CLK, CAN1_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 18 | SSP_CLK_DE | | Одиночная неисправимая ошибка в регистрах SSP0_CLK, SSP1_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 17 | RTC_CLK_SE | | Одиночная неисправимая ошибка в регистре RTC_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 16 | USBMAC_CLK_SE | | Одиночная неисправимая ошибка в регистре USBMAC_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 15 | USB_CLK_SE | | Одиночная неисправимая ошибка в регистре USB_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 14 | EMAC_CLK_SE | | Одиночная неисправимая ошибка в регистре EMAC_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 13 | PER_CLK_SE | | Одиночная неисправимая ошибка в регистрах PER0_CLK, PER1_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 12 | CPU_CLK_SE | | Одиночная неисправимая ошибка в регистрах CPU_CLK, CPU_CHK0, CPU_CHK1, CPU_CHK2 блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 11 | PLL3_CLK_SE | | Одиночная неисправимая ошибка в регистрах PLL3_CLK, PLL3_CHK0, PLL3_CHK1, PLL3_CHK2 блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |

| | | | |
|---|---|---|---|
| 10 | PLL2_CLK_SE | | Одиночная неисправимая ошибка в регистрах PLL2_CLK, PLL2_CHK0, PLL2_CHK1, PLL2_CHK2 блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 9 | PLL1_CLK_SE | | Одиночная неисправимая ошибка в регистрах PLL1_CLK, PLL1_CHK0, PLL1_CHK1, PLL1_CHK2 блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 8 | PLL0_CLK_SE | | Одиночная неисправимая ошибка в регистрах PLL0_CLK, PLL0_CHK0, PLL0_CHK1, PLL0_CHK2 блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 7 | HSE1_CLK_SE | | Одиночная неисправимая ошибка в регистрах HSE1_CLK, HSE1_CHK0, HSE1_CHK1, HSE1_CHK2 блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 6 | HSE0_CLK_SE | | Одиночная неисправимая ошибка в регистрах HSE0_CLK, HSE0_CHK0, HSE0_CHK1, HSE0_CHK2 блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 5 | LSE_CLK_SE | | Одиночная неисправимая ошибка в регистрах LSE_CLK, LSE_CHK0, LSE_CHK1, LSE_CHK2 блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 4 | LSI_CLK_SE | | Одиночная неисправимая ошибка в регистрах LSI_CLK, LSI_CHK0, LSI_CHK1, LSI_CHK2 блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 3 | KEY_SE | | Одиночная неисправимая ошибка в регистре KEY блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 2 | MAX_CLK_SE | | Одиночная неисправимая ошибка в регистре MAX_CLK блока CLK_CNTR<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 1…0 | - | | Зарезервировано |

### 20.9.17 *EVENT10 (LOW)*

| Base ADDR= | 0x4000_3000 | | | | Offset= | | 0x0000_0048 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | PORTD_CURLIM | PORTC_CURLIM | PORTB_CURLIM | PORTA_CURLIM |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…4 | - | | Зарезервировано |
| 3 | PORTD_CURLIM | | Ошибка превышения выходного тока в PORTD<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 2 | PORTC_CURLIM | | Ошибка превышения выходного тока в PORTC<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 1 | PORTB_CURLIM | | Ошибка превышения выходного тока в PORTB<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 0 | PORTA_CURLIM | | Ошибка превышения выходного тока в PORTA<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |

### 20.9.18 *EVENT11 (LOW)*

| Base ADDR= | 0x4000_3000 | Offset= | 0x0000_004C |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EMAC_DE | | | | | | | | MIL_DE | | | | | CAN1_DE | CAN0_DE |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…15 | - | | Зарезервировано |
| 14 | EMAC_DE | 0 | Двойная неисправимая ошибка в массивах памяти ETHERNETMAC<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 13..7 | - | | Зарезервировано |
| 6 | MIL_DE | 0 | Двойная неисправимая ошибка в массивах памяти MIL<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 5..2 | - | | Зарезервировано |
| 1 | CAN1_DE | 0 | Двойная неисправимая ошибка в массивах памяти CAN1<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 0 | CAN0_DE | 0 | Двойная неисправимая ошибка в массивах памяти CAN0<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |

### 20.9.19 *EVENT12 (LOW)*

| Base ADDR= | 0x4000_3000 | Offset= | 0x0000_0050 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EMAC_SE | | | | | | | | MIL_SE | | | | | CAN1_SE | CAN0_SE |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…15 | - | | Зарезервировано |
| 14 | EMAC_SE | 0 | Одиночная неисправимая ошибка в массивах памяти ETHERNETMAC<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 13..7 | - | | Зарезервировано |
| 6 | MIL_SE | 0 | Одиночная неисправимая ошибка в массивах памяти MIL<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 5..2 | - | | Зарезервировано |
| 1 | CAN1_SE | 0 | Одиночная неисправимая ошибка в массивах памяти CAN1<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |
| 0 | CAN0_SE | 0 | Одиночная неисправимая ошибка в массивах памяти CAN0<br>0 – нет ошибки<br>1 – есть ошибка<br>Флаг сбрасывается записью 1, если в момент сброса происходит очередное событие, флаг не сбрасывается |

### 20.9.20 *RESET_EVENT0*

| Base ADDR= | | 0x4000_3000 | | | Offset= | | 0x0000_0054 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…18 | - | | Зарезервировано |
| 17…0 | RESET_EN[17:0] | 18'h0000 | Биты разрешения запуска отложенного сброса при возникновении событий в EVENT0<br>0 – сброс не формируется<br>1 – сброс формируется |

### 20.9.21 *RESET_EVENT1*

| Base ADDR= | | 0x4000_3000 | | | Offset= | | 0x0000_0058 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…20 | - | | Зарезервировано |
| 25…0 | RESET_EN[25:0] | 26'h0000 | Биты разрешения запуска отложенного сброса при возникновении событий в EVENT1<br>0 – сброс не формируется<br>1 – сброс формируется<br>Задействованы номера битов, используемых в регистре EVENT1 |

### 20.9.22 *RESET_EVENT2*

| Base ADDR= | 0x4000_3000 | | | | Offset= | | 0x0000_005C | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…12 | - | | Зарезервировано |
| 11…0 | RESET_EN[11:0] | 12’h0000 | Биты разрешения запуска отложенного сброса при возникновении событий в EVENT2<br>0 – сброс не формируется<br>1 – сброс формируется<br>Задействованы номера битов, используемых в регистре EVENT2 |

### 20.9.23 *RESET_EVENT4*

| Base ADDR= | 0x4000_3000 | | | | Offset= | | 0x0000_0064 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…12 | - | | Зарезервировано |
| 11…0 | RESET_EN[11:0] | 12’h000 | Биты разрешения запуска отложенного сброса при возникновении событий в EVENT4<br>0 – сброс не формируется<br>1 – сброс формируется<br>Задействованы номера битов, используемых в регистре EVENT4 |

### 20.9.24 *IE_EVENT5*

| Base ADDR= | | 0x4000_3000 | | | Offset= | | 0x0000_0068 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…27 | - | | Зарезервировано |
| 26…2 | INT_EN[26:2] | 25'h0000 | Биты разрешения запроса прерывания FT_IF2 по событиям в EVENT5<br>0 – запрос прерывания не формируется<br>1 – запрос прерывания формируется |
| 1…0 | - | | Зарезервировано |

### 20.9.25 *IE_EVENT6*

| Base ADDR= | | 0x4000_3000 | | | Offset= | | 0x0000_006C | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…8 | - | | Зарезервировано |
| 8…0 | INT_EN[8:0] | 9'h0000 | Биты разрешения запроса прерывания FT_IF2 по событиям в EVENT6<br>0 – запрос прерывания не формируется<br>1 – запрос прерывания формируется |

### 20.9.26 *IE_EVENT7*

| Base ADDR= | 0x4000_3000 | | | | Offset= | | 0x0000_0070 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…20 | - | | Зарезервировано |
| 19…0 | INT_EN[19:0] | 20'h0000 | Биты разрешения запроса прерывания FT_IF2 по событиям в EVENT7<br>0 – запрос прерывания не формируется<br>1 – запрос прерывания формируется<br>Задействованы номера битов, используемых в регистре EVENT7 |

### 20.9.27 *IE_EVENT9*

| Base ADDR= | 0x4000_3000 | | | | Offset= | | 0x0000_0078 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…27 | - | | Зарезервировано |
| 26…2 | INT_EN[26:2] | 25'h0000 | Биты разрешения запроса прерывания FT_IF3 по событиям в EVENT9<br>0 – запрос прерывания не формируется<br>1 – запрос прерывания формируется |
| 1…0 | - | | Зарезервировано |

### 20.9.28 *IE_EVENT10*

| Base ADDR= | | 0x4000_3000 | | | Offset= | | 0x0000_007C | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…4 | - | | Зарезервировано |
| 3…0 | INT_EN[3:0] | 4'h0000 | Биты разрешения запроса прерывания FT_IF3 по событиям в EVENT10<br>0 – запрос прерывания не формируется<br>1 – запрос прерывания формируется |

### 20.9.29 *IE_EVENT11*

| Base ADDR= | | 0x4000_3000 | | | Offset= | | 0x0000_0080 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…15 | - | | Зарезервировано |
| 14…0 | INT_EN[14:0] | 15'h0000 | Биты разрешения запроса прерывания FT_IF3 по событиям в EVENT11<br>0 – запрос прерывания не формируется<br>1 – запрос прерывания формируется<br>Задействованы номера битов, используемых в регистре EVENT11 |

### 20.9.30 *IE_EVENT12*

| Base ADDR= | 0x4000_3000 | Offset= | 0x0000_0084 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…15 | - | | Зарезервировано |
| 14…0 | INT_EN[14:0] | 15'h0000 | Биты разрешения запроса прерывания FT_IF3 по событиям в EVENT12<br>0 – запрос прерывания не формируется<br>1 – запрос прерывания формируется<br>Задействованы номера битов, используемых в регистре EVENT12 |

## 20.10 Описание регистров контроллера ПЗУ ROM_CNTR

Контроллер ПЗУ не имеет каких-либо функций по управлению блоком ПЗУ. Данный блок используются всегда, и контроллер предоставляет центральному процессору только информацию об обнаруженных ошибках при работе с ПЗУ.

Таблица 140 – Регистры контроллера ПЗУ

| Базовый адрес | Смещение | Название | Состояние после сброса | Описание |
|---|---|---|---|---|
| 0x40005000 | 0x0000_0000 | KEY | | Регистр ключа, разрешающего модификацию остальных регистров |
| | 0x0000_0004 | ECCCS0 | | Регистр статуса ошибок ECCROM |
| | 0x0000_0008 | ECCADR | | Регистр адреса последней ошибки |
| | 0x0000_000C | ECCDATA | | Регистр данных последней ошибки |
| | 0x0000_0010 | ECCECC | | Регистр данных ECC |

### 20.10.1 *KEY*

| Base ADDR= | 0x40005000 | Offset= | 0x0000_0000 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| REG Name: | KEY | | | | | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | KEY[31:0] | 0000_0000 | При записи в регистр значения 0x8555AAA1 открывается возможность записи в другие регистры блока ROM_CNTR |

### 20.10.2 *ECCCS*

| Base ADDR= | 0x40005000 | Offset= | 0x0000_0004 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SECC_CNT[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DECC_CNT[7:0] | | | | | | | | CLR_DCNT | CLR_SCNT | FIX_DECC | FIX_SECC | DECC_IE | SECC_IE | DECC | SECC |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…16 | SECC_CNT[15:0] | | Счетчик числа одиночных ошибок<br>0 – нет ошибок<br>1 – одна ошибка<br>…<br>65535 – 65535 или более ошибок |
| 15:8 | DECC_CNT[7:0] | | Счетчик числа двойных ошибок<br>0 – нет ошибок<br>1 – одна ошибка<br>…<br>255 – 255 или более ошибок |
| 7 | CLR_DCNT | | Бит сброса счетчика двойных ошибок<br>Запись 1 сбрасывает счетчик DECC_CNT<br>Всегда читается как 0 |
| 6 | CLR_SCNT | | Бит сброса счетчика двойных ошибок<br>Запись 1 сбрасывает счетчик DECC_CNT<br>Всегда читается как 0 |
| 5 | FIX_DECC | | Бит разрешения фиксации в регистрах адреса и данных адреса последней ошибки при двойной ошибке<br>0 – запрещено<br>1 – разрешено |
| 4 | FIX_SECC | | Бит разрешения фиксации в регистрах адреса и данных адреса последней ошибки при одиночной ошибке<br>0 – запрещено<br>1 – разрешено |
| 3 | DECC_IE | | Бит разрешения прерывания при возникновении двойной ошибки ECC<br>0 – прерывание запрещено<br>1 – прерывание разрешено |
| 2 | SECC_IE | | Бит разрешения прерывания при возникновении одиночной ошибки ECC<br>0 – прерывание запрещено<br>1 – прерывание разрешено |
| 1 | DECC | | Флаг возникновения двойной ошибки<br>0 – ошибок не было<br>1 – ошибка была<br>Сбрасывается записью 1 |
| 0 | SECC | | Флаг возникновения одиночной ошибки<br>0 – ошибок не было<br>1 – ошибка была<br>Сбрасывается записью 1 |

### 20.10.3 *ECCADR*

| Base ADDR= | | 0x40005000 | | | Offset= | | 0x0000_0008 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| ECCADR[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECCADR[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | ECCADR[31:0] | | Адрес последней двойной или одиночной ошибки |

### 20.10.4 *ECCDATA*

| Base ADDR= | | 0x40005000 | | | Offset= | | 0x0000_000C | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| ECCDATA[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECCDATA[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | ECCDATA[31:0] | | Считанные данные при последней двойной или одиночной ошибке, без корректировки ECC |

### 20.10.5 *ECCECC*

| Base ADDR= | | 0x40005000 | | | Offset= | | 0x0000_0010 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | | | | | | | | ECC[7:0] | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…8 | - | | Зарезервировано |
| 7…0 | ECC[7:0] | | Считанные ECC-биты при последней двойной или одиночной ошибке, без корректировки ECC |

## 20.11 Описание регистров контроллеров ОЗУ RAMC_CNTR и RAMD_CNTR

Таблица 141 – Регистры контроллеров ОЗУ

| Базовый адрес | Смещение | Название | Состояние после сброса | Описание |
|---|---|---|---|---|
| 0x4000_8000<br>0x4000_9000<br>0x4000_A000<br>0x4000_B000 | 0x0000_0000 | KEY | | Регистр ключа, разрешающего модификацию остальных регистров |
| | 0x0000_0004 | ECCCS | | Регистр статуса ошибок ECCROM |
| | 0x0000_0008 | ECCADR | | Регистр статуса ошибок ECCCRAM |
| | 0x0000_000C | ECCDATA | | Регистр статуса ошибок ECCDRAM |
| | 0x0000_0010 | ECCECC | | Регистр данных ECC последней ошибки |
| | 0x0000_0014 | TEST_TUNING | | Регистр подстройки параметров блока |

### 20.11.1 *KEY*

| Base ADDR= | | 0x4000_8000<br>0x4000_9000<br>0x4000_A000<br>0x4000_B000 | | | Offset= | | 0x0000_0000 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | KEY | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |

| KEY[31:16] | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | KEY[31:0] | 0000_0000 | При записи в регистр значения 0x8555AAA1 открывается возможность записи в другие регистры блока RAM_CNTR |

### 20.11.2 *ECCCS*

| Base ADDR= | 0x4000_8000<br>0x4000_9000<br>0x4000_A000<br>0x4000_B000 | Offset= | 0x0000_0004 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SECC_CNT[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DECC_CNT[7:0] | | | | | | | | CLR_DCNT | CLR_SCNT | FIX_DECC | FIX_SECC | DECC_IE | SECC_IE | DECC | SECC |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…16 | SECC_CNT[15:0] | | Счетчик числа одиночных ошибок<br>0 – нет ошибок<br>1 – одна ошибка<br>…<br>65535 – 65535 или более ошибок |
| 15:8 | DECC_CNT[7:0] | | Счетчик числа двойных ошибок<br>0 – нет ошибок<br>1 – одна ошибка<br>…<br>255 – 255 или более ошибок |
| 7 | CLR_DCNT | | Бит сброса счетчика двойных ошибок<br>Запись 1 сбрасывает счетчик DECC_CNT<br>Всегда читается как 0 |
| 6 | CLR_SCNT | | Бит сброса счетчика двойных ошибок<br>Запись 1 сбрасывает счетчик DECC_CNT<br>Всегда читается как 0 |
| 5 | FIX_DECC | | Бит разрешения фиксации в регистрах адреса и данных адреса последней ошибки при двойной ошибке<br>0 – запрещено<br>1 – разрешено |
| 4 | FIX_SECC | | Бит разрешения фиксации в регистрах адреса и данных адреса последней ошибки при одиночной ошибке<br>0 – запрещено<br>1 – разрешено |
| 3 | DECC_IE | | Бит разрешения прерывания при возникновении двойной ошибки ECC<br>0 – прерывание запрещено<br>1 – прерывание разрешено |
| 2 | SECC_IE | | Бит разрешения прерывания при возникновении одиночной ошибки ECC<br>0 – прерывание запрещено<br>1 – прерывание разрешено |
| 1 | DECC | | Флаг возникновения двойной ошибки<br>0 – ошибок не было<br>1 – ошибка была<br>Сбрасывается записью 1 |
| 0 | SECC | | Флаг возникновения одиночной ошибки<br>0 – ошибок не было<br>1 – ошибка была<br>Сбрасывается записью 1 |

### 20.11.3 *ECCADR*

| Base ADDR= | 0x4000_8000<br>0x4000_9000<br>0x4000_A000<br>0x4000_B000 | Offset= | 0x0000_0008 |
|---|---|---|---|

REG Name:

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECCADR[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECCADR[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | ECCADR[31:0] | | Адрес последней двойной или одиночной ошибки |

### 20.11.4 *ECCDATA*

| Base ADDR= | 0x4000_8000<br>0x4000_9000<br>0x4000_A000<br>0x4000_B000 | Offset= | 0x0000_000C |
|---|---|---|---|

REG Name:

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECCDATA[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECCDATA[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | ECCDATA[31:0] | | Считанные данные при последней двойной или одиночной ошибке, без корректировки ECC |

### 20.11.5 *ECCECC*

| Base ADDR= | 0x4000_8000<br>0x4000_9000<br>0x4000_A000<br>0x4000_B000 | Offset= | 0x0000_0010 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | | | | | | | | ECC[7:0] | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…8 | - | | Зарезервировано |
| 7…0 | ECC[7:0] | | Считанные ECC-биты при последней двойной или одиночной ошибке, без корректировки ECC |

### 20.11.6 *TEST_TUNING*

| Base ADDR= | 0x4000_8000<br>0x4000_9000<br>0x4000_A000<br>0x4000_B000 | Offset= | 0x0000_0014 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31 | RET1N | 1’b1 | Тестовый регистр управления. Не изменять |
| 30..21 | Зарезервировано | 1’b0 | Тестовый регистр управления. Не изменять |
| 20 | Ecc_dis | 1’b0 | Бит запрещения проверки ecc данных RAM.<br>Запрещение проверки позволяет повысить рабочую частоту микроконтроллера<br>0 – разрешено<br>1 – запрещено |
| 19..16 | FMS | 4’b1100 | Тестовый регистр управления. Не изменять |
| 15..9 | Зарезервировано | 0 | Тестовый регистр управления. Не изменять |
| 8..6 | EMA | 010 | Тестовый регистр управления. Не изменять |
| 5..4 | EMAW | 0 | Тестовый регистр управления. Не изменять |
| 3 | WAWL | 0 | Тестовый регистр управления. Не изменять |
| 2…0 | WAWLM | 0 | Тестовый регистр управления. Не изменять |

## 20.12 Описание регистров контроллера FLASH FLASH_CNTR

Таблица 142 – Регистры контроллера FLASH

| Базовый адрес | Смещение | Название | Состояние после сброса | Описание |
|---|---|---|---|---|
| 0x40006000 | 0x0000_0000 | KEY | | Регистр ключа, разрешающего модификацию остальных регистров |
| | 0x0000_0004 | CNTR | | Регистр управления контроллером FLASH |
| | 0x0000_0008 | ADDR | | Регистр адреса записываемых данных |
| | 0x0000_000C<br>0x0000_0010<br>0x0000_0014<br>0x0000_0018 | WDATA3…<br>WDATA0 | | Регистры данных для регистровой записи в банки FLASH данных |
| | 0x0000_001C<br>0x0000_0020 | WECC0…<br>WECC1 | | Регистры данных для регистровой записи в банки FLASH контрольной суммы |
| | 0x0000_0024<br>0x0000_0028<br>0x0000_002C<br>0x0000_0030 | RDATA3…<br>RDATA0 | | Регистры данных, считанных из FLASH данных |
| | 0x0000_0034<br>0x0000_0038 | RECC0…<br>RECC1 | | Регистры данных, считанных из FLASH контрольной суммы |
| | 0x0000_003C | ECCCS0 | | Регистр управления/статуса ошибочными состояниями контрольной суммы |
| | 0x0000_0040 | ECCADR | | Адрес ячейки, в которой возникла ECC-ошибка |
| | 0x0000_0044 | ECCDATA | | Данные ячейки, в которых возникла ecc ошибка |
| | 0x0000_0048 | ECCECC | | ECC поле, в котором возникла ECC-ошибка |
| | 0x0000_004C | BLOCK | | Регистр доступа к защищенной области flash |

### 20.12.1 *KEY*

| Base ADDR= | | 0x40006000 | | | Offset= | | 0x0000_0000 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | KEY | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| KEY[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | KEY[31:0] | 0000_0000 | При записи в регистр значения 0x8555AAA1 открывается возможность записи в другие регистры блока FLASH_CNTR |

### 20.12.2 *CNTR*

| Base ADDR= | 0x40006000 | Offset= | 0x0000_0004 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25…24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FLASH5_CR | FLASH4_CR | FLASH3_CR | FLASH2_CR | FLASH1_CR | FLASH0_CR | breakthrough_mode | ERASE_DONE | rdata_ready | IFREN | NVSTR | PROG | MAS1 | ERASE | YE |
| | | | | | | | | | | | | | | |

| 15 | 14…9 | 8 | 7 | 6 | 5 | 4 | 3…0 |
|---|---|---|---|---|---|---|---|
| XE | SE[5:0] | TMR | ERASE_START | BRKTHRU_DONE | BRKTHRU_STAR | MODE | WAITCYCLE |
| | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31 | FLASH5_CR | | Разрешение управления первым банком ECC (ECC1) из контрольных регистров |
| 30 | FLASH4_CR | | Разрешение управления нулевым банком ECC (ECC0) из контрольных регистров |
| 29 | FLASH3_CR | | Разрешение управления третьим банком данных (FLASH3) из контрольных регистров |
| 28 | FLASH2_CR | | Разрешение управления вторым банком данных (FLASH2) из контрольных регистров |
| 27 | FLASH1_CR | | Разрешение управления первым банком данных (FLASH1) из контрольных регистров |
| 26 | FLASH0_CR | | Разрешение управления нулевым банком данных (FLASH0) из контрольных регистров<br>Каждый бит FLASH_CR отвечает за свой банк памяти. Если бит установлен, мультиплексор для соответствующего банка переключается на управление из регистров.<br>Всего 6 банков: 4 данные и 2 ECC для них.<br>Чтение производится аналогично, значение каждого банка вычитывается в свой регистр, но только в случае, если установлен FLASH_CR банка.<br>Возможно производить одновременное чтение/стирание/запись.<br>Например, осуществить операцию чтения можно так:<br>1. Установить все FLASH_CR и mode.<br>2. Развернуть на регистрах диаграмму чтения.<br>3. Установить сигнал rdata_ready (можно попробовать вместе с SE)<br>4. В регистрах rdata и recc должны быть считанные значения |
| 25..24 | breakthrough_mode | 0b00 | Статус доступа к защищенной области памяти:<br>0b00 – доступ закрыт;<br>0b01 – доступ открыт, защищенный режим 1;<br>0b10 – доступ открыт, защищенный режим 2;<br>0b11 – доступ открыт, открытый режим |
| 23 | ERASE_DONE | | Флаг завершения процедуры стирания FLASH |
| 22 | rdata_ready | | Бит разрешения записи прочитанных данных в регистры |

| | | | |
|---|---|---|---|
| 21 | IFREN | | Бит выбора типа памяти при доступе в память<br>0 – выбор основной памяти;<br>1 – выбор информационной памяти |
| 20 | NVSTR | | Бит задания сигнала программирования памяти в режиме регистрового доступа |
| 19 | PROG | | Бит задания сигнала программирования памяти в режиме регистрового доступа |
| 18 | MAS1 | | Бит задания сигнала общего стирания памяти в режиме регистрового доступа |
| 17 | ERASE | | Бит задания сигнала стирания памяти в режиме регистрового доступа. |
| 16 | YE | | Бит задания сигнала разрешения старшей части адреса в режиме регистрового доступа. |
| 15 | XE | | Бит формирования сигнала разрешения младшей части адреса в режиме регистрового доступа. |
| 14..9 | SE[5:0] | | Биты формирования сигнала обращения к блокам памяти FLASH<br>SE[0] - выбор нулевого банка данных, FLASH 0;<br>SE[1] - выбор первого банка данных, FLASH 1;<br>SE[2] - выбор второго банка данных, FLASH 2;<br>SE[3] - выбор третьего банка данных, FLASH 3;<br>SE[4] - выбор нулевого банка ECC, ECC 0;<br>SE[5] - выбор первого банка ECC, ECC1 |
| 8 | TMR | | Бит формирования сигнала тестового режима для блоков FLASH |
| 7 | ERASE_START | | Начало процедуры стирания FLASH, в результате которой открывается доступ ко FLASH |
| 6 | BRKTHRU_ DONE | | Флаг завершения процедуры доступа к закрытой части памяти |
| 5 | BRKTHRU_ START | | Начало процедуры доступа к закрытой части памяти Самоочищаемый бит |
| 4 | MODE | | Режим доступа во FLASH<br>0 – режим доступа со стороны системы;<br>1 – режим доступа со стороны регистров |
| 3..0 | WAITCYCLE | | Количества циклов ожидания при обращении в память.<br>0000 – зарезервировано;<br>0001 – $T_{READ}$ = 0,5 • $T_{HCLK}$;<br>0010 – $T_{READ}$ = 1,5 • $T_{HCLK}$;<br>0011 – $T_{READ}$ = 2,5 • $T_{HCLK}$;<br>0100 – $T_{READ}$ = 3,5 • $T_{HCLK}$;<br>....<br>1111 – $T_{READ}$ = 14,5 • $T_{HCLK}$ |

### 20.12.3 *ADDR*

| Base ADDR= | | 0x40006000 | | | Offset= | | 0x0000_0008 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| ADDR[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDR[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | ADDR[31:0] | | Адрес, по которому производится запись/чтение |

### 20.12.4 *WDATA3…WDATA0*

| Base ADDR= | 0x40006000 | Offset= | 0x0000_000C<br>0x0000_0010<br>0x0000_0014<br>0x0000_0018 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WDATA X | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WDATA X | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | WDATA X | | Данные записываемые в блоки памяти |

### 20.12.5 *WECC0 .. 1*

| Base ADDR= | 0x40006000 | Offset= | 0x0000_001C<br>0x0000_0020 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WECC | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WECC | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | WECC | | ECC для записи в блок |

### 20.12.6 *RDATA3…RDATA0*

| Base ADDR= | 0x40006000 | Offset= | 0x0000_0024<br>0x0000_0028<br>0x0000_002C<br>0x0000_0030 |
|---|---|---|---|

| REG Name: | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RDATA X | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RDATA X | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | RDATA X | | Данные читаемые из блока памяти |

### 20.12.7 *RECC 0..1*

| Base ADDR= | 0x40006000 | Offset= | 0x0000_0034<br>0x0000_0038 |
|---|---|---|---|

| REG Name: | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RECC | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RECC | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | RECC | | ECC читаемый из блока |

### 20.12.8 *ECCCS*

| Base ADDR= | 0x40006000 | Offset= | 0x0000_003C |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SECC_CNT[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DECC_CNT[7:0] | | | | | | | | CLR_DCNT | CLR_SCNT | FIX_DECC | FIX_SECC | DECC_IE | SECC_IE | DECC | SECC |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…16 | SECC_CNT[15:0] | | Счетчик числа одиночных ошибок<br>0 – нет ошибок<br>1 – одна ошибка<br>…<br>65535 – 65535 или более ошибок |
| 15:8 | DECC_CNT[7:0] | | Счетчик числа двойных ошибок<br>0 – нет ошибок<br>1 – одна ошибка<br>…<br>255 – 255 или более ошибок |
| 7 | CLR_DCNT | | Бит сброса счетчика двойных ошибок<br>Запись 1 сбрасывает счетчик DECC_CNT<br>Всегда читается как 0 |
| 6 | CLR_SCNT | | Бит сброса счетчика двойных ошибок<br>Запись 1 сбрасывает счетчик DECC_CNT<br>Всегда читается как 0 |
| 5 | FIX_DECC | | Бит разрешения фиксации в регистрах адреса и данных адреса последней ошибки при двойной ошибке<br>0 – запрещено<br>1 – разрешено |
| 4 | FIX_SECC | | Бит разрешения фиксации в регистрах адреса и данных адреса последней ошибки при одиночной ошибке<br>0 – запрещено<br>1 – разрешено |
| 3 | DECC_IE | | Бит разрешения прерывания при возникновении двойной ошибки ECC<br>0 – прерывание запрещено<br>1 – прерывание разрешено |
| 2 | SECC_IE | | Бит разрешения прерывания при возникновении одиночной ошибки ECC<br>0 – прерывание запрещено<br>1 – прерывание разрешено |
| 1 | DECC | | Флаг возникновения двойной ошибки<br>0 – ошибок не было<br>1 – ошибка была<br>Сбрасывается записью 1 |
| 0 | SECC | | Флаг возникновения одиночной ошибки<br>0 – ошибок не было<br>1 – ошибка была<br>Сбрасывается записью 1 |

### 20.12.9 *ECCADR*

| Base ADDR= | 0x40006000 | Offset= | 0x0000_0040 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECCADR[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECCADR[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | ECCADR[31:0] | | Адрес последней двойной или одиночной ошибки |

### 20.12.10 *ECCDATA*

| Base ADDR= | 0x40006000 | Offset= | 0x0000_0044 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECCDATA[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECCDATA[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | ECCDATA[31:0] | | Считанные данные при последней двойной или одиночной ошибке, без корректировки ECC |

### 20.12.11 *ECCECC*

| Base ADDR= | 0x40006000 | Offset= | 0x0000_0048 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | | | | | | | | ECC[7:0] | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…8 | - | | Зарезервировано |
| 7…0 | ECC[7:0] | | Считанные ECC-биты при последней двойной или одиночной ошибке, без корректировки ECC. |

### 20.12.12 *BLOCK*

| Base ADDR= | 0x40006000 | Offset= | 0x0000_004C |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BLOCK | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BLOCK | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | BLOCK | | Запись значения 32’hb3c3b3c3 разрешает процесс открытия закрытой области памяти |

## 20.13 Описание регистров контроллера батарейного домена BKP_CNTR

Батарейный домен имеет 64 встроенных 32-разрядных регистров. С 60 по 64 регистры служат для хранения битов управления батарейным доменом, микроконтроллером и часами реального времени, оставшиеся 60 регистров могут быть использованы разработчиком программы.

Таблица 143 – Регистры контроллера батарейного домена

| Базовый адрес | Смещение | Название | Состояние после сброса | Описание |
|---|---|---|---|---|
| 0x4000_1000 | 0x0000_00F0 | KEY | | Регистр ключа, разрешающего модификацию остальных регистров |
| **Память батарейного домена** | | | | |
| | 0x0000_0000 | REG_00 | | Регистр батарейной памяти 00 |
| | 0x0000_0004 | REG_01 | | Регистр батарейной памяти 01 |
| | | | | |
| | 0x0000_00EC | REG_59 | | Регистр батарейной памяти 59 |
| | | | | |
| | 0x0000_010C | REG_64_TMR0 | | |
| | 0x0000_011C | REG_64_TMR1 | | |
| | 0x0000_012C | REG_64_TMR2 | | |
| | | | | |
| | 0x0000_0100 | REG_60_SYS0 | | Регистр батарейной памяти 60 с функциями управления |
| | 0x0000_0104 | REG_61_PWR0 | | Регистр батарейной памяти 61 с функциями управления |
| | 0x0000_0108 | REG_62_PWR0 | | Регистр батарейной памяти 62 с функциями управления |
| | 0x0000_010C | REG_63_CLK0 | | Регистр батарейной памяти 63 с функциями управления |
| | | | | |
| | 0x0000_0110 | REG_60_SYS1 | | Регистр батарейной памяти 60 с функциями управления |
| | 0x0000_0114 | REG_61_PWR1 | | Регистр батарейной памяти 61 с функциями управления |
| | 0x0000_0118 | REG_62_PWR1 | | Регистр батарейной памяти 62 с функциями управления |
| | 0x0000_011C | REG_63_CLK1 | | Регистр батарейной памяти 63 с функциями управления |
| | | | | |
| | 0x0000_0120 | REG_60_SYS2 | | Регистр батарейной памяти 60 с функциями управления |
| | 0x0000_0124 | REG_61_PWR2 | | Регистр батарейной памяти 61 с функциями управления |
| | 0x0000_0128 | REG_62_PWR2 | | Регистр батарейной памяти 62 с функциями управления |
| | 0x0000_012C | REG_63_CLK2 | | Регистр батарейной памяти 63 с функциями управления |
| **Регистры часов реального времени** | | | | |
| | 0x0000_0130 | RTC_CNT_TMR0 | | Регистр основного счетчика часов реального времени |
| | 0x0000_0134 | RTC_DIV_TMR0 | | Регистр счетчика предварительного делителя |
| | 0x0000_0138 | RTC_PRL_TMR0 | | Регистр коэффициента деления предварительного делителя |

| Базовый адрес | Смещение | Название | Состояние после сброса | Описание |
|---|---|---|---|---|
| | 0x0000_013C | RTC_ALRM_TMR0 | | Регистр значения для сравнения с основным счетчиком RTC_CNT и выработки сигнала ALRF |
| | 0x0000_0140 | RTC_CS_TMR0 | | Регистр управления и состояния часов реального времени |
| | | | | |
| | 0x0000_0150 | RTC_CNT_TMR1 | | Регистр основного счетчика часов реального времени |
| | 0x0000_0154 | RTC_DIV_TMR1 | | Регистр счетчика предварительного делителя |
| | 0x0000_0158 | RTC_PRL_TMR1 | | Регистр коэффициента деления предварительного делителя |
| | 0x0000_015C | RTC_ALRM_TMR1 | | Регистр значения для сравнения с основным счетчиком RTC_CNT и выработки сигнала ALRF |
| | 0x0000_0160 | RTC_CS_TMR1 | | Регистр управления и состояния часов реального времени |
| | | | | |
| | 0x0000_0170 | RTC_CNT_TMR2 | | Регистр основного счетчика часов реального времени |
| | 0x0000_0174 | RTC_DIV_TMR2 | | Регистр счетчика предварительного делителя |
| | 0x0000_0178 | RTC_PRL_TMR2 | | Регистр коэффициента деления предварительного делителя |
| | 0x0000_017C | RTC_ALRM_TMR2 | | Регистр значения для сравнения с основным счетчиком RTC_CNT и выработки сигнала ALRF |
| | 0x0000_0180 | RTC_CS_TMR2 | | Регистр управления и состояния часов реального времени |
| | | | | |
| | 0x0000_0190 | BLDO_CNTR0 | | Регистр управления батарейным питанием |
| | 0x0000_0194 | BLDO_CNTR1 | | Регистр управления батарейным питанием |
| | 0x0000_019C | BLDO_CNTR2 | | Регистр управления батарейным питанием |

### 20.13.1 *KEY*

| Base ADDR= | 0x4000_1000 | Offset= | 0x0000_00F0 |
|---|---|---|---|
| REG Name: | KEY | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY[31:16] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31…0 | KEY[31:0] | R/W | 0000_0000 | При записи в регистр значения 0x8555AAA1 открывается возможность записи в другие регистры блока BKP_CNTR |

### 20.13.2 *BLDO_CNTRx*

| Base ADDR= | 0x4000_1000 | Offset= | 0x0000_0190<br>0x0000_0194<br>0x0000_019C |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30…16 |
|---|---|
| BLDO_ERR_CLR | |
| R/W | - |

| 15…8 | 7 | 6…4 | 3…1 | 0 |
|---|---|---|---|---|
| | BLDO_READY | BLDO_SIRLOW | BLDO_TRIM | BLDO_DIS |
| - | RO | R/W | R/W | R/W |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31 | BLDO_ERR_CLR | R/W | 0 | Бит сброса флага ошибки в BLDO_CNTRLx<br>0 – нет сброса;<br>1 – сброс<br>Флаг не сбрасывается, если в момент его сброса событие сбоя продолжается. |
| 30…8 | - | - | - | Зарезервировано |
| 7 | BLDO_READY | RO | | Флаг готовности питания батарейного домена |
| 6..4 | BLDO_SIRLOW | R/W | 000 | Калибровка регулятора питания батарейного домена<br>000 – типовое значение (потребление 15мА);<br>001…011 – увеличение ожидаемого потребления;<br>100…111 – уменьшение ожидаемого потребления. |
| 3..1 | BLDO_TRIM | R/W | 000 | Подстройка регулятора питания батарейного домена<br>000 – типовое значение;<br>001…011 – увеличение ожидаемого потребления;<br>100…111 – уменьшение ожидаемого потребления. |
| 0 | BLDO_DIS | R/W | 0 | Отключение регулятора питания батарейного домена<br>0 – включено<br>1 – выключено |

### 20.13.3 *REG_00…REG_59*

| Base ADDR= | 0x4000_1000 | Offset= | 0x0000_0000<br>0x0000_0004<br>..<br>0x0000_00EC |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA[31:16] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31…0 | DATA[31:0] | R/W | | Данные памяти батарейного домена |

### 20.13.4 *REG_60_SYSx*

| Base ADDR= | 0x4000_1000 | Offset= | 0x0000_0100<br>0x0000_0110<br>0x0000_0120 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29, 28 | 27 | 26 | 25 | 24…16 |
|---|---|---|---|---|---|---|
| ERR_IE | CLR_ERR | | ERR_63 | ERR_LDO | ERR_60 | |
| R/W | R/W | - | RO | RO | RO | - |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6…0 |
|---|---|---|---|---|---|---|---|---|---|
| | WDGRST0n | FTRST | IORST0n | | SYSRSTn | LOCKSTEP_DIS | DISABLE_JTAG | FPOR | MODE[6:0] |
| - | RO | RO | RO | - | RO | R/W | R/W | R/W | R/W |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31 | ERR_IE | R/W | 0 | Разрешение прерывания BKP_IF по ошибкам ERR<br>0 –прерывание запрещено<br>1 – прерывание разрешено |
| 30 | CLR_ERR | R/W | 0 | Бит сброса флагов ошибки<br>Запись 0 – нет эффекта<br>Запись 1 – сброс флагов ERR_63, ERR_LDO, ERR_60<br>Флаги не сбрасываются, если в момент их сброса событие сбоя продолжается.<br>Сигнал сброса вырабатывается, если во всех трех регистрах REG_60_SYSx записаны единицы.<br>После сброса флагов ошибок бит CLR_ERR необходимо программно сбросить в 0 |
| 29 | Зарезервировано | | | Зарезервировано |
| 28 | Зарезервировано | | | Зарезервировано |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 27 | ERR_63 | RO | 0 | Сбой конфигурации регистров REG_63<br>0 – нет сбоя<br>1 – есть сбой |
| 26 | ERR_LDO | RO | 0 | Сбой конфигурации регистров REG_61 и REG_62<br>0 – нет сбоя<br>1 – есть сбой |
| 25 | ERR_60 | RO | 0 | Сбой конфигурации регистров REG_60<br>0 – нет сбоя<br>1 – есть сбой |
| 24..15 | Зарезервировано | RO | | Зарезервировано |
| 14 | WDGRST0n | RO | 0 | Флаг события сброса с вывода WDGRSTn<br>0 – не было сброса по WDGRSTn<br>1 – был сброс по WDGRSTn<br>Флаг сбрасывается записью 1, если в момент очистки флага происходит очередное событие, флаг не будет очищен |
| 13 | FTRST | RO | 0 | Флаг события сброса с вывода FTRST<br>0 – не было сброса по FTRST<br>1 – был сброс по FTRST<br>Флаг сбрасывается записью 1, если в момент очистки флага происходит очередное событие, флаг не будет очищен |
| 12 | IORST0n | RO | 0 | Флаг события сброса с вывода IORST0n<br>0 – не было сброса по IORST0n<br>1 – был сброс по IORST0n<br>Флаг сбрасывается записью 1, если в момент очистки флага происходит очередное событие, флаг не будет очищен |
| 11 | - | - | - | Зарезервировано |
| 10 | SYSRSTn | RO | 0 | Флаг события сброса с вывода SYSRSTn<br>0 – не было сброса по SYSRSTn<br>1 – был сброс по SYSRSTn<br>Флаг сбрасывается записью 1, если в момент очистки флага происходит очередное событие, флаг не будет очищен |
| 9 | LOCKSTEP_DIS | R/W | 0 | Флаг разрешения работы в режиме LOCKSTEP<br>0 – режим LOCKSTEP включен<br>1 – режим LOCKSTEP отключен (режим DUALCORE)<br><br>Сбрасывается в 0 при сбросах HW RESET, SYSRESETREQ.<br><br>Режим LOCKSTEP выключается, если во всех трех каналах записана 1, если хотя бы в одном канале остаётся 0, то выбран режим LOCKSTEP |
| 8 | DISABLE_JTAG | R/W | 0 | Запрещение отладочного интерфейса JTAG<br>0 –JTAG_A и/или JTAG_B разрешены<br>1–JTAG_A и/или JTAG_B запрещены |
| 7 | FPOR | R/W | 0 | Флаг успешного завершения работы загрузочной программы<br>0 – загрузочная программа не завершена<br>1 – загрузочная программа завершена<br>Возможна только запись 1. Сброс по любому сигналу сброса |
| 6…0 | MODE[6:0] | R/W | xxxxxx | Режим запуска микроконтроллера с выводов MODE<br>JTAG_A = (MODE[0]==0) & ~DISABLE_JTAG;<br>JTAG_B = (MODE[0]==1) & ~DISABLE_JTAG;<br>JTAG_EN = (MODE[3:0]==4'b1111) |

### 20.13.5 *REG_61_PWRx*

| Base ADDR= | | 0x4000_1000 | | Offset= | | 0x0000_0104<br>0x0000_0114<br>0x0000_0124 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| STANDBY | TPROT_EVENT | CPROT_EVENT | HLDO_RDY | DCDC_RDY | LDO_RDY | nOVRST3p3 | nPOR1p2 | nPOR1p6 | nPOR3p3 | POR3p3_RDY | | CPROT_EN | TPROT_EN | LLDO_DIS | HLDO_DIS |
| R/W | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | - | R/W | R/W | R/W | R/W |

| 15…13 | 12 | 11…9 | 8 | 7…5 | 4…2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| HLDO_SIRLOW | | DCDC_TRIM_TOFF | DCDC_EN | TRIM_Vcc1p2 | TRIM_Vcc1p6 | OVR3p3_DIS | POR3p3_DIS |
| R/W | - | R/W | R/W | R/W | R/W | R/W | R/W |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31 | STANDBY | R/W | 0 | Перевод регулятора в режим пониженного потребления<br>0 – нормальный режим<br>1 – режим пониженного потребления (выключение регуляторов)<br>В случае сброса блока, возникновения высокого уровня сигналов ALRF или WAKEUP сбрасывается |
| 30 | TPROT_EVENT | RO | 0 | Сигнал превышения уровня температуры<br>0 – нормальная температура<br>1 – уровень превышен |
| 29 | CPROT_EVENT | RO | 0 | Флаг превышения лимита количества превышений токовых значений:<br>0 – cprot_cnt < cprot_lim:<br>1 – cprot_cnt >= cprot_lim |
| 28 | HLDO_RDY | RO | 0 | Флаг готовности LDO 1,6 В<br>0 – не готов<br>1 – готов |
| 27 | DCDC_RDY | RO | 0 | Флаг готовности DCDC<br>0 – не готов<br>1 – готов |
| 26 | LDO_RDY | RO | 0 | Флаг готовности LDO 1,2 В<br>0 – не готов<br>1 – готов |
| 25 | nOVRST3p3 | RO | 0 | Сброс по превышению питания 3,3 В |
| 24 | nPOR1p2 | RO | 0 | Сброс по питанию 1,2 В |
| 23 | nPOR1p6 | RO | 0 | Сброс по питанию 1,6 В |
| 22 | nPOR3p3 | RO | 0 | Сброс по питанию 3,3 В |
| 21 | POR3p3_RDY | RO | 0 | Сигнал готовности питания 3,3 В |
| 20 | - | | | Зарезервировано |
| 19 | CPROT_EN | R/W | 0 | Разрешение защиты по превышению тока<br>0 – запрещено<br>1 – разрешено |
| 18 | TPROT_EN | R/W | 0 | Разрешение защиты по температуре<br>0 – запрещено<br>1 – разрешено |
| 17 | LLDO_DIS | R/W | 0 | Запрещение работы регулятора питания 1,2 В |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| | | | | 0 – разрешено<br>1 – запрещено |
| 16 | HLDO_DIS | R/W | 0 | Запрещение работы регулятора питания 1,6 В<br>0 – разрешено<br>1 – запрещено |
| 15…13 | HLDO_SIRLOW | R/W | 000 | Настройка работы регулятора питания 1,6 В<br>001…011 – увеличение дополнительного тока;<br>000 – типовое значение дополнительного тока;<br>100…111 – уменьшение дополнительного тока<br>Рекомендуется устанавливать значение 011 |
| 12 | - | | | Зарезервировано |
| 11...9 | DCDC_TRIM_ TOFF | R/W | 000 | Подстройка параметра toff для DCDC<br>000 – типовое значение toff;<br>001…011 – увеличение toff;<br>100…111 – уменьшение toff |
| 8 | DCDC_EN | R/W | 0 | Разрешение работы регулятора питания 1,6 В<br>0 – запрещено<br>1 – разрешено |
| 7…5 | TRIM_Vcc1p2 | R/W | 000 | Подстройка напряжения питания 1,2В<br>011 – номинальное значение VLLDO;<br>001…011 – увеличение VLLDO;<br>100…111 – уменьшение VLLDO |
| 4…2 | TRIM_Vcc1p6 | R/W | 000 | Подстройка напряжения питания 1,6В<br>101 – номинальное значение VHLDO;<br>001…011 – увеличение VHLDO;<br>100…111 – уменьшение VHLDO |
| 1 | OVR3p3_DIS | R/W | 0 | Разрешение работы блока сброса по превышению питания<br>0 – разрешено<br>1 – запрещено |
| 0 | POR3p3_DIS | R/W | 0 | Запрещение работы блока сброса по установлению питания<br>0 – разрешено<br>1 – запрещено |

### 20.13.6 *REG_62_PWRx*

| Base ADDR= | 0x4000_1000 | Offset= | 0x0000_0108<br>0x0000_0118<br>0x0000_0128 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23…19 | 18…16 |
|---|---|---|---|---|---|---|---|---|---|
| PMU_BG_DIS | SEL_SW | SEL_nSW | PWRM_IO_Vcc_EVENT | PWRM_B_Vcc_EVENT | PWRM_Vcc_EVENT | PWRM_IO_Vcc_EN | PWRM_B_Vcc_EN | PWRM_IO_Vcc_LVL | PWRM_B_Vcc_LVL |
| R/W | RO | RO | RO | RO | RO | R/W | R/W | R/W | R/W |

| 15…14 | 13 | 12…8 | 7 | 6…0 |
|---|---|---|---|---|
| PWRM_B_Vcc_LVL | PWRM_Vcc_EN | PWRM_Vcc_LVL | BLDO_RDY | |
| R/W | R/W | R/W | RO | - |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31 | PMU_BG_DIS | R/W | 0 | Бит разрешения работы генератора опорного напряжения pmu<br>0 – включено;<br>1 – выключено |
| 30 | SEL_SW | RO | 0 | Флаг переключения напряжения питания<br>1 – выбрано основное питание;<br>0 – выбрано батарейное питание |
| 29 | SEL_nSW | RO | 0 | Инверсный флаг переключения напряжения питания<br>1 – выбрано батарейное питание;<br>0 – выбрано основное питание |
| 28 | PWRM_IO_Vcc_EVENT | RO | 0 | Флаг события от блока монитора напряжения io<br>0 – нет превышения;<br>1 – превышение напряжения, задаваемого через PWRM_IO_Vcc_LVL |
| 27 | PWRM_B_Vcc_EVENT | RO | 0 | Флаг события от блока монитора батарейного напряжения<br>0 – нет превышения;<br>1 - превышение напряжения, задаваемого через PWRM_B_Vcc_LVL |
| 26 | PWRM_Vcc_EVENT | RO | 0 | События от блока монитора 3В напряжения<br>0 – нет превышения;<br>1 - превышение напряжения, задаваемого через PWRM_Vcc_LVL |
| 25 | PWRM_IO_Vcc_EN | R/W | 0 | Разрешение монитора питания $V_{CC_IO}$<br>0 – монитор выключен;<br>1 – монитор включен |
| 24 | PWRM_B_Vcc_EN | R/W | 0 | Разрешение монитора питания $V_{CC_BKP}$<br>0 – монитор выключен<br>1 – монитор включен |
| 23..19 | PWRM_IO_Vcc_LVL | R/W | 00000 | Уровень монитора питания IO<br>00000 – более 1,2В<br>00001 – более 1,3В<br>…<br>11111 – более 4,3В |
| 18..14 | PWRM_B_Vcc_LVL | R/W | 00000 | Уровень монитора питания BKP<br>00000 – более 1,2В<br>00001 – более 1,3В |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| | | | | …<br>11111 – более 4,3В |
| 13 | PWRM_Vcc_EN | R/W | 0 | Разрешение монитора питания $V_{CC_DCDC}$<br>0 – монитор выключен<br>1 – монитор включен |
| 12..8 | PWRM_Vcc_LVL | R/W | 00000 | Уровень монитора питания DCDC<br>00000 – более 1,2В<br>00001 – более 1,3В<br>…<br>11111 – более 4,3В |
| 7 | BLDO_RDY | RO | 0 | Флаг готовности питания батарейного домена BLDO<br>0 – питание не сформировано;<br>1 – питание батарейного домена готово |
| 6..0 | - | - | - | Зарезервировано |

### 20.13.7 *REG_63_CLKx*

| Base ADDR= | | 0x4000_1000 | Offset= | 0x0000_010C<br>0x0000_011C<br>0x0000_012C | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | |

| 31 | 30 | 29 | 28…22 | 21 | 20…16 |
|---|---|---|---|---|---|
| gen_por_rdy | | hsi_off | hsi_trim | hsi_rdy | lsi_trim |
| RO | - | R/W | R/W | RO | R/W |

| 15…14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| lsi_trim | lsi_off | lsi_rdy | hse1_on | hse1_byp | hse1_filter_en | hse1_rdy | hse0_on | hse0_byp | hse0_filter_en | hse0_rdy | lse_on | lse_byp | lse_filter_en | lse_rdy |
| R/W | R/W | RO | R/W | R/W | R/W | RO | R/W | R/W | R/W | RO | R/W | R/W | R/W | RO |

| Бит | Имя | Значение по умолчанию | Доступ | Описание |
|---|---|---|---|---|
| 31 | gen_por_rdy | 0 | RO | Флаг готовности встроенного генератора POR:<br>0 – генератор не вышел в режим<br>1 – генератор находится в рабочем режиме |
| 30 | - | | | Зарезервировано |
| 29 | hsi_off | 0 | R/W | Бит выключения HSI-генератора<br>0 – генератор включен<br>1 – генератор выключен |
| 28..22 | hsi_trim | 1000000 | R/W | Коэффициент подстройки частоты генератора HSI. |
| 21 | hsi_rdy | 0 | RO | Флаг выхода генератора HSI в рабочий режим:<br>0 – генератор не запущен или не вышел в режим;<br>1 – генератор работает в рабочем режиме |
| 20..14 | lsi_trim | 1000000 | R/W | Коэффициент подстройки частоты генератора LSI |
| 13 | lsi_off | 0 | R/W | Бит управления генератором LSI:<br>0 – генератор включен;<br>1 – генератор выключен; |
| 12 | lsi_rdy | 0 | RO | Флаг выхода генератора LSI в рабочий режим:<br>0 – генератор не запущен или не вышел в режим;<br>1 – генератор находится в рабочем режиме |
| 11 | hse1_on | 0 | R/W | Бит разрешения работы HSE1 генератора<br>0 – генератор выключен<br>1 – генератор включен |

| Бит | Имя | Значение по умолчанию | Доступ | Описание |
|---|---|---|---|---|
| 10 | hse1_byp | 0 | R/W | Бит управления генератором HSE1:<br>0 – режим работы с внешним резонатором<br>1 – режим работы с внешним генератором |
| 9 | hse1_filter_en | 0 | R/W | Фильтрация входных сигналов генератора HSE1<br>0 – фильтр выключен<br>1 – фильтр включен |
| 8 | hse1_rdy | 0 | RO | Флаг выхода генератора HSE1 в рабочий режим:<br>0 – генератор не запущен или не вышел в режим;<br>1 – генератор работает в рабочем режиме |
| 7 | hse0_on | 0 | R/W | Бит разрешения работы HSE0 генератора<br>0 – генератор выключен<br>1 – генератор включен |
| 6 | hse0_byp | 0 | R/W | Бит управления генератором HSE0:<br>0 – режим работы с внешним резонатором<br>1 – режим работы с внешним генератором |
| 5 | hse0_filter_en | 0 | R/W | Фильтрация входных сигналов генератора HSE0<br>0 – фильтр выключен<br>1 – фильтр включен |
| 4 | hse0_rdy | 0 | RO | Флаг выхода генератора HSE0 в рабочий режим:<br>0 – генератор не запущен или не вышел в режим;<br>1 – генератор работает в рабочем режиме |
| 3 | lse_on | 0 | R/W | Бит разрешения работы LSE генератора<br>0 – генератор выключен<br>1 – генератор включен |
| 2 | lse_byp | 0 | R/W | Бит управления генератором LSE:<br>0 – режим работы с внешним резонатором<br>1 – режим работы с внешним генератором |
| 1 | lse_filter_en | 0 | R/W | Фильтрация входных сигналов генератора LSE<br>0 – фильтр выключен<br>1 – фильтр включен |
| 0 | lse_rdy | 0 | RO | Флаг выхода генератора LSE в рабочий режим:<br>0 – генератор не запущен или не вышел в режим;<br>1 – генератор работает в рабочем режиме |

### 20.13.8 *REG_64_TMRx*

| Base ADDR= | 0x4000_1000 | | Offset= | 0x0000_010C<br>0x0000_011C<br>0x0000_012C | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | |

| 31…22 | 21…16 |
|---|---|
| cprot_cnt | cprot_lim |
| RO | R/W |

| 15…12 | 11 | 10 | 9…2 | 1…0 |
|---|---|---|---|---|
| cprot_lim | RTCEN | nRTCRST | RTCCAL | RTCSEL |
| R/W | R/W | R/W | R/W | R/W |

| Бит | Имя | Значение | Доступ | Описание |
|---|---|---|---|---|
| 31..22 | cprot_cnt | 0 | RO | Счетчик достижения предельного значения тока |
| 21..12 | cprot_lim | 0 | R/W | Лимит достижения предельного значения тока |
| 11 | RTCEN | 0 | R/W | Бит разрешения работы часов реального времени:<br>0 – работа запрещена;<br>1 – работа разрешена.<br>После записи задержка включения/выключения RTC составляет 1 такт частоты RTC_CLK |

| Бит | Имя | Значение | Доступ | Описание |
|---|---|---|---|---|
| 10 | nRTCRST | 0 | R/W | Сброс часов реального времени:<br>0 – часы сбрасываются;<br>1 – часы не сбрасываются |
| 9…2 | RTCCAL | 00000000 | R/W | Коэффициент подстройки тактовой частоты часов реального времени RTC_CLK. Из каждых $2^{20}$ тактов RTC_CLK будет замаскировано RTCCAL тактов:<br>00000000 – 0 тактов<br>00000001 – 1 такт<br>….<br>11111111 – 255 тактов<br>Таким образом, если исходная частота RTC_CLK равна 32768,00000 Гц, то после подстройки средняя частота тактирования составляет:<br>RTC_CLK_C = 32768,00000 Гц при RTCCAL = 0;<br>RTC_CLK_C = 32767,96875 Гц при RTCCAL = 1;<br>…<br>RTC_CLK_C = 32760,03125 Гц при RTCCAL = 255.<br>В общем случае средняя тактовая частота RTC_CLK после подстройки составляет:<br>RTC_CLK_C = RTC_CLK · (1 – RTCCAL[7:0] / $2^{20}$)<br>Единица подстройки (замедления) равна $1/2^{20}$,<br>или ~0,95 ppm. |
| 1..0 | RTCSEL | 00 | R/W | Биты выбора источника тактовой синхронизации часов реального времени RTC_CLK:<br>00 – LSI<br>01 – LSE<br>10 – RTCCLK (формируется в блоке CLK_CNTR)<br>11 - зарезервировано |

### 20.13.9 *RTC_CNT_TMRx*

| Base ADDR= | | 0x4000_1000 | | | Offset= | | 0x0000_0130<br>0x0000_0150<br>0x0000_0170 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RTC_CNT[31:16] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RTC_CNT[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31…0 | RTC_CNT[31:0] | R/W | 0000_0000 | Значение основного счетчика часов реального времени |

### 20.13.10 *RTC_DIV_TMRx*

| Base ADDR= | | 0x4000_1000 | | | Offset= | | 0x0000_0134<br>0x0000_0154<br>0x0000_0174 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | | | | | | | | | | | | RTC_DIV[19:16] | | | |
| - | - | - | - | - | - | - | - | - | - | - | - | RO | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RTC_DIV[15:0] | | | | | | | | | | | | | | | |
| RO | | | | | | | | | | | | | | | |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31…20 | - | - | - | Зарезервировано |
| 19…0 | RTC_DIV[19:0] | RO | 00000 | Значение счетчика предварительного делителя часов реального времени |

### 20.13.11 *RTC_PRL_TMRx*

| Base ADDR= | | 0x4000_1000 | | | Offset= | | 0x0000_0138<br>0x0000_0158<br>0x0000_0178 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | | | | | | | | | | | | RTC_PRL[19:16] | | | |
| - | - | - | - | - | - | - | - | - | - | - | - | R/W | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RTC_PRL[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31…20 | - | - | - | Зарезервировано |
| 19…0 | RTC_PRL[19:0] | R/W | 00000 | Коэффициент деления тактовой частоты RTC_CLK:<br>SEC_CLK = RTC_CLK / (RTC_PRL + 1),<br>где RTC_CLK – частота тактирования с учетом подстройки RTCCAL[7:0] |

### 20.13.12 *RTC_ALRM_TMRx*

| Base ADDR= | | 0x4000_1000 | | | Offset= | | 0x0000_013C<br>0x0000_015C<br>0x0000_017C | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RTC_ALRM[31:16] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RTC_ALRM[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31…0 | RTC_ALRM[31:0] | R/W | 0000_0000 | Значение для сравнения с основным счетчиком RTC_CNT и выработки сигнала ALRF.<br>Сигнал ALRF вырабатывается в момент превышения основным счётчиком RTC_CNT значения в регистре RTC_ALRM:<br>RTC_CNT[31:0] == RTC_ALRM[31:0] + 1 |

### 20.13.13 *RTC_CS_TMRx*

| Base ADDR= | 0x4000_1000 | Offset= | 0x0000_0140<br>0x0000_0160<br>0x0000_0180 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | | | | | | | | | | | | | | | |
| - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ERR_IE | ERR_CMx | ERR_DIVx | ERR_CNTx | ERR_CC | ERR_CM | ERR_DIV | ERR_CNT | CLR_ERRx | WEC | ALRF_IE | SECF_IE | OWF_IE | ALRF | SECF | OWF |
| R/W | RO | RO | RO | RO | RO | RO | RO | WO | RO | R/W | R/W | R/W | R/W | R/W | R/W |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31…16 | - | - | - | Зарезервировано |
| 15 | ERR_IE | RW | 0 | Флаг разрешения прерывания RTC_IF по событиям сбоя<br>0 – прерывание запрещено<br>1 – прерывание разрешено |
| 14 | ERR_CMx | RO | 0 | Сбой в данном канале тактовой частоты RTC<br>0 – нет сбоя<br>1 – есть сбой |
| 13 | ERR_DIVx | RO | 0 | Сбой в данном канале DIV часов RTC<br>0 – нет сбоя<br>1 – есть сбой |
| 12 | ERR_CNTx | RO | 0 | Сбой в данном канале CNT часов RTC<br>0 – нет сбоя<br>1 – есть сбой |
| 11 | ERR_CC | RO | 0 | Отказ источника тактовой частоты RTC<br>0 – нет сбоя<br>1 – есть сбой<br>При отказе источника тактовой частоты, RTC автоматически переключается на LSI-генератор. |
| 10 | ERR_CM | RO | 0 | Сбой в одном из каналов тактовой частоты RTC превышения максимальной частоты<br>0 – нет сбоя<br>1 – есть сбой |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 9 | ERR_DIV | RO | 0 | Сбой в одном из каналов DIV часов RTC<br>0 – нет сбоя<br>1 – есть сбой |
| 8 | ERR_CNT | RO | 0 | Сбой в одном из каналов CNT часов RTC<br>И конфигурации RTC<br>0 – нет сбоя<br>1 – есть сбой |
| 7 | CLR_ERRx | WO | 0 | Бит сброса флагов ошибки в данном канале<br>0 – нет сброса.<br>1 – сброс флагов ERR_DIVx, ERR_CNTx, ERR_CMx<br>и сброса ошибок конфигурации RTC (кроме PRL и ALRM).<br>Флаги не сбрасываются, если в момент их сброса событие сбоя продолжается.<br>После сброса флагов ошибок бит CLR_ERRx необходимо программно сбросить в 0.<br>Для сброса флага ERR_CC необходима запись единиц во все CLR_ERRx |
| 6 | WEC | RO | 0 | Флаг выполнения записи в регистр RTC_CNT:<br>0 – можно записывать в регистр RTC_CNT;<br>1 – запись в регистр RTC_CNT (RTC_CNT_TMR1-3) запрещена, идет запись |
| 5 | ALRF_IE | R/W | 0 | Флаг разрешения прерывания RTC_IF по событию ALRF<br>0 – прерывание запрещено<br>1 – прерывание разрешено |
| 4 | SECF_IE | R/W | 0 | Флаг разрешения прерывания RTC_IF по событию SECF<br>0 – прерывание запрещено<br>1 – прерывание разрешено |
| 3 | OWF_IE | R/W | 0 | Флаг разрешения прерывания RTC_IF по событию OWF<br>0 – прерывание запрещено<br>1 – прерывание разрешено |
| 2 | ALRF | R/W | 0 | Флаг возникновения события превышения основным счетчиком RTC_CNT значения в регистре RTC_ALRM<br>0 – нет события<br>1 – есть событие<br>Флаг устанавливается при<br>RTC_CNT[31:0] == RTC_ALRM[31:0] + 1<br>Сбрасывается записью 1 |
| 1 | SECF | R/W | 0 | Флаг совпадения значений регистров RTC_DIV и RTC_PRL:<br>0 – не было совпадения;<br>1 – было совпадение<br>По событиям совпадения RTC_DIV и RTC_PRL формируется частота SEC_CLK. Таким образом флаг SECF устанавливается на каждом такте частоты SEC_CLK.<br>Сбрасывается записью 1 |
| 0 | OWF | R/W | 0 | Флаг возникновения события переполнения основного счетчика RTC_CNT<br>0 – нет события<br>1 – есть событие<br>Сбрасывается записью 1 |

## 20.14 Описание регистров контроллера сторожевого таймера WDG_CNTR

Таблица 144 – Регистры котнроллера сторожевого таймера

| **Базовый Адрес** | **Название** | **Описание** |
|---|---|---|
| 0x4000_4000 | IWDG | Сторожевой таймер IWDG |
| **Смещение** | | |
| 0x0000_0000 | KEY | Регистр Ключей |
| 0x0000_0004 | PRL | Регистр периода счета сторожевого таймера |
| 0x0000_0008 | EN | Регистр разрешения работы |
| 0x0000_000C | CNT | Значение счетчика сторожевого таймера |

### 20.14.1 *KEY*

| Base ADDR= | 0x4000_4000 | Offset= | 0x0000_0000 | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| KEY[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | KEY[31:0] | 0 | Регистр управляющих ключей блока сторожевого таймера<br>0x05555 – значение разрешающее запись в регистр периода PRLсчета сторожевого таймера<br>0x0CCCC – значение, разрешающее запись в регистр разрешения работы EN сторожевого таймера<br>Запись 0x0AAAA – перезапускает счет сторожевого таймера<br><br>Отличные от указанных выше значения не имеют значения |

### 20.14.2 *PRL*

| Base ADDR= | 0x4000_4000 | Offset= | 0x0000_0004 | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PRL[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PRL[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|

| | | | |
|---|---|---|---|
| 31…0 | PRL[31:0] | 0 | Регистр периода PRL счета сторожевого таймера<br>Определяет период работы сторожевого таймера.<br>Счетчик сторожевого таймера CNT считает от значения PRL до 0. При достижении нулевого значения в счетчике CNT формируется сигнал сброса IWDG_RESET<br>Счетчик CNT считает на частоте LSI-генератора.<br>При значении PRL = 0xFFдолжен перезупускаться не реже чем раз в 2,5 мс<br>При значении PRL = 0xFFFFдолжен перезапускаться не реже чем раз в 655 мс<br>При значении PRL = 0xFFFFFFFFдолжен перезапускаться не реже чем раз в 10 часов |

### 20.14.3 *EN*

| Base ADDR= | | 0x4000_4000 | | | Offset= | | 0x0000_0008 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| EN[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EN[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | EN[31:0] | 0 | Значение разрешения работы счетчика<br>0x03333 – счетчик работает<br><br>Любые отличные значения останавливают счетчик |

### 20.14.4 *CNT*

| Base ADDR= | | 0x4000_4000 | | | Offset= | | 0x0000_000C | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CNT[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CNT[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | CNT[31:0] | | Текущее значение счетчика CNT<br>При чтении надо учитывать, что счетчик CNT работает на частоте LSI (~40 кГц) и при высокой частоте работы ядра может потребоваться дополнительное время для обновления счетчика |

## 20.15 Описание регистров контроллера прямого доступа в память DMA_CNTR

Таблица 145 – Регистры контроллера прямого доступа в память

| Смещение отн. базового адреса | Наименование | Тип | Значение по сбросу | Описание |
|---|---|---|---|---|
| 0xE004_2000 | DMA_CNTR | | | Контроллер DMA |
| 0x000 | STATUS | RO | 0x-0nn0000*) | Статусный регистр DMA |
| 0x004 | CFG | WO | - | Регистр конфигурации DMA |
| 0x008 | CNTR_BASE_PTR. | R/W | 0x00000000 | Регистр базового адреса управляющих данных каналов |
| 0x00C | CNTR_BASE_PTR | RO | 0x000000nn**) | Регистр базового адреса альтернативных управляющих данных каналов |
| 0x010 | WAITONREQ_STATUS | RO | 0x00000000 | Регистр статуса ожидания запроса на обработку каналов |
| 0x014 | CHNL_SW_REQUEST | WO | - | Регистр программного запроса на обработку каналов |
| 0x018 | CHNL_USEBURST_SET | R/W | 0x00000000 | Регистр установки пакетного обмена каналов |
| 0x01C | CHNL_USEBURST_CLR | WO | - | Регистр сброса пакетного обмена каналов |
| 0x020 | CHNL_REQ_MASK_SET | R/W | 0x00000000 | Регистр маскирования запросов на обслуживание каналов |
| 0x024 | CHNL_REQ_MASK_CLR | WO | - | Регистр очистки маскирования запросов на обслуживание каналов |
| 0x028 | CHNL_ENABLE_SET | R/W | 0x00000000 | Регистр установки разрешения каналов |
| 0x02C | CHNL_ENABLE_CLR | WO | - | Регистр сброса разрешения каналов |
| 0x030 | CHNL_PRI_ALT_SET | R/W | 0x00000000 | Регистр установки первичной/альтернативной структуры управляющих данных каналов |
| 0x034 | CHNL_PRI_ALT_CLR | WO | - | Регистр сброса первичной/альтернативной структуры управляющих данных каналов |
| 0x038 | CHNL_PRIORITY_SET | R/W | 0x00000000 | . установки приоритета каналов |
| 0x03C | CHNL_PRIORITY_CLR | WO | - | Регистр сброса приоритета каналов |
| 0x040-0x048 | - | | - | Зарезервировано |
| 0x04C | ERR_CLR | R/W | 0x00000000 | Регистр сброса флага ошибки |
| 0x050 | CHMUX0 | R/W | 0x00000000 | Регистр назначения каналов 0,1,2,3 |
| 0x054 | CHMUX1 | R/W | 0x00000000 | Регистр назначения каналов 4,5,6,7 |
| 0x058 | CHMUX2 | R/W | 0x00000000 | Регистр назначения каналов 8,9,10,11 |
| 0x05C | CHMUX3 | R/W | 0x00000000 | Регистр назначения каналов 12,13,14,15 |
| 0x060 | CHMUX4 | R/W | 0x00000000 | Регистр назначения каналов 16,17,18,19 |
| 0x064 | CHMUX5 | R/W | 0x00000000 | Регистр назначения каналов 20,21,22,23 |

| Смещение отн. базового адреса | Наименование | Тип | Значение по сбросу | Описание |
|---|---|---|---|---|
| 0x068 | CHMUX6 | R/W | 0x00000000 | Регистр назначения каналов 24,25,26,27 |
| 0x06C | CHMUX7 | R/W | 0x00000000 | Регистр назначения каналов 28,29,30,31 |
| 0x070-0xDFC | - | - | | 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.

### 20.15.1 *STATUS*

Регистр STATUS является статусным регистром контроллера 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 146 – 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 147 – 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:<br>0x0 = контроллер не имеет интегрированной схемы тестирования;<br>0x1 = контроллер имеет интегрированную схему тестирования;<br>0x2 – 0xF = не определено |
| 27...21 | - | Не определено |
| 20...16 | chnls_minus1 | Количество доступных каналов DMA минус 1.<br>Например,<br>b00000 = контроллер имеет 1 канал DMA;<br>b00001 = контроллер имеет 2 канала DMA;<br>b00010 = контроллер имеет 3 канала DMA;<br>…<br>b11111 = контроллер имеет 32 канала DMA |
| 15...8 | - | Не определено |
| 7...4 | state | Текущее состояние автомата управления контроллера.<br>Состояние может быть одним из следующих:<br>b0000 = в покое;<br>b0001 = чтение управляющих данных канала;<br>b0010 = чтение указателя конца данных источника;<br>b0011 = чтение указателя конца данных приемника;<br>b0100 = чтение данных источника;<br>b0101 = запись данных в приемник;<br>b0110 = ожидание запроса на выполнение DMA;<br>b0111 = запись управляющих данных канала;<br>b1000 = приостановлен;<br>b1001 = выполнен;<br>b1010 = режим работы с периферией «Исполнение с изменением конфигурации»;<br>b1011-b1111 = не определено |
| 3...1 | - | Не определено |

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 0 | master_enable | Состояние контроллера:<br>0 = работа контроллера запрещена;<br>= работа контроллера разрешена |

### 20.15.2 *CFG*

Регистр CFG является регистром конфигурации контроллера DMA.

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

Таблица 148 – Регистр конфигурации DMA

| Номер | 31...8 | 7...5 | 4...1 | 0 |
|---|---|---|---|---|
| Доступ | U | WO | U | WO |
| Сброс | 0 | 0 | 0 | 0 |
| | - | chnl_prot_CNTR | - | master_enable |

Таблица 149 – 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_CNTR | Определяет уровни индикации сигналов HPROT[3:1] защиты шины AHB-Lite:<br>Разряд 7 управляет сигналом HPROT[3], с целью индикации о появлении доступа с кэшированием;<br>Разряд 6 управляет сигналом HPROT[2], с целью индикации о появлении доступа с буферизацией;<br>Разряд 5 управляет сигналом HPROT[1], с целью индикации о появлении привилегированного доступа.<br>*Примечания:*<br>1. Если разряд[n] = 1, то соответствующий сигнал HPROT в состоянии 1;<br>2. Если разряд[n] = 0, то соответствующий сигнал HPROT в состоянии 0 |
| 4...1 | - | Не определено. Следует записывать 0. |
| 0 | master_enable | Определяет состояние контроллера:<br>0 – запрещает работу контроллера;<br>1 – разрешает работу контроллера |

### 20.15.3 *CNTR_BASE_PTR*

Регистр CNTR_BASE_PTR является регистром 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 150 – Регистр базового адреса управляющих данных каналов

| **Номер** | 31...10 | 9...0 |
|---|---|---|
| **Доступ** | R/W | U |
| **Сброс** | 0 | 0 |
| | CNTR_base_ptr | - |

Таблица 151 – Назначение разрядов регистра CNTR_base_ptr

| **№ бита** | **Функциональное имя бита** | **Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений** |
|---|---|---|
| 31…10 | CNTR_base_ptr | Указатель на базовый адрес первичной структуры управляющих данных. См. соответствующий раздел |
| 9…0 | - | Не определено. Следует записывать 0 |

### 20.15.4 *ALT_CNTR_BASE_PTR*

Регистр ALT_CNTR_BASE_PTR является регистром 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 152 – Регистр базового адреса альтернативных управляющих данных каналов

| **Номер** | 31... 0 |
|---|---|
| **Доступ** | RO |
| **Сброс** | 0 |
| | Alt_CNTR_base_ptr |

Таблица 153 – Назначение разрядов регистра alt_CNTR_base_ptr

| **№ бита** | **Функциональное имя бита** | **Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений** |
|---|---|---|
| 31…0 | alt_CNTR_base_ptr | Указатель базового адреса альтернативной структуры управляющих данных |

### 20.15.5 *WAITONREQ_STATUS*

Регистр WAITONREQ_STATUS является регистром 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 154 – 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 155 – 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.<br>При чтении:<br>Разряд [C] =0 означает, что dma_waitonreq[C] в состоянии 0<br>Разряд [C] =1 означает, что dma_waitonreq[C] в состоянии 1 |

### 20.15.6 *CHNL_SW_REQUEST*

Регистр CHNL_SW_REQUEST является регистром программного запроса на обработку каналов.

Данный регистр имеет доступ только на запись. Регистр позволяет устанавливать программно запрос на выполнение цикла DMA.

Таблица 156 – 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 157 – 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.<br>При записи:<br>Разряд [C] = 0 означает, что запрос на выполнение цикла DMA по каналу C не будет установлен;<br>Разряд [C] =1 означает, что запрос на выполнение цикла DMA по каналу C будет установлен.<br>Запись разряда, соответствующего нереализованному каналу, означает, что запрос на выполнение цикла DMA не будет установлен |

### 20.15.7 *CHNL_USEBURST_SET*

Регистр CHNL_USEBURST_SET является регистром 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 158 – 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 159 – 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.<br>При чтении:<br>Разряд [C] =0 означает, что канал DMA C выполняет циклы DMA в ответ на запросы, полученные от dma_sreq[] и dma_req[].<br>Контроллер выполняет одиночные передачи или $2^R$ передач.<br>Разряд [C] =1 означает, что канал DMA C выполняет циклы DMA в ответ на запросы, полученные только от dma_req[].<br>Контроллер выполняет $2^R$ передач.<br>При записи:<br>Разряд [C] =0 не дает эффекта. Необходимо использовать chnl_useburst_clr регистр и установить соответствующий разряд C в 0;<br>Разряд [C] =1 отключает возможность обрабатывать запросы на выполнение циклов DMA, полученные от dma_sreq[].<br>Контроллер выполняет $2^R$ передач.<br>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.

### 20.15.8 *CHNL_USEBURST_CLR*

Регистр CHNL_USEBURST_CLR является регистром сброса пакетного обмена каналов.

Данный регистр имеет доступ только на запись. Регистр разрешает выполнение одиночных запросов по установке dma_sreq[].

Таблица 160 – 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 161 – 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[].<br>При записи:<br>Разряд [C] = 0 не дает эффекта. Необходимо использовать chnl_useburst_set регистр для отключения обработки запросов от dma_sreq[];<br>Разряд [C] = 1 разрешает обрабатывать запросы на выполнение циклов DMA, полученные от dma_sreq[].<br>Запись разряда, соответствующего нереализованному каналу, не дает никакого эффекта |

### 20.15.9 *CHNL_REQ_MASK_SET*

Регистр CHNL_REQ_MASK_SET является регистром 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 162 – 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 163 – 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.<br>При чтении:<br>Разряд [C] = 0 означает, что канал DMA C выполняет циклы DMA в ответ на поступающие запросы;<br>Разряд [C] = 1 означает, что канал DMA C не выполняет циклы DMA в ответ на поступающие запросы.<br>При записи:<br>Разряд [C] = 0 не дает эффекта. Необходимо использовать chnl_req_mask_clr регистр для разрешения установки запросов;<br>Разряд [C] = 1 отключает установку запросов на выполнение циклов DMA, по dma_sreq[] и dma_req[].<br>Запись разряда, соответствующего нереализованному каналу, не дает никакого эффекта |

### 20.15.10 *CHNL_REQ_MASK_CLR*

Регистр CHNL_REQ_MASK_CLR является регистром 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 164 – 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 165 – 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.<br>При записи:<br>Разряд [C] =0 не дает эффекта. Необходимо использовать chnl_req_mask_set регистр для отключения установки запросов;<br>Разряд [C] =1 разрешает установку запросов на выполнение циклов DMA, по dma_sreq[] и dma_req[].<br>Запись разряда, соответствующего нереализованному каналу, не дает никакого эффекта |

### 20.15.11 *CHNL_ENABLE_SET*

Регистр CHNL_ENABLE_SET является регистром 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 166 – 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 167 – 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.<br>При чтении:<br>Разряд [C] = 0 означает, что канал DMA C отключен;<br>Разряд [C] = 1 означает, что работа канала DMA C разрешена.<br>При записи:<br>Разряд [C] = 0 не дает эффекта. Необходимо использовать chnl_enable_clr регистр для отключения канала;<br>Разряд [C] = 1 разрешает работу канала DMA C.<br>Запись разряда, соответствующего нереализованному каналу, не дает никакого эффекта |

### 20.15.12 *CHNL_ENABLE_CLR*

Регистр CHNL_ENABLE_CLR является регистром сброса разрешения каналов.

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

Таблица 168 – 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 169 – 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.<br>При записи:<br>Разряд [C] = 0 не дает эффекта. Необходимо использовать chnl_enable_set регистр для разрешения работы канала;<br>Разряд [C] = 1 запрещает работу канала DMA C.<br>Запись разряда, соответствующего нереализованному каналу, не дает никакого эффекта.<br>*Примечание* – Контроллер может отключить канал DMA, установив соответствующий разряд в следующих случаях:<br>- при завершении цикла DMA;<br>- при чтении из channel_cfg с полем cycle_cntr установленным в b000;<br>- при появлении ошибки на шине AHB-Lite |

### 20.15.13 *CHNL_PRI_ALT_SET*

Регистр CHNL_PRI_ALT_SET является регистром 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 170 – 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 171 – 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.<br>При чтении:<br>Разряд [C] = 0 означает, что канал DMA C использует первичную структуру управляющих данных;<br>Разряд [C] = 1 означает, что канал DMA C использует альтернативную структуру управляющих данных.<br>При записи:<br>Разряд [C] = 0 не дает эффекта. Необходимо использовать chnl_pri_alt_clr регистр для сброса разряда [C] в 0;<br>Разряд [C] = 1 подключает использование альтернативной структуры управляющих данных каналом DMA C.<br>Запись разряда, соответствующего нереализованному каналу, не дает никакого эффекта.<br>*Примечание* – Контроллер может переключить значение разряда chnl_pri_alt_set[C] в следующих случаях:<br>- 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»;<br>- при завершении всех передач DMA указанных в первичной структуре управляющих данных при выполнении цикла DMA в режиме «Пинг-понг»;<br>- при завершении всех передач DMA указанных в альтернативной структуре управляющих данных при выполнении цикла DMA в режимах:<br>- «пинг-понг»;<br>- работа с памятью «Исполнение с изменением конфигурации»;<br>- работа с периферией «Исполнение с изменением конфигурации |

### 20.15.14 *CHNL_PRI_ALT_CLR*

Регистр CHNL_PRI_ALT_CLR является регистром 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 172 – 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 173 – 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.<br>При записи:<br>Разряд [C] = 0 не дает эффекта. Необходимо использовать chnl_pri_alt_set регистр для выбора альтернативных управляющих данных;<br>Разряд [C] = 1 подключает использование первичной структуры управляющих данных каналом DMA C.<br>Запись разряда, соответствующего нереализованному каналу, не дает никакого эффекта.<br>*Примечание* – Контроллер может переключить значение разряда chnl_pri_alt_clr[C] в следующих случаях:<br>- 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»;<br>- при завершении всех передач DMA указанных в первичной структуре управляющих данных при выполнении цикла DMA в режиме «пинг-понг»;<br>- при завершении всех передач DMA указанных в альтернативной структуре управляющих данных при выполнении цикла DMA в режимах:<br>- «пинг-понг»;<br>- работа с памятью «Исполнение с изменением конфигурации»;<br>- работа с периферией «Исполнение с изменением конфигурации» |

### 20.15.15 *CHNL_PRIORITY_SET*

Регистр CHNL_PRIORITY_SET является регистром 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 174 – 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 175 – 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.<br>При чтении:<br>Разряд [C] = 0 означает, что каналу DMA с присвоен уровень приоритета по умолчанию;<br>Разряд [C] = 1 означает, что каналу DMA C присвоен высокий уровень приоритета.<br>При записи:<br>Разряд [C] = 0 не дает эффекта. Необходимо использовать chnl_priority_clr регистр для установки каналу C уровня приоритета по умолчанию;<br>Разряд [C] = 1 устанавливает каналу DMA C высокий уровень приоритета.<br>Запись разряда, соответствующего нереализованному каналу, не дает никакого эффекта |

### 20.15.16 *CHNL_PRIORITY_CLR*

Регистр CHNL_PRIORITY_CLR является регистром сброса приоритета каналов.

Данный регистр имеет доступ только на запись. Регистр позволяет присвоить каналу DMA уровень приоритета по умолчанию.

Таблица 176 – 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 177 – 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.<br>При записи:<br>Разряд [C] = 0 не дает эффекта. Необходимо использовать chnl_priority_set регистр для установки каналу C высокого уровня приоритета.<br>Разряд [C] = 1 устанавливает каналу DMA C уровень приоритета по умолчанию.<br>Запись разряда, соответствующего нереализованному каналу, не дает никакого эффекта |

### 20.15.17 *ERR_CLR*

Регистр ERR_CLR является регистром 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 178 – Регистр сброса флага ошибки

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

Таблица 179 – 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.<br>При чтении:<br>Разряд [C] = 0 означает, что dma_err находится в состоянии 0;<br>Разряд [C] = 1 означает, что dma_err находится в состоянии 1.<br>При записи:<br>Разряд [C] =0 не дает эффекта. Состояние dma_err останется неизменным;<br>Разряд [C] =1 сбрасывает сигнал (флаг) dma_err в состояние 0.<br>*Примечание* – При сбросе сигнала dma_err одновременно с появлением ошибки на шине AHB-Lite, то приоритет отдается ошибке, и, следовательно, значение регистра (и dma_err) останется неизменным (несброшенным) |

### 20.15.18 *CHMUX[x]*

Таблица 180 – Регистр мультиплексора каналов

| Base ADDR= | 0xE004_2000 | Offset= | 0x050<br>0x054<br>0x058<br>0x05C<br>0x060<br>0x064<br>0x068<br>0x06C |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CHANEL[x•4+3] | | | | | | | | CHANEL[x•4+2] | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CHANEL[x•4+1] | | | | | | | | CHANEL[x•4+0] | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 181 – Назначение разрядов регистра мультиплексора каналов

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…24 | CHANEL[x•4+3] | 0 | Номер виртуального канала запроса DMA назначенного на реальный канал x•4+3 |
| 23…16 | CHANEL[x•4+2] | 0 | Номер виртуального канала запроса DMA назначенного на реальный канал x•4+2 |
| 15…8 | CHANEL[x•4+1] | 0 | Номер виртуального канала запроса DMA назначенного на реальный канал x•4+1 |
| 7…0 | CHANEL[x•4+0] | 0 | Номер виртуального канала запроса DMA назначенного на реальный канал x•4+0 |

### 20.15.19 Структуры управления DMA

Область памяти под названием channel_cfg обеспечивает управление каждой передачей DMA.

Таблица 182 – Значения разрядов dst_data_end_ptr

| **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_cntr | src_prot_cntr | R_power | n_minus_1 | next_useburst | cycle_cntr |

Таблица 183 – Назначение разрядов channel_cfg

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31…30 | dst_src | Шаг инкремента адреса приемника.<br>Шаг инкремента адреса зависит от разрядности данных источника.<br>Разрядность данных источника = байт:<br>b00 = байт;<br>b01 = полуслово (16 разрядов);<br>b10 = слово (32 разряда);<br>b11 = нет инкремента. Адрес остается равным значению области памяти dst_data_end_ptr.<br>Разрядность данных источника = полуслово:<br>b00 = зарезервировано;<br>b01 = полуслово;<br>b10 = слово;<br>b11 = нет инкремента. Адрес остается равным значению области памяти dst_data_end_ptr.<br>Разрядность данных источника = слово:<br>b00 = зарезервировано;<br>b01 = зарезервировано;<br>b10 = слово (32 разряда);<br>b11 = нет инкремента. Адрес остается равным значению области памяти dst_data_end_ptr |
| 29…28 | dst_size | Размерность данных приемника<br>*Примечание* – Значение этого поля должно быть равно значению поля src_size |
| 27…26 | src_inc | Шаг инкремента адреса источника.<br>Шаг инкремента адреса зависит от разрядности данных источника.<br>Разрядность данных источника = байт:<br>b00 = байт;<br>b01 = полуслово;<br>b10 = слово (32 разряда);<br>b11 = нет инкремента. Адрес остается равным значению области памяти src_data_end_ptr.<br>Разрядность данных источника = полуслово:<br>b00 = зарезервировано<br>b01 = полуслово<br>b10 = слово<br>b11 = нет инкремента. Адрес остается равным значению области памяти src_data_end_ptr.<br>Разрядность данных источника = слово:<br>b00 = зарезервировано;<br>b01 = зарезервировано;<br>b10 = слово;<br>b11 = нет инкремента. Адрес остается равным значению области памяти src_data_end_ptr |
| 25…24 | src_size | b00 = байт;<br>b01 = полуслово;<br>b10 = слово (32 разряда);<br>b11 = зарезервировано |
| 23…21 | dst_prot_cntr | Задает состояние HPROT[3:1], когда контроллер записывает данные в приемник.<br>Разряд 23 управляет разрядом HPROT[3]:<br>0 = HPROT[3] в состоянии 0 и доступ не кэшируется;<br>1 = HPROT[3] в состоянии 1 и доступ кэшируется.<br>Разряд 22 управляет разрядом HPROT[2]:<br>0 = HPROT[2] в состоянии 0 и доступ не буферизуется;<br>1 = HPROT[2] в состоянии 1 и доступ буферизуется.<br>Разряд 21 управляет разрядом HPROT[1]: |

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| | | 0 = HPROT[1] в состоянии 0 и доступ непривилегированный;<br>1 = HPROT[1] в состоянии 1 и доступ привилегированный |
| 20…18 | src_prot_cntr | Задает состояние HPROT[3:1], когда контроллер считывает данные из источника.<br>Разряд 20 управляет разрядом HPROT[3]:<br>0 = HPROT[3] в состоянии 0 и доступ не кэшируется;<br>1 = HPROT[3] в состоянии 1 и доступ кэшируется.<br>Разряд 19 управляет разрядом HPROT[2]:<br>0 = HPROT[2] в состоянии 0 и доступ не буферизуется;<br>1 = HPROT[2] в состоянии 1 и доступ буферизуется.<br>Разряд 18 управляет разрядом HPROT[1]:<br>0 = HPROT[1] в состоянии 0 и доступ непривилегированный;<br>1 = HPROT[1] в состоянии 1 и доступ привилегированный |
| 17…14 | R_power | Задает количество передач DMA до выполнения контроллером процедуры арбитража.<br>Возможные значения:<br>b0000 - арбитраж производится после каждой передачи DMA;<br>b0001 - арбитраж производится после 2 передач DMA;<br>b0010 - арбитраж производится после 4 передач DMA;<br>b0011 - арбитраж производится после 8 передач DMA;<br>b0100 - арбитраж производится после 16 передач DMA;<br>b0101 - арбитраж производится после 32 передач DMA;<br>b0110 - арбитраж производится после 64 передач DMA;<br>b0111 - арбитраж производится после 128 передач DMA;<br>b1000 - арбитраж производится после 256 передач DMA;<br>b1001 - арбитраж производится после 512 передач DMA;<br>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.<br>10-разрядное число плюс 1 задает количество передач DMA.<br>Возможные значения:<br>b0000000000 = 1 передача DMA;<br>b0000000001 = 2 передачи DMA;<br>b0000000010 = 3 передачи DMA;<br>b0000000011 = 4 передачи DMA;<br>b0000000100 = 5 передач DMA;<br>b0000000101 = 6 передач DMA;<br>….<br>b1111111111 = 1024 передачи DMA.<br>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.<br>*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 $2^R$.<br>Установка next_useburst разряда определяет, будет ли контроллер дополнительно переопределять разряд chnl_useburst_set[C].<br>Если контроллер выполняет цикл DMA в режиме работы с периферией «Исполнение с изменением конфигурации», то |

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| | | после окончания цикла, использующего альтернативные управляющие данные, происходит следующее в зависимости от состояния next_useburst:<br>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.<br>1 - контроллер изменяет значение chnl_useburst_set[C] в состояние 1. Поэтому для оставшихся циклов DMA в режиме работы с периферией «Исполнение с изменением конфигурации», контроллер реагирует только на запросы по dma_req[], при выполнении циклов DMA он использует альтернативные управляющие данные. |
| 2…0 | cycle_cntr | Режим работы при выполнении цикла DMA:<br>b000 **Стоп**. Означает, что структура управляющих данных является «неправильной»;<br>b001 Основной. Контроллер должен получить новый запрос для окончания цикла DMA, перед этим он должен выполнить процедуру арбитража;<br>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;<br>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_cntr поле в состояние b001 или b010;<br>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;<br>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;<br>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;<br>b111 Режим работы с периферией «исполнение с изменением конфигурации». Смотрите соответствующий раздел.<br>При работе контроллера в данном режиме значение этого поля в альтернативной структуре управляющих данных должно быть b111 |

В начале цикла DMA или $2^R$ передачи DMA контроллер считывает значение channel_cfg из системной памяти. После выполнения $2^R$ или N передач он сохраняет обновленное значение channel_cfg в системную память.

Контроллер не поддерживает значений dst_size, отличных от значений src_size. Если контроллер обнаруживает неравные значения этих полей, он использует значение

src_size в качестве размера данных и приемника, и источника и при ближайшем обновлении поля n_minus_1, он также устанавливает значение поля dst_size, равное src_size.

После выполнения контроллером N передач, контроллер устанавливает значение поля cycle_cntr в b000, делая тем самым channel_cfg данные «неправильными». Это позволяет избежать повторения выполненной передачи DMA.

### 20.15.20 *Распределение виртуальных каналов DMA*

Таблица 184 – *Распределение виртуальных каналов DMA*

| Номер канала | Источник sreg | Источник reg | Тип | Описание |
|---|---|---|---|---|
| 0 | - | - | | Программный |
| 1 | SDIO_W | SDIO_W | | Программный |
| 2 | SDIO_R | SDIO_R | | Программный |
| 3 | CRPT_OUT_DRQ[0] | CRPT_OUT_DRQ[0] | | Запрос DMA от шлюза защищенной передачи данных |
| 4 | CRPT_OUT_DRQ[1] | CRPT_OUT_DRQ[1] | | Запрос DMA от шлюза защищенной передачи данных |
| 5 | CRPT_IN_DRQ[0] | CRPT_IN_DRQ[0] | | Запрос DMA от шлюза защищенной передачи данных |
| 6 | CRPT_IN_DRQ[1] | CRPT_IN_DRQ[1] | | Запрос DMA от шлюза защищенной передачи данных |
| 7 | ETH[0] | ETH[0] | | Запрос DMA от ETH по приему |
| 8 | ETH[1] | ETH[1] | | Запрос DMA от ETH по передаче |
| 9 | TMR0[0] | TMR0[0] | | Запрос 0 DMA таймера 0 |
| 10 | TMR1[0] | TMR1[0] | | Запрос 0 DMA таймера 1 |
| 11 | TMR2[0] | TMR2[0] | | Запрос 0 DMA таймера 2 |
| 12 | TMR3[0] | TMR3[0] | | Запрос 0 DMA таймера 3 |
| 13 | TMR0[1] | TMR0[1] | | Запрос 1 DMA таймера 0 |
| 14 | TMR1[1] | TMR1[1] | | Запрос 1 DMA таймера 1 |
| 15 | TMR2[1] | TMR2[1] | | Запрос 1 DMA таймера 2 |
| 16 | TMR3[1] | TMR3[1] | | Запрос 1 DMA таймера 3 |
| 17 | TMR0[2] | TMR0[2] | | Запрос 2 DMA таймера 0 |
| 18 | TMR1[2] | TMR1[2] | | Запрос 2 DMA таймера 1 |
| 19 | TMR2[2] | TMR2[2] | | Запрос 2 DMA таймера 2 |
| 20 | TMR3[2] | TMR3[2] | | Запрос 2 DMA таймера 3 |
| 21 | TMR0[3] | TMR0[3] | | Запрос 3 DMA таймера 0 |
| 22 | TMR1[3] | TMR1[3] | | Запрос 3 DMA таймера 1 |
| 23 | TMR2[3] | TMR2[3] | | Запрос 3 DMA таймера 2 |
| 24 | TMR3[3] | TMR3[3] | | Запрос 3 DMA таймера 3 |
| 25 | TMR0[4] | TMR0[4] | | Запрос 4 DMA таймера 0 |
| 26 | TMR1[4] | TMR1[4] | | Запрос 4 DMA таймера 1 |
| 27 | CRDC_DRQ[0] | CRDC_DRQ[0] | | Блок тригонометрических преобразований, модуль готов принять данные (входное FIFO не полно и нет активной записи в него) |
| 28 | CRDC_DRQ[1] | CRDC_DRQ[1] | | Блок тригонометрических преобразований модуль готов отдать данные (выходное FIFO не пусто) |
| 29 | SSP0 TX | SSP0 TX | | Запрос DMA от SSP0 по передаче |
| 30 | SSP1 TX | SSP1 TX | | Запрос DMA от SSP1 по передаче |
| 31 | SSP0 RX | SSP0 RX | | Запрос DMA от SSP0 по приему |
| 32 | SSP1 RX | SSP1 RX | | Запрос DMA от SSP1 по приему |
| 33 | UART0 TX | UART0 TX | | Запрос DMA от UART0 по передаче |
| 34 | UART1 TX | UART1 TX | | Запрос DMA от UART1 по передаче |
| 35 | UART2 TX | UART2 TX | | Запрос DMA от UART2 по передаче |
| 36 | UART3 TX | UART3 TX | | Запрос DMA от UART3 по передаче |

| Номер канала | Источник sreg | Источник reg | Тип | Описание |
|---|---|---|---|---|
| 37 | UART0 RX | UART0 RX | | Запрос DMA от UART0 по приему |
| 38 | UART1 RX | UART1 RX | | Запрос DMA от UART1 по приему |
| 39 | UART2 RX | UART2 RX | | Запрос DMA от UART2 по приему |
| 40 | UART3 RX | UART3 RX | | Запрос DMA от UART3 по приему |
| 41 | DAC 0 RX | DAC 0 RX | | Запрос DMA от DAC 0 |
| 42 | DAC 1 RX | DAC 1 RX | | Запрос DMA от DAC 1 |
| 43 | DAC 2 RX | DAC 2 RX | | Запрос DMA от DAC 2 |
| 44 | DAC 3 RX | DAC 3 RX | | Запрос DMA от DAC 3 |
| 45 | SHDWP_PWM9 | SHDWP_PWM9 | | Запрос DMA P_FIFO EMTY PWM 9 |
| 46 | SHDWP_PWM8 | SHDWP_PWM8 | | Запрос DMA P_FIFO EMTY PWM 8 |
| 47 | SHDWP_PWM7 | SHDWP_PWM7 | | Запрос DMA P_FIFO EMTY PWM 7 |
| 48 | SHDWP_PWM6 | SHDWP_PWM6 | | Запрос DMA P_FIFO EMTY PWM 6 |
| 49 | SHDWP_PWM5 | SHDWP_PWM5 | | Запрос DMA P_FIFO EMTY PWM 5 |
| 50 | SHDWP_PWM4 | SHDWP_PWM4 | | Запрос DMA P_FIFO EMTY PWM 4 |
| 51 | SHDWP_PWM3 | SHDWP_PWM3 | | Запрос DMA P_FIFO EMTY PWM 3 |
| 52 | SHDWP_PWM2 | SHDWP_PWM2 | | Запрос DMA P_FIFO EMTY PWM 2 |
| 53 | SHDWP_PWM1 | SHDWP_PWM1 | | Запрос DMA P FIFO EMTY PWM 1 |
| 54 | SHDWB_PWM9 | SHDWB_PWM9 | | Запрос DMA B FIFO EMTY PWM 9 |
| 55 | SHDWB_PWM8 | SHDWB_PWM8 | | Запрос DMA B FIFO EMTY PWM 8 |
| 56 | SHDWB_PWM7 | SHDWB_PWM7 | | Запрос DMA B FIFO EMTY PWM 7 |
| 57 | SHDWB_PWM6 | SHDWB_PWM6 | | Запрос DMA B FIFO EMTY PWM 6 |
| 58 | SHDWB_PWM5 | SHDWB_PWM5 | | Запрос DMA B FIFO EMTY PWM 5 |
| 59 | SHDWB_PWM4 | SHDWB_PWM4 | | Запрос DMA B FIFO EMTY PWM 4 |
| 60 | SHDWB_PWM3 | SHDWB_PWM3 | | Запрос DMA B FIFO EMTY PWM 3 |
| 61 | SHDWB_PWM2 | SHDWB_PWM2 | | Запрос DMA B FIFO EMTY PWM 2 |
| 62 | SHDWB_PWM1 | SHDWB_PWM1 | | Запрос DMA B FIFO EMTY PWM 1 |
| 63 | SHDWA_PWM9 | SHDWA_PWM9 | | Запрос DMA A FIFO EMTY PWM 9 |
| 64 | SHDWA_PWM8 | SHDWA_PWM8 | | Запрос DMA A FIFO EMTY PWM 8 |
| 65 | SHDWA_PWM7 | SHDWA_PWM7 | | Запрос DMA A FIFO EMTY PWM 7 |
| 66 | SHDWA_PWM6 | SHDWA_PWM6 | | Запрос DMA A FIFO EMTY PWM 6 |
| 67 | SHDWA_PWM5 | SHDWA_PWM5 | | Запрос DMA A FIFO EMTY PWM 5 |
| 68 | SHDWA_PWM4 | SHDWA_PWM4 | | Запрос DMA A FIFO EMTY PWM 4 |
| 69 | SHDWA_PWM3 | SHDWA_PWM3 | | Запрос DMA A FIFO EMTY PWM 3 |
| 70 | SHDWA_PWM2 | SHDWA_PWM2 | | Запрос DMA A FIFO EMTY PWM 2 |
| 71 | SHDWA_PWM1 | SHDWA_PWM1 | | Запрос DMA A FIFO EMTY PWM 1 |
| 72 | ENDofCONV_ADC12 | ENDofCONV_ADC12 | | Запрос DMA по завершению преобразования ADC12 |
| 73 | ENDofCONV_ADC11 | ENDofCONV_ADC11 | | Запрос DMA по завершению преобразования ADC11 |
| 74 | ENDofCONV_ADC10 | ENDofCONV_ADC10 | | Запрос DMA по завершению преобразования ADC10 |
| 75 | FIFO_LIM_ADC12 | FIFO_LIM_ADC12 | | Запрос DMA по событию достижения порога заполнения FIFO ADC12 |
| 76 | FIFO_LIM_ADC11 | FIFO_LIM_ADC11 | | Запрос DMA по событию достижения порога заполнения FIFO ADC11 |
| 77 | FIFO_LIM_ADC10 | FIFO_LIM_ADC10 | | Запрос DMA по событию достижения порога заполнения FIFO ADC10 |
| 78 | FIFOnEMPTY_ADC12 | FIFOnEMPTY_ADC12 | | Запрос DMA по событию наличия данных в FIFO ADC12 |
| 79 | FIFOnEMPTY_ADC11 | FIFOnEMPTY_ADC11 | | Запрос DMA по событию наличия данных в FIFO ADC11 |
| 80 | FIFOnEMPTY_ADC10 | FIFOnEMPTY_ADC10 | | Запрос DMA по событию наличия данных в FIFO ADC10 |
| 81 | FIFO_FULL_ADC12 | FIFO_FULL_ADC12 | | Запрос DMA по событию заполнения FIFO ADC12 |
| 82 | FIFO_FULL_ADC11 | FIFO_FULL_ADC11 | | Запрос DMA по событию заполнения FIFO ADC11 |

| Номер канала | Источник sreg | Источник reg | Тип | Описание |
|---|---|---|---|---|
| 83 | FIFO_FULL_ADC10 | FIFO_FULL_ADC10 | | Запрос DMA по событию заполнения FIFO ADC10 |
| 84 | ENDofCONV_ADC02 | ENDofCONV_ADC02 | | Запрос DMA по завершению преобразования ADC02 |
| 85 | ENDofCONV_ADC01 | ENDofCONV_ADC01 | | Запрос DMA по завершению преобразования ADC01 |
| 86 | ENDofCONV_ADC00 | ENDofCONV_ADC00 | | Запрос DMA по завершению преобразования ADC00 |
| 87 | FIFO_LIM_ADC02 | FIFO_LIM_ADC02 | | Запрос DMA по событию достижения порога заполнения FIFO ADC02 |
| 88 | FIFO_LIM_ADC01 | FIFO_LIM_ADC01 | | Запрос DMA по событию достижения порога заполнения FIFO ADC01 |
| 89 | FIFO_LIM_ADC00 | FIFO_LIM_ADC00 | | Запрос DMA по событию достижения порога заполнения FIFO ADC00 |
| 90 | FIFOnEMPTY_ADC02 | FIFOnEMPTY_ADC02 | | Запрос DMA по событию наличия данных в FIFO ADC02 |
| 91 | FIFOnEMPTY_ADC01 | FIFOnEMPTY_ADC01 | | Запрос DMA по событию наличия данных в FIFO ADC01 |
| 92 | FIFOnEMPTY_ADC00 | FIFOnEMPTY_ADC00 | | Запрос DMA по событию наличия данных в FIFO ADC00 |
| 93 | FIFO_FULL_ADC02 | FIFO_FULL_ADC02 | | Запрос DMA по событию заполнения FIFO ADC02 |
| 94 | FIFO_FULL_ADC01 | FIFO_FULL_ADC01 | | Запрос DMA по событию заполнения FIFO ADC01 |
| 95 | FIFO_FULL_ADC00 | FIFO_FULL_ADC00 | | Запрос DMA по событию заполнения FIFO ADC00 |

### 20.16 Описание регистров контроллеров портов ввода-вывода PORT_CNTR

Таблица 185 – Регистры контроллеров портов ввода-вывода

| Базовый адрес | Смещение | Название | Состояние после сброса | Описание |
|---|---|---|---|---|
| 0x40080000 | PORTA | | | |
| 0x40081000 | PORTB | | | |
| 0x40082000 | PORTC | | | |
| 0x40083000 | PORTD | | | |
| | 0x0000_0000 | KEY | | Регистр ключа, разрешающего модификацию остальных регистров |
| | 0x0000_0004 | RXTX | | Регистр данных порта |
| | 0x0000_0008 | SRXTX | | Регистр установки порта |
| | 0x0000_000C | CRXTX | | Регистр сброса порта |
| | 0x0000_0010 | SOE | | Регистр установки направления вывода порта |
| | 0x0000_0014 | COE | | Регистр сброса направления вывода порта |
| | 0x0000_0018 | SFUNC0 | | Регистр установки функции вывода порта (7-0) |
| | 0x0000_001C | SFUNC1 | | Регистр установки функции вывода порта (15-8) |
| | 0x0000_0020 | SFUNC2 | | Регистр установки функции вывода порта (23-16) |
| | 0x0000_0024 | SFUNC3 | | Регистр установки функции вывода порта (31-24) |

| Базовый адрес | Смещение | Название | Состояние после сброса | Описание |
|---|---|---|---|---|
| | 0x0000_0028 | CFUNC0 | | Регистр сброса функции вывода порта (7-0) |
| | 0x0000_002C | CFUNC1 | | Регистр сброса функции вывода порта (15-8) |
| | 0x0000_0030 | CFUNC2 | | Регистр сброса функции вывода порта (23-16) |
| | 0x0000_0034 | CFUNC3 | | Регистр сброса функции вывода порта (31-24) |
| | 0x0000_0038 | SANALOG | | Регистр установки типа вывода порта |
| | 0x0000_003C | CANALOG | | Регистр сброса типа вывода порта |
| | 0x0000_0040 | SPULLUP | | Регистр установки подтяжки к питанию вывода порта |
| | 0x0000_0044 | CPULLUP | | Регистр сброса подтяжки к питанию вывода порта |
| | 0x0000_0048 | SPULLDOWN | | Регистр установки подтяжки к земле вывода порта |
| | 0x0000_004C | CPULLDOWN | | Регистр сброса подтяжки к земле вывода порта |
| | 0x0000_0050 | SPD | | Регистр установки типа драйвера вывода порта |
| | 0x0000_0054 | CPD | | Регистр сброса типа драйвера вывода порта |
| | 0x0000_0058 | SPWR0 | | Регистр установки скорости вывода порта (15-0) |
| | 0x0000_005C | SPWR1 | | Регистр установки скорости вывода порта (31-16) |
| | 0x0000_0060 | CPWR0 | | Регистр сброса скорости вывода порта (15-0) |
| | 0x0000_0064 | CPWR1 | | Регистр сброса скорости вывода порта (31-16) |
| | 0x0000_0068 | SCL | | Регистр установки ограничения по току вывода порта |
| | 0x0000_006C | CCL | | Регистр сброса ограничения фильтра вывода порта |
| | 0x0000_0070 | SIE | | Регистр установки разрешения прерывания от вывода |
| | 0x0000_0074 | CIE | | Регистр сброса разрешения прерывания от вывода |
| | 0x0000_0078 | SIT | | Регистр установки активного уровня прерывания |
| | 0x0000_007C | CIT | | Регистр сброса активного уровня прерывания |
| | 0x0000_0080 | SIR | | Регистр установки флага прерывания |
| | 0x0000_0084 | CIR | | Регистр сброса флага прерывания |
| | 0x0000_0088 | HCUR | | Регистр флага превышения выходного тока |

### 20.16.1 *KEY*

Таблица 186 – Регистр ключа

| Base ADDR= | | 0x40080000<br>0x40081000<br>0x40082000<br>0x40083000 | | | Offset= | | 0x0000_0000 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | KEY | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| KEY[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 187 – Назначение разрядов регистра KEY

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | KEY[31:0] | 0000_0000 | При записи в регистр значения 0x8555AAA1 открывается возможность записи в другие регистры блока |

### 20.16.2 *RXTX*

Таблица 188 – Регистр данных порта

| Base ADDR= | | 0x40080000<br>0x40081000<br>0x40082000<br>0x40083000 | | | Offset= | | 0x0000_0004 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RXTX[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RXTX[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 189 – Назначение разрядов регистра RXTX

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | RXTX | | Регистр прямого доступа данных порта<br>При чтении отображает состояние на выводах порта.<br>При записи осуществляет запись в выходные триггера порта<br>RXTX[0] соответствует выводу порта с номером 0<br>RXTX[1] соответствует выводу порта с номером 1 и так далее.<br>Внимание: При выполнении операции чтения-модификация-запись возможно возникновения ошибок, например:<br>- вывод порта настроен на выдачу высокого уровня (в выходном триггере 1), но при этом нагрузка на выводе такая, что уровень стал ниже нижней границы высокого уровня, в этом случае при чтении регистра RXTX может бить считан 0. И при его последующей записи измениться значение в выходном триггере с 1 на 0. |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| | | | - при выполнении операции чтение-модификации-записи произошло переключение задачи, либо уход на обработку прерывания сразу после выполнения чтения. В другой задаче либо обработчике прерывания изменяется значение в выходных триггерах и осуществляется возврат в текущую задачу, либо выход из обработчика. И выполняется вторая часть модификация и запись, но при этом эти действия выполняются над данными, полученными до ухода и по их завершению могут быть изменены данные записанные в другой задаче, либо обработчике прерываний.<br>Для избегания такого рода проблем рекомендуется использовать биты установки и сброса значений, например, SRXTX и CRXTX и другие аналогичные |

### 20.16.3 *SRXTX*

Таблица 190 – Регистр установки порта

| Base ADDR= | 0x40080000<br>0x40081000<br>0x40082000<br>0x40083000 | Offset= | 0x0000_0008 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SRXTX[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SRXTX[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 191 – Назначение разрядов регистра SRXTX

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | SRXTX | | Регистр установки порта<br>При чтении отображает состояние выходного триггеров порта.<br>При записи осуществляет запись в выходные триггера порта<br>SRXTX[0] соответствует выводу порта с номером 0<br>SRXTX[1] соответствует выводу порта с номером 1 и так далее.<br><br>При записи 1 производится установка в 1 выходного триггера порта, при записи 0, значение выходного триггера не изменяется.<br>**Внимание!** При чтении из регистра возвращается значение записанное в выходные триггера, но при этом если вывод, например, не настроен на выход, считанное значение может отличаться от реального значения на соответствующем выводе порта |

### 20.16.4 *CRXTX*

Таблица 192 – Регистр сброса порта

| Base ADDR= | 0x40080000<br>0x40081000<br>0x40082000<br>0x40083000 | Offset= | 0x0000_000C |
|---|---|---|---|

| REG Name: | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CRXTX[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CRXTX[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 193 – Назначение разрядов регистра CRXTX

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | CRXTX | | Регистр сброса порта<br>При чтении отображает состояние выходного триггеров порта.<br>При записи осуществляет запись в выходные триггера порта<br>CRXTX[0] соответствует выводу порта с номером 0<br>CRXTX[1] соответствует выводу порта с номером 1 и так далее.<br><br>При записи 1 производится установка в 0 выходного триггера порта, при записи 0, значение выходного триггера не изменяется.<br>**Внимание!** При чтении из регистра возвращается значение, записанное в выходные триггера, но при этом, если вывод, например, не настроен на выход, считанное значение может отличаться от реального значения на соответствующем выводе порта |

### 20.16.5 *SOE*

Таблица 194 – Регистр установки направления вывода порта

| Base ADDR= | 0x40080000<br>0x40081000<br>0x40082000<br>0x40083000 | Offset= | 0x0000_0010 |
|---|---|---|---|

| REG Name: | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| SOE[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SOE[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 195 – Назначение разрядов регистра SOE

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | SOE[31:0] | | Регистр установки направления вывода порта<br>При записи 1 производится установка в 1 сигнала направления порта, при записи 0, значение выходного триггера не изменяется.<br>SOE[0] соответствует выводу порта с номером 0<br>SOE[1] соответствует выводу порта с номером 1 и так далее.<br>0 – вывод работает на вход<br>1 – вывод работает на выход<br>Значение направления, задаваемое регистром SOE и COE, имеет смысл при функционировании порта в режиме пользовательского вывода FUNC= 0000<br>При выборе других функций для вывода его направление задается в соответствии с выбранной функцией автоматически |

### 20.16.6 *COE*

Таблица 196 – Регистр сброса направления вывода порта

| Base ADDR= | 0x40080000<br>0x40081000<br>0x40082000<br>0x40083000 | Offset= | 0x0000_0014 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| COE[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| COE[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 197 – Назначение разрядов регистра COE

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | COE[31:0] | | Регистр установки направления вывода порта<br>При записи 1 производится установка в 0 сигнала направления порта, при записи 0, значение выходного триггера не изменяется.<br>COE[0] соответствует выводу порта с номером 0<br>COE[1] соответствует выводу порта с номером 1 и так далее.<br>0 – вывод работает на вход<br>1 – вывод работает на выход<br>Значение направления, задаваемое регистром SOE и COE, имеет смысл при функционировании порта в режиме пользовательского вывода FUNC= 0000<br>При выборе других функций для вывода, его направление задается в соответствии с выбранной функцией автоматически |

### 20.16.7 *SFUNCx*

Таблица 198 – Регистр установки функции вывода порта

| Base ADDR= | 0x40080000<br>0x40081000<br>0x40082000<br>0x40083000 | Offset= | 0x0000_0018<br>0x0000_001C<br>0x0000_0020<br>0x0000_0024 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FUNC7[3:0] | | | | FUNC6[3:0] | | | | FUNC5[3:0] | | | | FUNC4[3:0] | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FUNC3[3:0] | | | | FUNC2[3:0] | | | | FUNC1[3:0] | | | | FUNC0[3:0] | | | |
| | | | | | | | | | | | | | | | |

Таблица 199 – Назначение разрядов регистра SFUNCx

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…28 | FUNC7[3:0] | | Функция, выполняемая выводом<br>При записи 1 производится установка в 1 сигнала настройки порта, при записи 0, значение выходного триггера не изменяется.<br><br>0000 – пользовательский вывод<br>0001 – Функция 1<br>…<br>1111 – Функция 15<br><br>Конкретное назначение функции смотрите в описании выводов микросхемы.<br>Выбор аналоговой функции приоритетен<br><br>SFUNC0[3…0] – вывод 0<br>SFUNC0[7…4] – вывод 1<br>SFUNC0[11…8] – вывод 2<br>SFUNC0[15…12] – вывод 3<br>SFUNC0[19…16] – вывод 4<br>SFUNC0[23…20] – вывод 5<br>SFUNC0[27…24] – вывод 6<br>SFUNC0[31…28] – вывод 7<br><br>SFUNC1[3…0] – вывод 8<br>SFUNC1[7…4] – вывод 9<br>SFUNC1[11…8] – вывод 10<br>SFUNC1[15…12] – вывод 11<br>SFUNC1[19…16] – вывод 12<br>SFUNC1[23…20] – вывод 13<br>SFUNC1[27…24] – вывод 14<br>SFUNC1[31…28] – вывод 15<br><br>SFUNC2[3…0] – вывод 16<br>SFUNC2[7…4] – вывод 17<br>SFUNC2[11…8] – вывод 18<br>SFUNC2[15…12] – вывод 19<br>SFUNC2[19…16] – вывод 20<br>SFUNC2[23…20] – вывод 21<br>SFUNC2[27…24] – вывод 22<br>SFUNC2[31…28] – вывод 23 |
| 27…24 | FUNC6[3:0] | | |
| 23…20 | FUNC5[3:0] | | |
| 19…16 | FUNC4[3:0] | | |
| 15…12 | FUNC3[3:0] | | |
| 11…8 | FUNC2[3:0] | | |
| 7…4 | FUNC1[3:0] | | |
| 3…0 | FUNC0[3:0] | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| | | | SFUNC3[3…0] – вывод 24<br>SFUNC3[7…4] – вывод 25<br>SFUNC3[11…8] – вывод 26<br>SFUNC3[15…12] – вывод 27<br>SFUNC3[19…16] – вывод 28<br>SFUNC3[23…20] – вывод 29<br>SFUNC3[27…24] – вывод 30<br>SFUNC3[31…28] – вывод 31 |

### 20.16.8 *CFUNCx*

Таблица 200 – Регистр сброса функции вывода порта

| Base ADDR= | 0x40080000<br>0x40081000<br>0x40082000<br>0x40083000 | Offset= | 0x0000_0028<br>0x0000_002C<br>0x0000_0030<br>0x0000_0034 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FUNC7[3:0] | | | | FUNC6[3:0] | | | | FUNC5[3:0] | | | | FUNC4[3:0] | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FUNC3[3:0] | | | | FUNC2[3:0] | | | | FUNC1[3:0] | | | | FUNC0[3:0] | | | |
| | | | | | | | | | | | | | | | |

Таблица 201 – Назначение разрядов регистра CFUNCx

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…28 | FUNC7[3:0] | | Функция, выполняемая выводом<br>При записи 1 производится установка в 0 сигнала настройки порта, при записи 0, значение выходного триггера не изменяется.<br><br>0000 – пользовательский вывод<br>0001 – Функция 1<br>…<br>1111 – Функция 15<br><br>Конкретное назначение функции смотрите в описании выводов микросхемы<br><br>CFUNC0[3…0] – вывод 0<br>CFUNC0[7…4] – вывод 1<br>CFUNC0[11…8] – вывод 2<br>CFUNC0[15…12] – вывод 3<br>CFUNC0[19…16] – вывод 4<br>CFUNC0[23…20] – вывод 5<br>CFUNC0[27…24] – вывод 6<br>CFUNC0[31…28] – вывод 7<br><br>CFUNC1[3…0] – вывод 8<br>CFUNC1[7…4] – вывод 9<br>CFUNC1[11…8] – вывод 10<br>CFUNC1[15…12] – вывод 11<br>CFUNC1[19…16] – вывод 12<br>CFUNC1[23…20] – вывод 13<br>CFUNC1[27…24] – вывод 14<br>CFUNC1[31…28] – вывод 15 |
| 27…24 | FUNC6[3:0] | | |
| 23…20 | FUNC5[3:0] | | |
| 19…16 | FUNC4[3:0] | | |
| 15…12 | FUNC3[3:0] | | |
| 11…8 | FUNC2[3:0] | | |
| 7…4 | FUNC1[3:0] | | |
| 3…0 | FUNC0[3:0] | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| | | | CFUNC2[3…0] – вывод 16<br>CFUNC2[7…4] – вывод 17<br>CFUNC2[11…8] – вывод 18<br>CFUNC2[15…12] – вывод 19<br>CFUNC2[19…16] – вывод 20<br>CFUNC2[23…20] – вывод 21<br>CFUNC2[27…24] – вывод 22<br>CFUNC2[31…28] – вывод 23<br><br>CFUNC3[3…0] – вывод 24<br>CFUNC3[7…4] – вывод 25<br>CFUNC3[11…8] – вывод 26<br>CFUNC3[15…12] – вывод 27<br>CFUNC3[19…16] – вывод 28<br>CFUNC3[23…20] – вывод 29<br>CFUNC3[27…24] – вывод 30<br>CFUNC3[31…28] – вывод 31 |

### 20.16.9 *SANALOG*

Таблица 202 – Регистр установки типа вывода порта

| Base ADDR= | | 0x40080000<br>0x40081000<br>0x40082000<br>0x40083000 | | | Offset= | | 0x0000_0038 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| SANALOG[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SANALOG [15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 203 – Назначение разрядов регистра SANALOG

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | SANALOG[31:0] | | Регистр установки типа вывода порта<br>При записи 1 производится установка в 1 сигнала настройки порта, при записи 0, значение выходного триггера не изменяется.<br>SANALOG[0] соответствует выводу порта с номером 0<br>SANALOG[1] соответствует выводу порта с номером 1 и так далее.<br>0 – вывод выполняет аналоговую функцию<br>1 – вывод выполняет цифровую функцию<br>При выборе аналоговой функции выключается вход вывода микросхемы, и из RXTX всегда будет считана 1 в соответствующем разряде |

### 20.16.10 *CANALOG*

Таблица 204 – Регистр сброса типа вывода порта

| Base ADDR= | | 0x40080000<br>0x40081000<br>0x40082000<br>0x40083000 | | | Offset= | | 0x0000_003C | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CANALOG[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CANALOG [15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 205 – Назначение разрядов регистра CANALOG

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | CANALOG[31:0] | | Регистр сброса типа вывода порта<br>При записи 1 производится установка в 0 сигнала настройки порта, при записи 0, значение выходного триггера не изменяется.<br>CANALOG[0] соответствует выводу порта с номером 0<br>CANALOG[1] соответствует выводу порта с номером 1 и так далее.<br>0 – вывод выполняет аналоговую функцию<br>1 – вывод выполняет цифровую функцию<br>При выборе аналоговой функции выключается вход вывода микросхемы, и из RXTX всегда будет считана 1 в соответствующем разряде |

### 20.16.11 *SPULLUP*

Таблица 206 – Регистр установки подтяжки к питанию вывода порта

| Base ADDR= | | 0x40080000<br>0x40081000<br>0x40082000<br>0x40083000 | | | Offset= | | 0x0000_0040 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| SPULLUP[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SPULLUP [15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 207 – Назначение разрядов регистра SPULLUP

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | SPULLUP[31:0] | | Регистр установки подтяжки к питанию вывода порта<br>При записи 1 производится установка в 1 сигнала настройки порта, при записи 0, значение выходного триггера не изменяется.<br>SPULLUP[0] соответствует выводу порта с номером 0<br>SPULLUP[1] соответствует выводу порта с номером 1 и так далее.<br>0 – нет подтяжки<br>1 – есть подтяжка к питанию Ucc через резистор $R_{PULLUP}$ с тип. значением 50 кОм |

### 20.16.12 *CPULLUP*

Таблица 208 – Регистр сброса подтяжки к питанию вывода порта

| Base ADDR= | 0x40080000<br>0x40081000<br>0x40082000<br>0x40083000 | Offset= | 0x0000_0044 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CPULLUP[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CPULLUP [15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 209 – Назначение разрядов регистра CPULLUP

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | CPULLUP[31:0] | | Регистр сброса подтяжки к питанию вывода порта<br>При записи 1 производится установка в 0 сигнала настройки порта, при записи 0, значение выходного триггера не изменяется.<br>SPULLUP[0] соответствует выводу порта с номером 0<br>SPULLUP[1] соответствует выводу порта с номером 1 и так далее.<br>0 – нет подтяжки<br>1 – есть подтяжка к питанию Uссчерез резистор $R_{PULLUP}$ с тип. значением 50 кОм |

### 20.16.13 *SPULLDOWN*

Таблица 210 – Регистр установки подтяжки к земле вывода порта

| Base ADDR= | 0x40080000<br>0x40081000<br>0x40082000<br>0x40083000 | Offset= | 0x0000_0048 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SPULLDOWN[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SPULLDOWN [15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 211 – Назначение разрядов регистра SPULLDOWN

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | SPULLDOWN[31:0] | | Регистр установки подтяжки кземле вывода порта<br>При записи 1 производится установка в 1 сигнала настройки порта, при записи 0, значение выходного триггера не изменяется.<br>SPULLDOWN[0] соответствует выводу порта с номером 0<br>SPULLDOWN[1] соответствует выводу порта с номером 1 и так далее.<br>0 – нет подтяжки<br>1 – есть подтяжка к земле через резистор $R_{PULLDOWN}$ с тип. значением 50 кОм |

### 20.16.14 *CPULLDOWN*

Таблица 212 – Регистр сброса подтяжки к земле вывода порта

| Base ADDR= | 0x40080000<br>0x40081000<br>0x40082000<br>0x40083000 | Offset= | 0x0000_004C |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CPULLDOWN[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CPULLDOWN [15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 213 – Назначение разрядов регистра CPULLDOWN

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | CPULLDOWN[31:0] | | Регистр сброса подтяжки кземле вывода порта<br>При записи 1 производится установка в 0 сигнала настройки порта, при записи 0, значение выходного триггера не изменяется.<br>SPULLDOWN[0] соответствует выводу порта с номером 0<br>SPULLDOWN[1] соответствует выводу порта с номером 1 и так далее.<br>0 – нет подтяжки<br>1 – есть подтяжка к земле через резистор $R_{PULLDOWN}$ с тип. значением 50 кОм |

### 20.16.15 *SPD*

Таблица 214 – Регистр установки типа драйвера вывода порта

| Base ADDR= | 0x40080000<br>0x40081000<br>0x40082000<br>0x40083000 | Offset= | 0x0000_0050 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SPD[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SPD [15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 215 – Назначение разрядов регистра SPD

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | SPD [31:0] | | Регистр установки типа драйвера вывода порта<br>При записи 1 производится установка в 1 сигнала настройки порта, при записи 0, значение выходного триггера не изменяется.<br>SPD[0] соответствует выводу порта с номером 0<br>SPD[1] соответствует выводу порта с номером 1 и так далее.<br>0 – драйвер с активным высоким и низким уровнями<br>1 – драйвер с активным и низким уровнем (открытый сток) |

### 20.16.16 *CPD*

Таблица 216 – Регистр сброса типа драйвера вывода порта

| Base ADDR= | 0x40080000<br>0x40081000<br>0x40082000<br>0x40083000 | Offset= | 0x0000_0054 |
|---|---|---|---|

REG Name:

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CPD[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CPD [15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 217 – Назначение разрядов регистра CPD

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | CPD [31:0] | | Регистр сброса типа драйвера вывода порта<br>При записи 1 производится установка в 0 сигнала настройки порта, при записи 0, значение выходного триггера не изменяется.<br>CPD[0] соответствует выводу порта с номером 0<br>CPD[1] соответствует выводу порта с номером 1 и так далее.<br>0 – драйвер с активным высоким и низким уровнями<br>1 – драйвер с активным и низким уровнем (открытый сток) |

### 20.16.17 *SPWRx*

Таблица 218 – Регистр установки скорости вывода порта

| Base ADDR= | 0x40080000<br>0x40081000<br>0x40082000<br>0x40083000 | Offset= | 0x0000_0058<br>0x0000_005C |
|---|---|---|---|

REG Name:

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PWR15 | | PWR14 | | PWR13 | | PWR12 | | PWR11 | | PWR10 | | PWR9 | | PWR8 | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PWR7 | | PWR6 | | PWR5 | | PWR4 | | PWR3 | | PWR2 | | PWR1 | | PWR0 | |
| | | | | | | | | | | | | | | | |

Таблица 219 – Назначение разрядов регистра SPWRx

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…30 | PWR0[1:0] | | Регистр задания скорости драйвера<br>При записи 1 производится установка в 1 сигнала настройки порта, при записи 0, значение выходного триггера не изменяется.<br>00 – драйвер выключен<br>01 – медленный фронт 300 нс<br>10 – средний фронт 100 нс<br>11 – бытстрый фронт 10 нс<br><br>SPWR0[1:0] – вывод 0 |
| 29…28 | PWR0[1:0] | | |
| 27…26 | PWR0[1:0] | | |
| 25…24 | PWR0[1:0] | | |
| 23…22 | PWR0[1:0] | | |
| 21…20 | PWR0[1:0] | | |
| 19…18 | PWR0[1:0] | | |
| 17…16 | PWR0[1:0] | | |
| 15…14 | PWR0[1:0] | | |
| 13…12 | PWR0[1:0] | | |

| | | | |
|---|---|---|---|
| 11…10 | PWR0[1:0] | | SPWR0[3:2] – вывод 1<br>SPWR0[5:4] – вывод 2<br>SPWR0[7:6] – вывод 3<br>SPWR0[9:8] – вывод 4<br>SPWR0[11:10] – вывод 5<br>SPWR0[13:12] – вывод 6<br>SPWR0[15:14] – вывод 7<br>SPWR0[17:16] – вывод 8<br>SPWR0[19:18] – вывод 9<br>SPWR0[21:20] – вывод 10<br>SPWR0[23:22] – вывод 11<br>SPWR0[25:24] – вывод 12<br>SPWR0[27:26] – вывод 13<br>SPWR0[29:28] – вывод 14<br>SPWR0[31:30] – вывод 15<br><br>SPWR1[1:0] – вывод 16<br>SPWR1[3:2] – вывод 17<br>SPWR1[5:4] – вывод 18<br>SPWR1[7:6] – вывод 19<br>SPWR1[9:8] – вывод 20<br>SPWR1[11:10] – вывод 21<br>SPWR1[13:12] – вывод 22<br>SPWR1[15:14] – вывод 23<br>SPWR1[17:16] – вывод 24<br>SPWR1[19:18] – вывод 25<br>SPWR1[21:20] – вывод 26<br>SPWR1[23:22] – вывод 27<br>SPWR1[25:24] – вывод 28<br>SPWR1[27:26] – вывод 29<br>SPWR1[29:28] – вывод 30<br>SPWR1[31:30] – вывод 31 |
| 9…8 | PWR0[1:0] | | |
| 7…6 | PWR0[1:0] | | |
| 5...4 | PWR0[1:0] | | |
| 3…2 | PWR0[1:0] | | |
| 1…0 | PWR0[1:0] | | |

### 20.16.18 *CPWRx*

Таблица 220 – Регистр сброса скорости вывода порта

| Base ADDR= | | 0x40080000<br>0x40081000<br>0x40082000<br>0x40083000 | | Offset= | | 0x0000_0060<br>0x0000_0064 | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PWR15 | | PWR14 | | PWR13 | | PWR12 | | PWR11 | | PWR10 | | PWR9 | | PWR8 | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PWR7 | | PWR6 | | PWR5 | | PWR4 | | PWR3 | | PWR2 | | PWR1 | | PWR0 | |
| | | | | | | | | | | | | | | | |

Таблица 221 – Назначение разрядов регистра CPWRx

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…30 | PWR0[1:0] | | Регистр сброса скорости драйвера<br>При записи 1 производится установка в 0 сигнала настройки порта, при записи 0, значение выходного триггера не изменяется.<br>00 – драйвер выключен<br>01 – медленный фронт 300 нс<br>10 – средний фронт 100 нс<br>11 – бытстрый фронт 10 нс |
| 29…28 | PWR0[1:0] | | |
| 27…26 | PWR0[1:0] | | |
| 25…24 | PWR0[1:0] | | |
| 23…22 | PWR0[1:0] | | |
| 21…20 | PWR0[1:0] | | |
| 19…18 | PWR0[1:0] | | |
| 17…16 | PWR0[1:0] | | |
| 15…14 | PWR0[1:0] | | |

| | | | |
|---|---|---|---|
| 13…12 | PWR0[1:0] | | CPWR0[1:0] – вывод 0<br>CPWR0[3:2] – вывод 1<br>CPWR0[5:4] – вывод 2<br>CPWR0[7:6] – вывод 3<br>CPWR0[9:8] – вывод 4<br>CPWR0[11:10] – вывод 5<br>CPWR0[13:12] – вывод 6<br>CPWR0[15:14] – вывод 7<br>CPWR0[17:16] – вывод 8<br>CPWR0[19:18] – вывод 9<br>CPWR0[21:20] – вывод 10<br>CPWR0[23:22] – вывод 11<br>CPWR0[25:24] – вывод 12<br>CPWR0[27:26] – вывод 13<br>CPWR0[29:28] – вывод 14<br>CPWR0[31:30] – вывод 15<br><br>CPWR1[1:0] – вывод 16<br>CPWR1[3:2] – вывод 17<br>CPWR1[5:4] – вывод 18<br>CPWR1[7:6] – вывод 19<br>CPWR1[9:8] – вывод 20<br>CPWR1[11:10] – вывод 21<br>CPWR1[13:12] – вывод 22<br>CPWR1[15:14] – вывод 23<br>CPWR1[17:16] – вывод 24<br>CPWR1[19:18] – вывод 25<br>CPWR1[21:20] – вывод 26<br>CPWR1[23:22] – вывод 27<br>CPWR1[25:24] – вывод 28<br>CPWR1[27:26] – вывод 29<br>CPWR1[29:28] – вывод 30<br>CPWR1[31:30] – вывод 31 |
| 11…10 | PWR0[1:0] | | |
| 9…8 | PWR0[1:0] | | |
| 7…6 | PWR0[1:0] | | |
| 5...4 | PWR0[1:0] | | |
| 3…2 | PWR0[1:0] | | |
| 1…0 | PWR0[1:0] | | |

### 20.16.19 *SCL*

Таблица 222 – Регистр установки ограничения по току вывода порта

| Base ADDR= | | 0x40080000<br>0x40081000<br>0x40082000<br>0x40083000 | | | Offset= | | 0x0000_0068 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CL[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CL[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 223 – Назначение разрядов регистра SCL

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | CL[31:0] | | Регистр установки разрешения ограничения по току для вывода<br>При записи 1 производится установка в 1 сигнала настройки порта, при записи 0, значение выходного триггера не изменяется.<br>0 – нет контроля ограничения по току<br>1 – есть контроль ограничения по току<br>CL[0] соответствует выводу порта с номером 0<br>CL[1] соответствует выводу порта с номером 1 и так далее |

### 20.16.20 *CCL*

Таблица 224 – Регистр сброса ограничения фильтра вывода порта

| Base ADDR= | 0x40080000<br>0x40081000<br>0x40082000<br>0x40083000 | Offset= | 0x0000_006C |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CL[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CL[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 225 – Назначение разрядов регистра CCL

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | CL[31:0] | | Регистр сброса разрешения ограничения по току для вывода<br>При записи 1 производится установка в 0 сигнала настройки порта, при записи 0, значение выходного триггера не изменяется.<br>0 – нет контроля ограничения по току<br>1 – есть контроль ограничения по току<br>CL[0] соответствует выводу порта с номером 0<br>CL[1] соответствует выводу порта с номером 1 и так далее |

### 20.16.21 *SIE*

Таблица 226 – Регистр установки разрешения прерывания от вывода

| Base ADDR= | 0x40080000<br>0x40081000<br>0x40082000<br>0x40083000 | Offset= | 0x0000_0070 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IE[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IE[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 227 – Назначение разрядов регистра SIE

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | IE[31:0] | | Регистр установки разрешения прерывания по выводу<br>При записи 1 производится установка в 1 сигнала настройки порта, при записи 0, значение выходного триггера не изменяется.<br>0 – прерывание и фиксация прерывания запрещены<br>1 – прерывание и фиксация прерывания разрешены<br>IE[0] соответствует выводу порта с номером 0<br>IE[1] соответствует выводу порта с номером 1 и так далее |

### 20.16.22 *CIE*

Таблица 228 – Регистр сброса разрешения прерывания от вывода

| Base ADDR= | 0x40080000<br>0x40081000<br>0x40082000<br>0x40083000 | Offset= | 0x0000_0074 |
|---|---|---|---|

| REG Name: | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| IE[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IE[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 229 – Назначение разрядов регистра CIE

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | IE[31:0] | | Регистр сброса разрешения прерывания по выводу<br>При записи 1 производится установка в 0 сигнала настройки порта, при записи 0, значение выходного триггера не изменяется.<br>0 – прерывание и фиксация прерывания запрещены<br>1 – прерывание и фиксация прерывания разрешены<br>IE[0] соответствует выводу порта с номером 0<br>IE[1] соответствует выводу порта с номером 1 и так далее |

### 20.16.23 *SIT*

Таблица 230 – Регистр установки активного уровня прерывания

| Base ADDR= | 0x40080000<br>0x40081000<br>0x40082000<br>0x40083000 | Offset= | 0x0000_0078 |
|---|---|---|---|

| REG Name: | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| IT[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IT[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 231 – Назначение разрядов регистра SIT

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | IT[31:0] | | Регистр установки рабочего уровня прерывания по выводу<br>При записи 1 производится установка в 1 сигнала настройки порта, при записи 0, значение выходного триггера не изменяется.<br>0 – прерывание и фиксация прерывания по низкому уровню<br>1 – прерывание и фиксация прерывания по высокому уровню<br>Имеет смысл только при разрешении прерывания по соотвествующему выводу в регистре IE<br>IT[0] соответствует выводу порта с номером 0<br>IT[1] соответствует выводу порта с номером 1 и так далее |

### 20.16.24 *CIT*

Таблица 232 – Регистр сброса активного уровня прерывания

| Base ADDR= | 0x40080000<br>0x40081000<br>0x40082000<br>0x40083000 | Offset= | 0x0000_007C |
|---|---|---|---|

| REG Name: | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| IT[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IT[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 233 – Назначение разрядов регистра CIT

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | IT[31:0] | | Регистр сброса рабочего уровня прерывания по выводу<br>При записи 1 производится установка в 0 сигнала настройки порта, при записи 0, значение выходного триггера не изменяется.<br>0 – прерывание и фиксация прерывания по низкому уровню<br>1 – прерывание и фиксация прерывания по высокому уровню<br>Имеет смысл только при разрешении прерывания по соответствующему выводу в регистре IE<br>IT[0] соответствует выводу порта с номером 0<br>IT[1] соответствует выводу порта с номером 1 и так далее |

### 20.16.25 *SIR*

Таблица 234 – Регистр установки флага прерывания

| Base ADDR= | 0x40080000<br>0x40081000<br>0x40082000<br>0x40083000 | Offset= | 0x0000_0080 |
|---|---|---|---|

| REG Name: | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| IR[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IR[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 235 – Назначение разрядов регистра SIR

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | IR[31:0] | | Регистр установки запроса прерывания по выводу<br>При записи 1 производится установка в 1 сигнала настройки порта, при записи 0, значение выходного триггера не изменяется.<br>0 – нет запроса прерывания<br>1 – есть запрос прерывания |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| | | | Имеет смысл только при разрешении прерывания по соответствующему выводу в регистре IE<br>IR[0] соответствует выводу порта с номером 0<br>IR[1] соответствует выводу порта с номером 1 и так далее |

### 20.16.26 *CIR*

Таблица 236 – Регистр сброса флага прерывания

| Base ADDR= | 0x40080000<br>0x40081000<br>0x40082000<br>0x40083000 | Offset= | 0x0000_0084 |
|---|---|---|---|

| REG Name: | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| IR[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IR[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 237 – Назначение разрядов регистра CIR

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | IR[31:0] | | Регистр сброса запроса прерывания по выводу<br>При записи 1 производится установка в 0 сигнала настройки порта, при записи 0, значение выходного триггера не изменяется.<br>0 – нет запроса прерывания<br>1 – есть запрос прерывания<br>Имеет смысл только при разрешении прерывания по соответствующему выводу в регистре IE<br>IR[0] соответствует выводу порта с номером 0<br>IR[1] соответствует выводу порта с номером 1 и так далее |

### 20.16.27 *HCUR*

Таблица 238 – Регистр флага превышения выходного тока

| Base ADDR= | 0x40080000<br>0x40081000<br>0x40082000<br>0x40083000 | Offset= | 0x0000_0088 |
|---|---|---|---|

| REG Name: | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| HCUR[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HCUR[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 239 – Назначение разрядов регистра HCUR

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | HCUR[31:0] | | Регистр флага превышения по току по выводу<br>При записи 1 производится сброс флага, если при этом закончилось событие превышения<br>0 – нет превышения по току<br>1 – есть превышение по току<br>HCUR[0] соответствует выводу порта с номером 0<br>HCUR [1] соответствует выводу порта с номером 1 и так далее |

## 20.17 Описание регистров контроллера внешней шины EXT_BUS_CNTR

Таблица 240 – Описание регистров контроллера внешней шины

| Базовый адрес | Смещение | Название | Состояние после сброса | Описание |
|---|---|---|---|---|
| 0x4000C000 | | | | |
| | 0x0000_000 | REGION[0].CNTRL | | Настройки региона 0 |
| | 0x0000_0004 | REGION[0].ECCBASE | | Базовый адрес таблицы ECC региона 0 |
| | 0x0000_0008 | REGION[0].ECCCR | | Регистр флагов и управления счётчиками ошибок ECC региона 0 |
| | 0x0000_00 | REGION[0].ECCST | | Регистр счётчиков ошибок ECC региона 0 |
| | … | | | |
| | 0x0000_0070 | REGION[7].CNTRL | | Настройки региона 7 |
| | 0x0000_007 | REGION[7].ECCBASE | | Базовый адрес таблицы ECC региона 7 |
| | 0x0000_0078 | REGION[0].ECCCR | | Регистр флагов и управления счётчиками ошибок ECC региона 7 |
| | 0x0000_007 | REGION[7].ECCST | | Регистр статуса ошибок ECC региона 7 |
| | 0x0000_0080 | KEY | | Регистр ключа, разрешающего модификацию остальных регистров |
| | 0x0000_0084 | ECCADR | | Регистр адреса последней ошибки ECC |
| | 0x0000_0088 | ECCDATA | | Регистр данных последней ошибки ECC |
| | 0x0000_008C | ECCECC | | Регистр ECC последней ошибки ECC |
| | 0x0000_0090 | OCLKCTL | | Регистр управления частотой OCLK |

### 20.17.1 *Регистр KEY*

Таблица 241 – Регистр ключа

| Base ADDR= | | 0x4000C000 | | | Offset= | | 0x0000_0080 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | KEY | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| KEY[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| KEY[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 242 – Назначение разрядов регистра KEY

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | KEY[31:0] | 0000_0000 | При записи в регистр значения 0x8555AAA1 открывается возможность записи в другие регистры блока EXT_BUS_CNTR |

### 20.17.2 *Регистр REGION[n].CNTRL*

Таблица 243 – Регистр настройки

| Base ADDR= | 0x4000C000 | Offset= | 0x0000_0000<br>0x0000_0010<br>0x0000_0020<br>0x0000_0030<br>0x0000_0040<br>0x0000_0050<br>0x0000_0060<br>0x0000_0070 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | ECC8BIT | WS_HOLD[2:0] | | | | WS_SETUP[2:0] | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WS_ACTIVE[7:0] | | | | | | | | CPOL | ROM | MODE[1:0] | | RDYWAIT | ECCMODE | ECCEN | EN |
| | | | | | | | | | | | | | | | |

Таблица 244 – Назначение разрядов регистра REGION[n].CNTRL

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| [31:25] | - | - | Зарезервировано |
| | | | |
| 24 | ECC8BIT | | Режим работы ECC<br>1 – 8 бит ECC<br>0 – 16 бит ECC |
| 23…20 | WS_HOLD[3:0] | | Длительность фазы удержания число тактов HCLK*(WS_HOLD+1) |
| 19…16 | WS_SETUP[3:0] | | Длительность фазы предустановки число тактов HCLK*(WS_SETUP+1) |
| 15…8 | WS_ACTIVE[7:0] | | Длительность активной фазы число тактов HCLK*(WS_ACTIVE+1) |
| 7 | CPOL | | Бит полярности сигнала CLOCK в фазе ACTIVE<br>0 – передний фронт<br>1 – задний фронт |
| 6 | ROM | | Режим работы региона<br>0 – RAM<br>1 – ROM<br>При записи в регион при ROM=1 транзакция завершается с флагом ошибка на шине |
| 5…4 | MODE[1:0] | | Режим организации шины данных<br>00 – 32-х разрядная шина данных (+8 разрядов при ECCMODE=0 и ECCEN=1)<br>01 – 16-ти разрядная шина данных (при ECCEN=0 или при ECCMODE=1и ECCEN=1)<br>10 – 8-ми разрядная шина данных (при ECCEN=0 или при ECCMODE=1и ECCEN=1)<br>11 –32-х разрядная шина данных (+16 разрядов при ECCMODE=0 и ECCEN=1) |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 3 | RDYWAIT | | Режим работы с ожиданием флага готовности READY<br>0 – режим работы без ожидания готовности, длительность активной фазы определяется битами WS_ACTIVE<br>1 – режим работы с ожиданием сигнала готовности, длительность активной фазы не больше чем задана битами WS_ACTIVE и завершается при наличии высокого уровня на линии READY (Необходимо учитывать два такта пересинхронизации сигнала READY на внутренний синхросигнал) |
| 2 | ECCMODE | | Режим ECC<br>0 – параллельная организация ECC (проверочные биты расположены в разрядах шины данных DATA[39:32] или DATA[47:32] в зависимости от бита ECC8BIT)<br>1 – последовательная организация ECC (проверочные биты расположены в верхних адресах, начиная с адреса ECCBASE) |
| 1 | ECCEN | | Бит разрешения контроля ECC<br>0 – контроль ECC не выполняется<br>1 – контроль ECC выполняется |
| 0 | EN | | Бит разрешения работы региона<br>0 – регион не работает<br>1 – регион работает<br>При обращении в адресное пространство не работающего региона транзакция завершается с флагом ошибка на шине |

### 20.17.3 *Регистр REGION[n].ECCBASE*

Таблица 245 – Регистр базового адреса таблицы ECC

| Base ADDR= | 0x4000C000 | Offset= | 0x0000_0004<br>0x0000_0014<br>0x0000_0024<br>0x0000_0034<br>0x0000_0044<br>0x0000_0054<br>0x0000_0064<br>0x0000_0074 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECCBASE[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECCBASE[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 246 – Назначение разрядов регистра REGION[n].ECCBASE

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | ECCBASE[31:0] | | Базовый адрес расположения таблицы ECC в регионе, при ECCMODE=1 и ECCEN=1, иначе не имеет значения |

### 20.17.4 *Регистр REGION[n].ECCCR*

Таблица 247 – Регистр флагов и управления счётчиками ошибок ECC

| Base ADDR= | 0x4000C000 | Offset= | 0x0000_0008<br>0x0000_0018<br>0x0000_0028<br>0x0000_0038<br>0x0000_0048<br>0x0000_0058<br>0x0000_0068<br>0x0000_0078 |
|---|---|---|---|

| REG Name: | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | CLR_TCNT | FIX_TECC | TECC_IE | TECC | CLR_DCNT | CLR_SCNT | FIX_DECC | FIX_SECC | DECC_IE | SECC_IE | DECC | SECC |
| | | | | | | | | | | | | | | | |

Таблица 248 – Назначение разрядов регистра REGION[n].ECCCR

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| [31:12] | - | - | Зарезервировано |
| 11 | CLR_TCNT | | Сброс счётчика тройных ошибок<br>Запись 1 сбрасывает счётчик TECC_CNT |
| 10 | FIX_TECC | | Разрешение фиксации в регистрах ECCADR, ECCDATA, ECCECC информации о последнем обращении с тройной ошибкой<br>1 – разрешено<br>0 – запрещено |
| 9 | TECC_IE | | Разрешение прерывания при возникновении тройной ошибки ECC<br>1 – разрешено<br>0 – запрещено |
| 8 | TECC | | Флаг тройной ошибки ECC<br>1 – ошибка была<br>0 – ошибок не было<br>Сбрасывается записью 1 |
| 7 | CLR_DCNT | | Бит сброса счетчика двойных ошибок<br>Запись 1 сбрасывает счетчик DECC_CNT<br>Всегда читается как 0 |
| 6 | CLR_SCNT | | Бит сброса счетчика двойных ошибок<br>Запись 1 сбрасывает счетчик DECC_CNT<br>Всегда читается как 0 |
| 5 | FIX_DECC | | Бит разрешения фиксации в регистрах адреса и данных адреса последней ошибки при двойной ошибке<br>1 – разрешено<br>0 – запрещено |
| 4 | FIX_SECC | | Бит разрешения фиксации в регистрах адреса и данных адреса последней ошибки при одиночной ошибке<br>1 – разрешено<br>0 – запрещено |
| 3 | DECC_IE | | Бит разрешения прерывания при возникновении двойной ошибки ECC<br>0 – прерывание запрещено<br>1 – прерывание разрешено |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 2 | SECC_IE | | Бит разрешения прерывания при возникновении одиночной ошибки ECC<br>0 – прерывание запрещено<br>1 – прерывание разрешено |
| 1 | DECC | | Флаг возникновения двойной ошибки<br>0 – ошибок не было<br>1 – ошибка была<br>Сбрасывается записью 1 |
| 0 | SECC | | Флаг возникновения одиночной ошибки<br>0 – ошибок не было<br>1 – ошибка была<br>Сбрасывается записью 1 |

### 20.17.5 *Регистр REGION[n].ECCST*

Таблица 249 – Регистр счётчиков ошибок ECC

| Base ADDR= | 0x4000C000 | Offset= | 0x0000_000C<br>0x0000_001C<br>0x0000_002C<br>0x0000_003C<br>0x0000_004C<br>0x0000_005C<br>0x0000_006C<br>0x0000_007C |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SECC_CNT[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DECC_CNT[7:0] | | | | | | | | TECC_CNT[7:0] | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 250 – Назначение разрядов регистра REGION[n].ECCST

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…16 | SECC_CNT[15:0] | | Счетчик числа одиночных ошибок<br>0 – нет ошибок<br>1 – одна ошибка<br>…<br>65535 – 65535 ошибок |
| 15:8 | DECC_CNT[7:0] | | Счетчик числа двойных ошибок<br>0 – нет ошибок<br>1 – одна ошибка<br>…<br>255 – 255 ошибок |
| 7:0 | TECC_CNT[7:0] | | Счётчик числа тройных ошибок<br>0 – нет ошибок<br>255 – 255 ошибок |

### 20.17.6 *Регистр ECCADR*

Таблица 251 – Регистр адреса последней ошибки ECC

| Base ADDR= | 0x4000C000 | Offset= | 0x0000_0084 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECCADR[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECCADR[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 252 – Назначение разрядов регистра ECCADR

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | ECCADR[31:0] | | Адрес последней тройной, двойной или одиночной ошибки. Предварительно необходимо разрешить фиксацию значений в регистре ECCCR, биты FIX_SECC, FIX_DECC, FIX_TECC |

### 20.17.7 *Регистр ECCDATA*

Таблица 253 – Регистр данных последней ошибки ECC

| Base ADDR= | 0x4000C000 | Offset= | 0x0000_0088 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECCDATA[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECCDATA[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 254 – Назначение разрядов регистра ECCDATA

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | ECCDATA[31:0] | | Считанные данные при последней тройной, двойной или одиночной ошибке, без корректировки ECC. Предварительно необходимо разрешить фиксацию значений в регистре ECCCR, биты FIX_SECC, FIX_DECC, FIX_TECC |

### 20.17.8 *Регистр ECCECC*

Таблица 255 – Регистр ECC последней ошибки ECC

| Base ADDR= | | 0x4000C000 | | | Offset= | | 0x0000_008C | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ECC[15:8] | | | | | | | | ECC[7:0] | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 256 – Назначение разрядов регистра ECCECC

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| [31:16] | - | | Зарезервировано |
| 15…0 | ECC[15:0] | | Считанные ECC-биты при последней тройной, двойной или одиночной ошибке, без корректировки ECC. Предварительно необходимо разрешить фиксацию значений в регистре ECCCR, биты FIX_SECC, FIX_DECC, FIX_TECC |

### 20.17.9 *OCLKCTL*

Таблица 257 – Регистр управления частотой OCLK

| Base ADDR= | | 0x4000C000 | | | Offset= | | 0x0000_0090 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | ENOCLK | DIVOCLK[3:0] | | | |
| | | | | | | | | | | | | | | | |

Таблица 258 – Назначение разрядов регистра OCLKCTL

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 4 | ENOCLK | | Бит разрешения выдачи синхросигнала OCLK<br>0 – OCLK не формируется<br>1 – OCLK формируется |
| 3…0 | DIVOCLK[3:0] | | Биты задания коэффициент деления частоты OCLK<br>FOCLK = HCLK / ($2^{DIVOCLK+1}$) |

## 20.18 Описание регистров контроллера Ethernet ETHERNETMAC_CNTR

Таблица 259 – Регистры контроллера Ethernet

| Базовый адрес | Смещение | Название | Состояние после сброса | Описание |
|---|---|---|---|---|
| 0x2100_8000 | | | | Буфер данных контроллера Ethernet 0 32 кБ |
| 0x2100_0000 | | | | Регистры управления контроллера Ethernet 0 |
| | | | | |
| | 0x0000_0000 | DELIMETR | RW, 0x4000 | Регистр границы буферов приемника и передатчика |
| | 0x0000_0002 | MAC_ADDRESS | | Регистр индивидуального MAC-адреса |
| | 0x0000_0002 | MAC_T | RW, 0x78AB | Младшая* часть индивидуального MAC-адреса |
| | 0x0000_0004 | MAC_M | RW, 0x3456 | Средняя часть индивидуального MAC-адреса |
| | 0x0000_0006 | MAC_H | RW, 0x0012 | Старшая часть индивидуального MAC-адреса |
| | | **HASH** | | **HASH-таблица групповых адресов** |
| | 0x08 | HASH0 | RW, 0x0000 | Младшая часть HASH-таблицы |
| | 0x0A | HASH1 | RW, 0x0000 | Средняя часть HASH-таблицы |
| | 0x0C | HASH2 | RW, 0x0000 | Средняя часть HASH-таблицы |
| | 0x0E | HASH3 | RW, 0x8000 | Старшая часть HASH-таблицы |
| | | | | |
| | 0x10 | IPG | RW, 0x0060 | Регистр задания межпакетного интервала для полнодуплекного режима |
| | 0x12 | PSC | RW, 0x0050 | Регистр задания предделителя шага изменения значений BAG и JitterWnd (1 мкс при частоте 50 МГц) |
| | 0x14 | BAG | RW, 0x0200 | Регистр задания периода следования пакетов (100 мкс при частоте 50 МГц) |
| | 0x16 | JITTERWND | RW, 0x0005 | Регистр задания джиттера при передаче пакетов (5 мкс при частоте 50 МГц) |
| | 0x0000_0018 | R_CFG | RW, 0x0507 | Регистр управления приемника |
| | 0x0000_001A | X_CFG | RW, 0x01FA | Регистр управления передатчика |
| | 0x0000_001C | G_CFGl | RW, 0x30004880 | Регистр общего управления блоком |
| | 0x0000_0020 | IMR | RW, 0x0000 | Регистр маски прерываний |
| | 0x0000_00202 | IFR | RW, 0x0000 | Регистр флагов прерываний |
| | 0x0000_0024 | MDIO_CNTR | RW, 0x0000 | Регистр управления канала MDIO интерфейса MII |
| | 0x0000_0026 | MDIO_DATA | RW, 0x0000 | Регистр данных канала MDIO интерфейса MII |
| | 0x28 | R_Head | RW, 0x0000 | Указатель начала области действительных данных приемника (указывает на первое непустое слово) |

| Базовый адрес | Смещение | Название | Состояние после сброса | Описание |
|---|---|---|---|---|
| | 0x2A | X_TAIL | RW, 0x4000 | Указатель конца области действительных данных передатчика (указывает на первое пустое слово) |
| | 0x2C | R_TAIL | R, 0x0000 | Указатель начала области действительных данных приемников (указывает на первое непустое слово) |
| | 0x2E | X_HEAD | R, 0x4000 | Указатель начала области действительных данных передатчика (указывает на первое непустое слово) |
| | 0x0000_0030 | STAT | R, 0x0303 | Регистр статуса |
| | 0x32 | RCOUNTER | R, 0x0000 | Счетчик количества принятых пакетов (циклический, без насыщения) |
| | 0x0000_0034 | PHY_CONTROL | RW, | Регистр управления PHY |

* - при значениях регистров MAC_T, MAC_M, MAC_H по умолчанию будут приниматься пакеты, начинающиеся с: 0xAB, 0x78, 0x56, 0x34, 0x12, 0x00… (где 0xAB – первый байт MAC адреса назначения пакета).

### 20.18.1 *Поле управления передачи пакета*

Таблица 260

| Base ADDR= | | | | | Offset= | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | CHIPID | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Length[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Описание |
|---|---|---|
| 31…16 | - | Зарезервировано |
| 15…0 | Length[15:0] | Количество байт в пакете |

### 20.18.2 *Поле состояния передачи пакета*

Таблица 261

| Base ADDR= | | | | | Offset= | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | CHIPID | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | | | | | | | | | UR | LC | RL | RCOUNT[3:0] | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Описание |
|---|---|---|
| 31…23 | - | Зарезервировано |
| 22 | UR | Флаг опустошения буфера передатчика<br>1 – буфер передатчика пуст;<br>0 – буфер передатчика не пуст. |
| 21 | LC | Флаг индикации Latecollision во время передачи пакета<br>1 – произошла Latecollision во время передачи пакета;<br>0 – Latecollision во время передачи пакета не происходила. |
| 20 | RL | Флаг исчерпания попыток передачи пакета<br>1 – превышено разрешенное количество попыток передачи пакета;<br>0 – количество попыток передачи пакета не превысило разрешенного значения; |
| 19…16 | RCOUNT[3:0] | Число попыток передачи пакета |
| 15…0 | - | Зарезервировано |

### 20.18.3 *Поле состояния приёма пакета*

Таблица 262

| Base ADDR= | | | | | Offset= | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | CHIPID | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | | | | | UCA | BCA | MCA | SMB_ERR | CRC_ERR | DN_ERR | LEN_ERR | SF_ERR | LF_ERR | CF_ERR | PF_ERR |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Length[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Описание |
|---|---|---|
| 31…27 | | Зарезервировано |
| 26 | UCA | Признак индивидуального пакета (MAC-адрес соответствует установленному)<br>1 – MAC-адрес принятого пакета совпадает с MAC-адресом Ethernet-контроллера;<br>0 – MAC-адрес принятого пакета не совпадает с MAC-адресом Ethernet-контроллера. |
| 25 | BCA | Признак широковещательного пакета (MAC = FF_FF_FF)<br>1 – принят широковещательный пакет;<br>0 – широковещательный пакет не принят. |
| 24 | MCA | Признак группового пакета (MAC соответствует HASH)<br>1 – принят пакет, удовлетворяющий фильтрации по HASH-таблице;<br>0 – принятый пакет не удовлетворяет фильтрации по HASH-таблице или фильтрация отключена. |
| 23 | SMB_ERR | Признак наличия в пакете ошибочных nibbles<br>1 – наличие<br>0 – отсутствие |
| 22 | CRC_ERR | Признак несоответствия CRC пакета<br>1 – произошла ошибка сравнения CRC-пакета с вычисленной CRC;<br>0 – CRC-пакета и вычисленной CRC совпадают. |
| 21 | DN_ERR | Количество бит в пакете не кратно 8<br>1 – не кратно 8<br>0 – кратно 8 |
| 20 | LEN_ERR | Признак несоответствия между реальной длиной и длинной указанной в поле длины – 13,14 октеты<br>1 – несоответствие<br>0 – соответствие |
| 19 | SF_ERR | Признак недостаточной длины пакета 64 октетов<br>1 – ошибочная длина<br>0 – корректная длина |
| 18 | LF_ERR | Признак превышение длины пакета 1518 октетов<br>1 – превышение<br>0 – норма |
| 17 | CF_ERR | Признак пакета управления (фильтрация по специальным MAC и тэгам в поле длины – 13,14 – октеты)<br>1 – пакет управления<br>0 – другой пакет |
| 16 | PF_ERR | Признак пакета PAUSE<br>1 – пакет PAUSE<br>0 – другой пакет |
| 15…0 | Length[15:0] | Количество байт в пакете, включая заголовок и CRC |

### 20.18.4 *DELIMETR*

Таблица 263 – Регистр границы буферов приемника и передатчика DELIMETR

| Base ADDR= | 0x0000_0000 | Offset= | 0x0000_0000 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DELIMETR[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 264 – Назначение разрядов регистра DELIMETR

| Бит | Имя | Описание |
|---|---|---|
| 15…0 | DELIMETR[15:0] | Смещение границы между буферами приёмника и передатчика относительно базового адреса буфера данных контроллера (не путать с базовым адресом регистров контроллера) |

### 20.18.5 *MAC_T*

Таблица 265 – Младшая часть индивидуального MAC-адреса

| Base ADDR= | 0x2100_0002 | Offset= | 0x0000_0002 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MAC_T[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 266 – Назначение разрядов регистра MAC_T

| Бит | Имя | Описание |
|---|---|---|
| 15…0 | MAC_T[15:0] | Младшая часть индивидуального MAC-адреса |

### 20.18.6 *MAC_M*

Таблица 267 – Средняя часть индивидуального MAC-адреса

| Base ADDR= | 0x2100_0004 | Offset= | 0x0000_0004 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| REG Name: | KEY | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MAC_M[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 268 – Назначение разрядов регистра MAC_M

| Бит | Имя | Описание |
|---|---|---|
| 15…0 | MAC_M[15:0] | Средняя часть индивидуального MAC-адреса |

### 20.18.7 *MAC_H*

Таблица 269 – Старшая часть индивидуального MAC-адреса

| Base ADDR= | 0x2100_0006 | Offset= | 0x0000_0006 | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MAC_H[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 270 – Назначение разрядов регистра MAC_H

| Бит | Имя | Описание |
|---|---|---|
| 15…0 | MAC_H[15:0] | Старшая часть MAC-адреса |

### 20.18.8 *G_CFG*

Таблица 271 – Регистр общего управления блоком

| Base ADDR= | 0x2100_0000 | Offset= | 0x0000_001C | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | | DBGXF_EN | DBGRF_EN | - | - | - | - | - | - | - | RMII_SPEED | RMIInMII | DLB | RRST | XRST |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | RCLR_EN | BUFF_MODE[1:0] | | EXT_EN | HD_EN | DTRM_EN | PAUSE_EN | ColWnd[7:0] | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 272 – Назначение разрядов регистра G_CFG

| Бит | Имя | Описание |
|---|---|---|
| [31:30] | - | Зарезервировано |
| 29 | DBGXF_EN | Разрешение автоматического изменения указателей FIFO передатчика.<br>0 – запрещено;<br>1 – разрешено |
| 28 | DBGRF_EN | Разрешение автоматического изменения указателей FIFO приемника.<br>0 – запрещено;<br>1 – разрешено |
| [27:21] | - | Зарезервировано |
| 20 | RMII_SPEED | Выбор режима работы MII или RMII<br>0 – 10 Мбит/с<br>1 – 100 Мбит/с |
| 19 | RMIInMII | Выбор режима работы MII или RMII<br>0 - MII<br>1 - RMII |

| Бит | Имя | Описание |
|---|---|---|
| 18 | DLB | Режим КЗ<br>0 – выключен;<br>1 – включен |
| 17 | RRST | Сброс приемника.<br>0 – работает;<br>1 – сброшен |
| 16 | XRST | Сброс передатчика.<br>0 – работает;<br>1 – сброшен |
| 15 | - | Зарезервировано |
| 14 | RCLR_EN | Сброс регистров статуса<br>0 – производиться записью в регистры статуса;<br>1 – регистры статуса сбрасываются при чтении |
| [13:12] | BUFF_MODE | Режим работы буфера.<br>2'b00 – линейный режим;<br>2'b 01 – режим с автоматическим изменением указателей;<br>2'b 10 – режим FIFO;<br>2'b 11 – зарезервировано (линейный режим) |
| 11 | EXT_EN | Включение режима дополнения коротких пакетов до размера slotTime полем "Extension" (При приеме отбрасывание слова осуществляется по полю length пакета, если оно отражает длину пакета).<br>0 – выключен;<br>1 – включен |
| 10 | HD_EN | Полудуплексный режим работы.<br>0 – выключен;<br>1 – включен |
| 9 | DTRM_EN | Режим детерминированного времени доставки.<br>0 – выключен<br>1 – включен |
| 8 | PAUSE_EN | Режим автоматической обработки пакета PAUSE.<br>0 – выключен;<br>1 – включен |
| 7:0 | ColWnd[7:0] | Размер «окна коллизий» |

### 20.18.9 *X_CFG*

Таблица 273 – Регистр управления передатчика X_CFG

| Base ADDR= | | 0x2100_0000 | | | Offset= | | 0x0000_001A | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EN | - | BE | MSB1st | - | EVNT_MODE | | | PAD_EN | PRE_EN | CRC_EN | IPG_EN | RtryCnt | | | |
| | | | | | | | | | | | | | | | |

Таблица 274 – Назначение разрядов регистра X_CFG

| Бит | Имя | Описание |
|---|---|---|
| 15 | EN | Разрешение работы передатчика.<br>0 – остановлен;<br>1 – разрешена работа |
| 14 | - | Зарезервировано |
| 13 | BE | Порядок следования байт в слове передатчика:<br>0 – LittleEndian;<br>1 – BigEndian |
| 12 | MSB1st | Порядок следования бит при передаче байтов данных.<br>0 – первым передается LSB;<br>1 – первым передается MSB |
| 11 | - | Зарезервировано |
| 10…8 | EVNT_MODE[2:0] | Выбор режима работы вывода EVNT[1]:<br>3’b000 – XFIFO пуст;<br>3’b001 – XFIFO заполнен на 1/4 объёма буфера;<br>3’b010 – XFIFO наполовину полон;<br>3’b011 – XFIFO заполнен на 3/4 объёма буфера;<br>3’b100 – XFIFO полон;<br>3’b101 – отправка пакета завершена;<br>3’b110 – передатчик считал слово данных из буфера;<br>3’b111 – передатчик начал очередную попытку передачи пакета |
| 7 | PAD_EN | Дополнение пакета до минимальной длины PAD-ами.<br>0 – выключено;<br>1 – включено |
| 6 | PRE_EN | Дополнение пакета преамбулой.<br>0 – выключено;<br>1 – включено |
| 5 | CRC_EN | Дополнение пакета автоматически высчитанным CRC.<br>0 – выключено;<br>1 – включено |
| 4 | IPG_EN | Режим выдержки паузы между отправкой пакетов.<br>0 – выключен;<br>1 – включен |
| 3…0 | RtryCnt[3:0] | Максимальное количество попыток отправки пакета |

Примечание – Для применения новых управляющих настроек передатчика необходимо произвести сброс передатчика путём последовательного выполнения операций установки и сброса бита XRST в регистре G_CFG.

### 20.18.10 *R_CFG*

Таблица 275 – Регистр управления приемника R_CFG

| Base ADDR= | | 0x2100_0000 | | | Offset= | | 0x0000_0018 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EN | - | BE | MSB1st | - | EVNT_MODE | | | SF_EN | LF_EN | CF_EN | EF_EN | AC_EN | UCA_EN | BCA_EN | MCA_EN |
| | | | | | | | | | | | | | | | |

Таблица 276 – Назначение разрядов регистра R_CFG

| Бит | Имя | Описание |
|---|---|---|
| 15 | EN | Разрешение работы приемника.<br>0 – приемник остановлен;<br>1 – разрешена работа |
| 14 | - | Зарезервировано |
| 13 | BE | Порядок следования байт в слове.<br>0 – LittleEndian;<br>1 – BigEndian |
| 12 | MSB1st | Порядок следования бит при приеме байтов данных.<br>0 – первым принимается LSB;<br>1 – первым принимается MSB |
| 11 | - | Зарезервировано |
| 10…8 | EVNT_MODE[2:0] | Выбор режима работы вывода EVNT[0]:<br>3’b000 – RFIFO не пуст;<br>3’b001 – RFIFO заполнен на 1/4 объёма буфера;<br>3’b010 – RFIFO наполовину пуст;<br>3’b011 – RFIFO заполнен на 3/4 объёма буфера;<br>3’b100 – RFIFO не полон;<br>3’b101 – прием пакета завершен;<br>3’b110 – приемник положил слово данных в буфер;<br>3’b111 – приемник отбросил пакет |
| 7 | SF_EN | Разрешение приема пакетов длиной меньше минимальной.<br>0 – выключено;<br>1 – включено |
| 6 | LF_EN | Разрешение приема пакетов длиной больше максимальной.<br>0 – выключено;<br>1 – включено |
| 5 | CF_EN | Разрешение приема управляющих пакетов.<br>0 – выключено;<br>1 – включено |
| 4 | EF_EN | Разрешение приема пакетов с ошибками.<br>0 – выключено;<br>1 – включено |
| 3 | AC_EN | Прием пакетов без фильтрации MAC-адреса.<br>0 – выключен;<br>1 – включен |
| 2 | UCA_EN | Прием пакетов с MAC-адресом, указанным в регистре MAC_Address.<br>0 – выключен;<br>1 – включен |
| 1 | BCA_EN | Прием пакетов с широковезательным MAC-адресом.<br>0 – выключен;<br>1 – включен |
| 0 | MCA_EN | Прием пакетов с групповым MAC-адресом с фильтрацией по HASH-таблице.<br>0 – выключен;<br>1 – включен |

Примечание – Для применения новых управляющих настроек приёмника необходимо произвести сброс приёмника путём последовательного выполнения операций установки и сброса бита RRST в регистре G_CFG.

### 20.18.11 *IMR/IFR*

Таблица 277 – Регистр маски прерываний IMR/ Регистр флагов прерываний IFR

| Base ADDR= | | 0x2100_0000 | | | Offset= | | 0x0000_0020<br>0x0000_0022 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MII_RDY | MDIO_INT | | CRS_LOST | LC | UNDF | XF_ERR | XF_OK | SF | LF | CF | CRC_ERR | SMB_ERR | OVF | MISSED_F | RF_OK |
| | | | | | | | | | | | | | | | |

Таблица 278 – Назначение разрядов регистров IMR/IFR

| Бит | Имя | Описание |
|---|---|---|
| 15 | MII_RDY | Индикатор завершения текущей команды обмена по MDIO интерфейсу |
| 14 | MDIO_INT | индикатор наличия прерывания по MDIO интерфейсу |
| 13 | - | Зарезервировано |
| 12 | CRS_LOST | Индикатор потери несущей во время передачи в полудуплексном режиме работы |
| 11 | LC | Индикатор наличия LateCollision в линии |
| 10 | UNDF | Индикатор опустошения буфера передатчика |
| 9 | XF_ERR | Индикатор наличия ошибок при передаче пакета |
| 8 | XF_OK | Индикатор успешной отправки пакета |
| 7 | SF | Индикатор приема пакета длинной менее минимальной |
| 6 | LF | Индикатор приема пакета длинной более максимальной |
| 5 | CF | Индикатор приема управляющих пакетов |
| 4 | CRC_ERR | Индикатор наличия несовпадения CRC пакета принятых данных с CRC пакета |
| 3 | SMB_ERR | Индикатор наличия ошибок в данных при приёме пакета |
| 2 | OVF | Индикатор переполнения буфера приемника |
| 1 | MISSED_F | Индикатор потери пакета из-за отсутствия места в буфере приемника |
| 0 | RF_OK | Индикатор успешно принятого пакета |

Примечание – Индикатор в состоянии единицы означает наличие события, в нуле – отсутствие события

### 20.18.12 *STAT*

Таблица 279 – Регистр статуса STAT

| Base ADDR= | | 0x2100_0000 | | | Offset= | | 0x0000_0030 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | X_FULL | X_AFULL | X_HALF | X_AEMPY | X_EMPY | RCOUNT | | | R_FULL | R_AFULL | R_HALF | R_AEMPY | R_EMPY |
| | | | | | | | | | | | | | | | |

Таблица 280 – Назначение разрядов регистра STAT

| Бит | Имя | Описание |
|---|---|---|
| [31:13] | | |
| 12 | X_FULL | 1 – буфер передатчика полон;<br>0 – буфер передатчика не полон |
| 11 | X_AFULL | 1 – буфер передатчика почти полон;<br>0 – буфер передатчика не в состоянии почти полон |
| 10 | X_HALF | 1 – буфер передатчика полуполон;<br>0 – буфер передатчика не полуполон |
| 9 | X_AEMPTY | 1 – буфер передатчика почти пуст;<br>0 – буфер передатчика не в состоянии почти пуст |
| 8 | X_EMPTY | 1 – буфер передатчика пуст;<br>0 – буфер передатчика не пуст |
| [7:5] | R_COUNT | Количество принятых, но не считанных пакетов.<br>0..6 – количество пакетов;<br>7 – количество несчитанных пакетов >=7.<br>Инкрементируется автоматически при получении нового пакета. Декрементировать необходимо программным путем после считывания пакета. Запись любого значения декрементирует счётчик на 1 |
| 4 | R_FULL | 1 – буфер приемника полон;<br>0 – буфер приемника не полон |
| 3 | R_AFULL | 1 – буфер приемника почти полон;<br>0 – буфер приемника не в состоянии почти полон |
| 2 | R_HALF | 1 – буфер приемника полуполон;<br>0 – буфер приемника не полуполон |
| 1 | R_AEMPTY | 1 – буфер приемника почти пуст;<br>0 – буфер приемника не в состоянии почти пуст |
| 0 | R_EMPTY | 1 – буфер приемника пуст;<br>0 – начало приема пакета.<br>Бит сбрасывается в "0" при начале приема пакета и при отсутствии данных в буфере |

### 20.18.13 *MDIO_CNTR*

Таблица 281 – Регистр управления канала MDIO интерфейса MII

| Base ADDR= | 0x2100_0000 | Offset= | 0x0000_0024 |
|---|---|---|---|
| REG Name: | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RDY | PRE_EN | OP | PHY_A | | | | | DIV | | | RG_A | | | | |
| | | | | | | | | | | | | | | | |

Таблица 282 – Назначение разрядов регистра MDIO_CNTR

| Бит | Имя | Описание |
|---|---|---|
| 15 | RDY | Управление/индикатор обмена по MDIO<br>После записи команды необходимо установить в единицу для инициирования исполнения команды в регистре MDIO_CNTR после одного такта сбрасывается в ноль и снова устанавливается в единицу после завершения цикла обмена по интерфейсу MDIO. |
| 14 | PRE_EN | Режим передачи.<br>1 – с передачей преамбулы (32 бита «1»);<br>0 – без передачи преамбулы. |
| 13 | OP | Операция.<br>1 – чтение;<br>0 – запись. |
| 12:8 | PHY_A[4:0] | Адрес модуля PHY |
| 7:5 | DIV | Коэффициент деления основной частоты для работы MDIO интерфейса<br>MDC= ETH_CLK / [(DIV+1)*16] |
| 4:0 | RG_A | Номер регистра PHY |

### 20.18.14 *PHY_CONTROL*

Таблица 283 – Регистр управления PHY

| Base ADDR= | 0x0000_0034 | Offset= | 0x0000_0034 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RET1N | - | - | - | - | - | - | - | - | - | - | - | FMS | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | - | EMA | | | EMAW | | WAWL | WAWLM | | |
| | | | | | | | | | | | | | | | |

Таблица 284 – Назначение разрядов регистра PHY_CONTROL

| Бит | Имя | Описание |
|---|---|---|
| 31 | RET1N | Тестовый регистр. Не изменять. |
| 30..20 | Зарезервировано | Зарезервировано |
| 19..16 | FMS | Тестовый регистр. Не изменять. |
| 15..9 | Зарезервировано | Зарезервировано |
| 8..6 | EMA | Тестовый регистр. Не изменять. |

| Бит | Имя | Описание |
|---|---|---|
| 5..4 | EMAW | Тестовый регистр. Не изменять. |
| 3 | WAWL | Тестовый регистр. Не изменять. |
| 2…0 | WAWLM | Тестовый регистр. Не изменять. |

## 20.19 Описание регистров контроллеров CAN CAN_CNTR

Таблица 285 – Регистры контроллеров CAN

| Базовый адрес | Смещение | Название | Состояние после сброса | Описание |
|---|---|---|---|---|
| 0x4008_B000 | | CAN0_CNTR | | Контроллер CAN0 |
| 0x4008_C000 | | CAN1_CNTR | | Контроллер CAN1 |
| | 0x0000_0000 | CONTROL | | Регистр управление контроллером CAN |
| | 0x0000_0004 | STATUS | | Регистр состояния контроллера CAN |
| | 0x0000_0008 | BITTMNG | | Регистр задания скорости работы |
| | 0x0000_0010 | INT_EN | | Регистр разрешения прерываний контроллера |
| | | | | |
| | 0x0000_001C | OVER | | Регистр границы счетчика ошибок |
| | | | | |
| | 0x0000_0020 | RXID | | Регистр принятого ID сообщения |
| | 0x0000_0024 | RXDLC | | Регистр принятого DLC сообщения |
| | 0x0000_0028 | RXDATAL | | Регистр принятых данных |
| | 0x0000_002C | RXDATAH | | Регистр принятых данных |
| | | | | |
| | 0x0000_0030 | TXID | | Регистр передаваемого ID сообщения |
| | 0x0000_0034 | TXDLC | | Регистр передаваемого DLC сообщения |
| | 0x0000_0038 | TXDATAL | | Регистр передаваемых данных |
| | 0x0000_003C | TXDATAH | | Регистр передаваемых данных |
| | | | | |
| | 0x0000_0040 | BUF_CON[0] | | Регистр управления буфером 00 |
| | | … | | |
| | 0x0000_00BC | BUF_CON[31] | | Регистр управления буфером 31 |
| | | | | |
| | 0x0000_00C0 | INT_RX | | Флаги разрешения прерываний от приемных буферов |
| | 0x0000_00C4 | RX | | Флаги RX_FULL от приемных буферов |
| | 0x0000_00C8 | INT_TX | | Флаги разрешения прерываний от передающих буферов |
| | 0x0000_00CC | TX | | Флаги ~TX_REQ от передающих буферов |
| | | | | |
| | 0x0000_0200 | BUF[0].ID | | ID сообщения буфера 00 |
| | 0x0000_0204 | BUF[0].DLC | | DLC сообщения буфера 00 |
| | 0x0000_0208 | BUF[0].DATAL | | Данные сообщения буфера 00 |
| | 0x0000_020C | BUF[0].DATAH | | Данные сообщения буфера 00 |
| | 0x0000_0210 | BUF[1].ID | | ID сообщения буфера 01 |
| | | … | | |
| | 0x0000_03FC | BUF[31].DATAH | | Данные сообщения буфера 31 |
| | | | | |
| | 0x0000_0500 | FILTER[0].MASK | | Маска для приема сообщения в буфер 00 |
| | 0x0000_0504 | FILTER[0].FILTER | | Фильтр для приема сообщения в буфер 00 |
| | 0x0000_0508 | FILTER[1].MASK | | Маска для приема сообщения в буфер 01 |
| | | ... | | |
| | 0x0000_05FC | FILTER[31].FILTER | | Фильтр для приема сообщения в буфер 31 |

Примечание – Регистры RXID, RXDLC, RXDATAL/H (0x20-0x2C) и TXID, TXDLC, TXDATAL/H (0x30 – 0x3C) предназначены для временного хранения данных принятого или передаваемого пакета соответственно. Использовать только для отладки.

### 20.19.1 *CONTROL*

Таблица 286 – Регистр управление контроллером CAN

| Base ADDR= | | 0x4008_B000<br>0x4008_C000 | | | Offset= | | 0x0000_0000 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | ROP | SAP | STM | ROM | CAN_EN |
| | | | | | | | | | | | | | | | |

Таблица 287 – Назначение разрядов регистра CONTROL

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31...5 | - | | Зарезервировано |
| 4 | ROP | | Прием собственных пакетов (Receiveownpackets):<br>1 – контроллер принимает собственные пакеты;<br>0 – контроллер принимает только чужие пакеты |
| 3 | SAP | | Подтверждение собственных пакетов (SendACKonownpackets):<br>1 – контроллер подтверждает прием собственных пакетов;<br>0 – контроллер подтверждает прием только чужих пакетов |
| 2 | STM | | Режим самотестирования (SelfTestMode):<br>1 – контроллер работает в режиме самотестирования;<br>0 – контроллер работает в нормальном режиме |
| 1 | ROM | | Режим «Только прием» (ReadOnlyMode):<br>1 – контроллер работает только на прием;<br>0 – контроллер работает в нормальном режиме |
| 0 | CAN_EN | | Режим работы контроллера CAN:<br>1 – разрешение работы;<br>0 – сброс |

### 20.19.2 *STATUS*

Таблица 288 – Регистр состояния контроллера CAN

| Base ADDR= | | 0x4008_B000<br>0x4008_C000 | | | Offset= | | 0x0000_0004 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| TXERRCNT [7:0] | | | | | | | | RXERRCNT [7:0] | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | - | - | TXERRCNT8 | RXERRCNT8 | ERRSTATUS [1:0] | | IDLOWER | ACKERR | FRAMEERR | CRCERR | BITSTUFF ERR | BITERR | ERROR OVER | TX READY | RX READY |
| | | | | | | | | | | | | | | | |

Таблица 289 – Назначение разрядов регистра STATUS

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31...24 | TXERRCNT [7:0] | | Счетчик ошибок передатчика TEC, биты [7:0]:<br>TEC > 127, ERROR PASSIVE |
| 23…16 | RXERRCNT [7:0] | | Счетчик ошибок приемника REC, биты [7:0]:<br>REC > 127, ERROR PASSIVE |
| 15…13 | - | | |
| 12 | TXERRCNT8 | | Счетчик ошибок передатчика TEC, бит 8:<br>0 – TEC менее 255;<br>1 – TEC более 255 |
| 11 | RXERRCNT8 | | Счетчик ошибок приемника REC, бит 8:<br>0 – REC менее 255;<br>1 – REC более 255 |
| 10…9 | ERRSTATUS[1:0] | | Статус состояния контроллера CAN:<br>00 – ERRORACTIVE, при возникновении ошибки отсылается флаг активной ошибки;<br>01 – ERRORPASSIVE, при возникновении ошибки отсылается флаг пассивной ошибки;<br>1x – BUSOFF, ожидается восстановление шины |
| 8 | ID<br>LOWER | | Флаг «проигрыша» арбитража:<br>0 – при передаче не было проигрыша арбитража;<br>1 – при передаче был проигран арбитраж |
| 7 | ACK<br>ERR | | Флаг ошибки подтверждения приема:<br>0 – нет ошибки;<br>1 – есть ошибка.<br>*Примечание* – сбрасывается записью нуля |
| 6 | FRAME<br>ERR | | Флаг ошибки формата пакета:<br>0 – нет ошибки;<br>1 – есть ошибка.<br>*Примечание* – сбрасывается записью нуля |
| 5 | CRC<br>ERR | | Флаг ошибки контрольной суммы пакета:<br>0 – нет ошибки;<br>1 – есть ошибка.<br>*Примечание* – сбрасывается записью нуля |
| 4 | BIT<br>STUFF ERR | | Флаг ошибки вставленных бит пакета:<br>0 – нет ошибки;<br>1 – есть ошибка.<br>*Примечание* – сбрасывается записью нуля |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 3 | BIT<br>ERR | | Флаг ошибки передаваемых бит пакета:<br>0 – нет ошибки;<br>1 – есть ошибка.<br>*Примечание* – сбрасывается записью нуля |
| 2 | ERROR<br>OVER | | Флаг превышения TEC и REC уровня, заданного ERROR_MAX:<br>0 – ERROR_MAX ≤ TEC и REC;<br>1 – ERROR_MAX > TEC или REC.<br>*Примечание* – сбрасывается записью нуля |
| 1 | TX<br>READY | | Флаг наличия данных в буфере для отправки:<br>0 – буферы пусты, нет сообщений готовых к передаче;<br>1 – есть не пустой буфер, в котором сообщения готовы к передаче<br>*Примечание – биты TX_READY регистра STATUS выставляются при условии, когда установлен бит TX_INT_EN = 1 (регистр CAN_INT_EN) и выставлен флаг разрешения прерывания от выполняющего отправку буфера в регистрах CAN_INT_TX.* |
| 0 | RX<br>READY | | Флаг наличия принятых сообщений:<br>0 – буферы пусты, нет принятых сообщений;<br>1 – есть буфер, содержащий принятые сообщения<br>*Примечание – биты RX_READY регистра STATUS выставляются при условии, когда установлен бит RX_INT_EN = 1 (регистр CAN_INT_EN) и выставлен флаг разрешения прерывания от выполняющего приём буфера в регистрах CAN_INT_RX.* |

Примечание – Для организации надежного сброса статусных флагов прерываний необходимо учитывать требования, изложенные в пункте 21.1.2 «Обработчики исключений».

### 20.19.3 *BITTMNG*

Таблица 290 – Регистр задания скорости работы BITTMNG

| Base ADDR= | 0x4008_B000<br>0x4008_C000 | Offset= | 0x0000_0008 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | SB | SJW[1:0] | | SEG2[2:0] | | | SEG1[2:0] | | | PSEG[2:0] | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BRP[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 291 – Назначение разрядов регистра BITTMNG

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31...28 | - | | Зарезервировано |
| 27 | SB | | Семплирование:<br>0 – однократное;<br>1 – трехкратное с мажоритарным контролем |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 26…25 | SJW [1:0] | | Значение размера фазы SJW:<br>11 = Synchronization jump width time = 4 x TQ<br>10 = Synchronization jump width time = 3 x TQ<br>01 = Synchronization jump width time = 2 x TQ<br>00 = Synchronization jump width time = 1 x TQ<br>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:<br>111 = Phase Segment 2 time = 8 x TQ<br>110 = Phase Segment 2 time = 7 x TQ<br>101 = Phase Segment 2 time = 6 x TQ<br>100 = Phase Segment 2 time = 5 x TQ<br>011 = Phase Segment 2 time = 4 x TQ<br>010 = Phase Segment 2 time = 3 x TQ<br>001 = Phase Segment 2 time = 2 x TQ<br>000 = Phase Segment 2 time = 1 x TQ<br>SEG2 – это время, используемое для сокращения битового интервала при подстройке |
| 21…19 | SEG1 [2:0] | | Значение размера фазы SEG1:<br>111 = Phase Segment 1 time = 8 x TQ<br>110 = Phase Segment 1 time = 7 x TQ<br>101 = Phase Segment 1 time = 6 x TQ<br>100 = Phase Segment 1 time = 5 x TQ<br>011 = Phase Segment 1 time = 4 x TQ<br>010 = Phase Segment 1 time = 3 x TQ<br>001 = Phase Segment 1 time = 2 x TQ<br>000 = Phase Segment 1 time = 1 x TQ<br>SEG1 – это время, используемое для увеличения битового интервала при подстройке |
| 18…16 | PSEG[2:0] | | Значение размера фазы PSEG<br>111 = Propagation time = 8 x TQ<br>110 = Propagation time = 7 x TQ<br>101 = Propagation time = 6 x TQ<br>100 = Propagation time = 5 x TQ<br>011 = Propagation time = 4 x TQ<br>010 = Propagation time = 3 x TQ<br>001 = Propagation time = 2 x TQ<br>000 = Propagation time = 1 x TQ<br>PSEG - это время, компенсирующее задержку распространения сигналов в шине CAN |
| 15…0 | BRP [15:0] | | Предделитель системной частоты:<br>CLK = CANCLK / (BRP + 1)<br>TQ(us) = 1 / CLK(MHz) = (BRP + 1) / CANCLK(MHz) |

### 20.19.4 *INT_EN*

Таблица 292 – Регистр разрешения прерываний контроллера INT_EN

| Base ADDR= | | 0x4008_B000<br>0x4008_C000 | | | Offset= | | 0x0000_0010 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | ERROVER INTN | ERRINTEN | TXINTEN | RXINTEN | GLBINTEN |
| | | | | | | | | | | | | | | | |

Таблица 293 – Назначение разрядов регистра INT_EN

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31...5 | - | | Зарезервировано |
| 4 | ERROVER INTEN | | Флаг разрешения прерывания по превышению TEC или REC допустимого значения в ERROR_MAX:<br>0 – запрещено прерывание;<br>1 – разрешено прерывание |
| 3 | ERRINT EN | | Флаг разрешения прерывания по возникновению ошибки:<br>0 – запрещено прерывание;<br>1 – разрешено прерывание |
| 2 | TXINT EN | | Флаг разрешения прерывания по возможности передачи:<br>0 – запрещено прерывание;<br>1 – разрешено прерывание |
| 1 | RXINT EN | | Флаг разрешения прерывания по приему сообщений:<br>0 – запрещено прерывание;<br>1 – разрешено прерывание |
| 0 | GLBINT EN | | Общий флаг разрешения прерывания блока CAN:<br>0 – запрещено прерывание;<br>1 – разрешено прерывание |

### 20.19.5 *OVER*

Таблица 294 – Регистр границы счетчика ошибок OVER

| Base ADDR= | | 0x4008_B000<br>0x4008_C000 | | | Offset= | | 0x0000_001C | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | ERROR_MAX[7:0] | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 295 – Назначение разрядов регистра OVER

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31...8 | - | | Зарезервировано |
| 7…0 | ERRORMAX [7:0] | | Регистр границы счетчика ошибок<br>Допустимое значение счетчиков ошибок TEC и REC, при превышении которого вырабатывается флаг ERROR_OVER |

### 20.19.6 *BUF_CON[x]*

Таблица 296 – Регистр управления буфером BUF_CON[x]

| Base ADDR= | | 0x4008_B000<br>0x4008_C000 | | | Offset= | | 0x0000_0040<br>..<br>0x0000_00BC | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | OVERWR | RXFULL | TXREQ | PRIOR 0 | RTREN | OVEREN | RXTXn | EN |
| | | | | | | | | | | | | | | | |

Таблица 297 – Назначение разрядов регистра BUF_CON[x]

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31...8 | - | | Зарезервировано |
| 7 | OVER_WR | | Флаг перезаписи принятого сообщения:<br>0 – не было перезаписи;<br>1 – была перезапись принятого сообщения |
| 6 | RX_FULL | | Флаг готовности приема:<br>0 – нет принятого сообщения;<br>1 – принятое сообщение в буфере |
| 5 | TX_REQ | | Запрос на отправку сообщения:<br>0 – нет запроса или отправлено;<br>1 – запрос на отправку |
| 4 | PRIOR_0 | | Приоритет при отправке:<br>0 – нет приоритета;<br>1 – приоритет |
| 3 | RTR_EN | | Режим ответа на RTR:<br>0 – не отвечать при приеме RTR;<br>1 – ответить при приеме RTR в буфер |
| 2 | OVER_EN | | Разрешение перезаписи принятого сообщения:<br>0 – не разрешена перезапись;<br>1 – разрешена перезапись сообщения |
| 1 | RX_TXn | | Режим работы буфера:<br>0 – на передачу;<br>1 – на прием |
| 0 | EN | | Разрешение работы буфера:<br>0 – отключен;<br>1 – работает |

### 20.19.7 *INT_RX*

Таблица 298 – Флаги разрешения прерываний от приемных буферов

| Base ADDR= | | 0x4008_B000<br>0x4008_C000 | | | Offset= | | 0x0000_00C0 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CAN_INT_RX[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CAN_INT_RX[31:15] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 299 – Назначение разрядов регистра INT_RX

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31...0 | CAN_INT_RX[31:0] | | Флаги разрешения прерываний от буферов по приему сообщений:<br>CAN_INT_RX[0] – для первого буфера<br>CAN_INT_RX[1] – для второго буфера и так далее |

### 20.19.8 *RX*

Таблица 300 – Флаги RX_FULL от приемных буферов

| Base ADDR= | | 0x4008_B000<br>0x4008_C000 | | | Offset= | | 0x0000_00C4 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CAN_RX[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CAN_RX[31:15] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 301 – Назначение разрядов регистра RX

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31...0 | CAN_RX[31:0] | | Флаги RX_FULL разрешенных на прием буферов:<br>CAN_RX[0] – флаг RX_FULL от первого буфера |

### 20.19.9 *INT_TX*

Таблица 302 – Флаги разрешения прерываний от передающих буферов

| Base ADDR= | | 0x4008_B000<br>0x4008_C000 | | | Offset= | | 0x0000_00C8 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CAN_INT_TX[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CAN_INT_TX[31:15] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 303 – Назначение разрядов регистра INT_TX

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31...0 | CAN_INT_TX[31:0] | | Флаги разрешения прерываний от буферов по передаче сообщений:<br>CAN_INT_TX[0] – для первого буфера<br>CAN_INT_TX[1] – для второго буфера и так далее |

### 20.19.10 *TX*

Таблица 304 – Флаги ~TX_REQ от передающих буферов

| Base ADDR= | | 0x4008_B000<br>0x4008_C000 | | | Offset= | | 0x0000_00CC | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CAN_TX[31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CAN_TX[31:15] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 305 – Назначение разрядов регистра TX

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31...0 | CAN_TX[31:0] | | Флаги ~TX_REQ разрешенных на передачу буферов:<br>CAN_TX[0] – флаг ~TX_REQ от первого буфера<br>CAN_TX[1] – флаг ~TX_REQ от второго буфера и так далее, доступны только на чтение |

### 20.19.11 *RXID*

### 20.19.12 *TXID*

### 20.19.13 *BUF[x].ID*

### 20.19.14 *FILTER[x].MASK*

### 20.19.15 *FILTER[x].FILTER*

Таблица 306 – Регистр принятого ID сообщения RXID/ Регистр передаваемого ID сообщения TXID/ ID сообщения буфера BUF[x].ID/ Маска для приема сообщения в буфер FILTER[x].MASK/ Фильтр для приема сообщения в буфер FILTER[x].FILTER

| Base ADDR= | 0x4008_B000<br>0x4008_C000 | Offset= | 0x0000_0020<br>0x0000_0030<br><br>0x0000_0200<br>0x0000_0210<br>..<br>0x0000_03F0<br><br>0x0000_0800<br>0x0000_0804<br>0x0000_0808<br>..<br>0x0000_08FC |
|---|---|---|---|

| REG Name: | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | SID[10:0] | | | | | | | | | | | EID[17:16] | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EID[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 307 – Назначение разрядов регистров RXID, TXID, BUF[x].ID, FILTER[x].MASK, FILTER[x].FILTER

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31...29 | - | | Зарезервировано |
| 28…18 | SID<br>[10:0] | | Поле SID.<br>Для стандартного и расширенного пакетов CAN.<br>Чем меньше значение поля, тем больший приоритет имеет пакет при арбитраже |
| 17…0 | EID<br>[17:0] | | Поле EID.<br>Для расширенных пакетов CAN.<br>Чем меньше значение поля, тем больший приоритет имеет пакет при арбитраже |

### 20.19.16 *RXDLC*

### 20.19.17 *TXDLC*

### 20.19.18 *BUF[x].DLC*

Таблица 308 – Регистр принятого DLC сообщения RXDLC/ Регистр передаваемого DLC сообщения TXDLC/ DLC сообщения буфера BUF[x].DLC

| Base ADDR= | 0x4008_B000<br>0x4008_C000 | Offset= | 0x0000_0024<br>0x0000_0034<br><br>0x0000_0204<br>0x0000_0214<br>..<br>0x0000_03F4 |
|---|---|---|---|
| REG Name: | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | IDE | SSR | R0 | R1 | RTR | | | | | DLC[3:0] | | | |
| | | | | | | | | | | | | | | | |

Таблица 309 – Назначение разрядов регистров RXDLC, TXDLC, BUF[x].DLC

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31...13 | - | | Зарезервировано |
| 12 | IDE | | Поле IDE.<br>Поле, обозначающее формат пакета:<br>0 – стандартный пакет;<br>1 – расширенный пакет |
| 11 | SSR | | Поле SSR, расширенного формата.<br>Всегда должно быть равно “1” |
| 10 | R0 | | Поле R0.<br>Всегда должно быть равно “0” |
| 9 | R1 | | Поле R1, расширенного формата.<br>Всегда должен быть“0” |
| 8 | RTR | | Поле RTR, запроса обратного ответа:<br>0 – нет запроса;<br>1 – есть запрос.<br>Если узел получил пакет с запросом обратного ответа, то он должен ответить |
| 7…4 | - | | Зарезервировано |
| 3…0 | DLC[3:0] | | Поле DLC, длина передаваемых данных в пакете:<br>0000 – нет данных<br>0001 – 1 байт<br>0010 – 2 байт<br>0011 – 3 байт<br>0100 – 4 байт<br>0101 – 5 байт<br>0110 – 6 байт<br>0111 – 7 байт<br>1000 – 8 байт<br>1xxx – 8 байт и недопустимо |

### 20.19.19 *RXDATAL*

### 20.19.20 *TXDATAL*

### 20.19.21 *BUF[x].DATAL*

Таблица 310 – Регистр принятых данных RXDATAL/ Регистр передаваемых данных TXDATAL/ Данные сообщения буфера BUF[x].DATAL

| Base ADDR= | 0x4008_B000<br>0x4008_C000 | Offset= | 0x0000_0028<br>0x0000_0038<br><br>0x0000_0208<br>0x0000_0218<br>..<br>0x0000_03F8 |
|---|---|---|---|

| REG Name: | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DB3[7:0] | | | | | | | | DB2[7:0] | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DB1[7:0] | | | | | | | | DB0[7:0] | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 311 – Назначение разрядов регистров RXDATAL, TXDATAL, BUF[x].DATAL

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 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 |

### 20.19.22 *RXDATAH*

### 20.19.23 *TXDATAH*

### 20.19.24 *BUF[x].DATAH*

Таблица 312 Регистр принятых данных RXDATAH/ Регистр передаваемых данных TXDATAH/ Данные сообщения буфера BUF[x].DATAH

| Base ADDR= | 0x4008_B000<br>0x4008_C000 | Offset= | 0x0000_002C<br>0x0000_003C<br><br>0x0000_020C<br>0x0000_021C<br>..<br>0x0000_03FC |
|---|---|---|---|

| REG Name: | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DB7[7:0] | | | | | | | | DB6[7:0] | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DB5[7:0] | | | | | | | | DB4[7:0] | | | | | | | |
| | | | | | | | | | | | | | | | |

Таблица 313 – Назначение разрядов регистров RXDATAH, TXDATAH, BUF[x].DATAH

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…24 | DB7[7:0] | | Поле DB7. Четвертый байт, передаваемый в пакете |
| 23…16 | DB6[7:0] | | Поле DB5. Третий байт, передаваемый в пакете |
| 15…8 | DB5[7:0] | | Поле DB4. Второй байт, передаваемый в пакете |
| 7…0 | DB4[7:0] | | Поле DB3. Первый байт, передаваемый в пакете |

## 20.20 Описание регистров контроллеров SSP SSP_CNTR

Таблица 314 – Регистры контроллеров SSP

| Базовый адрес | Наимено-вание | Тип | Значение после сброса | Раз-мер, бит | Описание |
|---|---|---|---|---|---|
| 0x4008_9000 | SSP0_CNTR | | | | Контроллер SSP0 |
| 0x4008_A000 | SSP1_CNTR | | | | Контроллер SSP1 |
| Смещение | | | | | |
| 0x0000 | CR0 | RW | 0x00000 | 18 | Регистр управления 0 |
| 0x0004 | CR1 | RW | 0x00 | 5 | Регистр управления 1 |
| 0x008 | DR | RW | 0x---- | 32 | Буфера FIFO приемника (чтение)<br>Буфер FIFO передатчика (запись) |
| 0x000C | SR | RO | 0x03 | 5 | Регистр состояния |
| 0x0010 | CPSR | RW | 0x00 | 8 | Регистр делителя тактовой частоты |
| 0x0014 | IMSC | RW | 0x00 | 7 | Регистр маски прерывания |
| 0x0018 | RIS | RO | 0x68 | 7 | Регистр состояния прерываний без учета маскирования |
| 0x001C | MIS | RO | 0x00 | 7 | Регистр состояния прерываний с учетом маскирования |
| 0x0020 | ICR | WO | 0x0 | 2 | Регистр сброса прерывания |
| 0x0024 | DMACR | RW | 0x0 | 2 | Регистр управления прямым доступом к памяти |

Примечание – В поле «тип» указан вид доступа к регистру: RW – чтение и запись, RO – только чтение, WO – только запись.

### 20.20.1 *CR0*

Регистр управления 0

Регистр CR0 содержит семь битовых полей, предназначенных для управления блоками модуля SSP. Назначение разрядов регистра представлено в таблице 315.

Таблица 315 – Формат регистра CR0

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31…18 | - | Зарезервировано |
| 17 | SSPFRX | Для ведомого модуля включает быстрый режим протокола SPI и активирует синхронизацию сигнала RXD.<br>1 – активация быстрого режима работы ведомого<br>0 – нормальный режим работы |
| 16 | DSS[4] | Размер слова данных: DSS[3:0] + 16 бит |
| 15…8 | SCR | Скорость последовательного обмена.<br>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.<br>00 – протокол SPI фирмы Motorola;<br>01 – протокол SSI фирмы Texas Instruments;<br>10 – протокол Microwire фирмы National Semiconductor;<br>11 – резерв |
| 3…0 | DSS[3:0] | Размер слова данных: |

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| | | 0000 – резерв<br>0001 – резерв<br>0010 – резерв<br>0011 – 4 бита<br>0100 – 5 бит<br>0101 – 6 бит<br>0110 – 7 бит<br>0111 – 8 бит<br>1000 – 9 бит<br>1001 – 10 бит<br>1010 – 11 бит<br>1011 – 12 бит<br>1100 – 13 бит<br>1101 – 14 бит<br>1110 – 15 бит<br>1111 – 16 бит |

### 20.20.2 *CR1*

*Регистр управления 1*

Регистр CR1 содержит четыре битовых поля, предназначенных для управления блоками модуля SSP. Назначение разрядов регистра представлено в таблице 316.

Таблица 316 – Регистр CR1

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 15…5 | | Резерв, при чтении результат не определен.<br>При записи следует устанавливать в 0 |
| 4 | RESTxFIFO | Программный сброс FIFO передатчика:<br>1 – активация сброса FIFO передатчика |
| 3 | SOD | Запрет выходных линий в режиме ведомого устройства.<br>Бит используется только в режиме ведомого устройства (MS=1). Это позволяет организовать двусторонний обмен данными в системах, содержащих одно ведущее и несколько ведомых устройств.<br>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.<br>0 – управление линией SSP_TXD в ведомом режиме разрешено.<br>1 – управление линией SSP_TXD в ведомом режиме запрещено |
| 2 | MS | Выбор ведущего или ведомого режима работы:<br>0 – ведущий модуль (устанавливается по умолчанию);<br>1 – ведомый модуль |
| 1 | SSE | Разрешение работы приемопередатчика:<br>0 – работа запрещена;<br>1 – работа разрешена |
| 0 | LBM | Тестирование по шлейфу:<br>0 – нормальный режим работы приемопередатчика;<br>1 – выход регистра сдвига передатчика соединен со входом регистра сдвига приемника |

### 20.20.3 *DR*

*Регистр данных*

Регистр SSPDR имеет разрядность 32 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 32 бит, перед записью в регистр SSPDR необходимо обеспечить выравнивание данных по правой границе. Блок передатчика игнорирует неиспользуемые биты. Принятые информационные слова автоматически выравниваются по правой границе в блоке приемника.

В режиме обмена данными Microwire фирмы National Semiconductor модуль SSP по умолчанию работает с восьмиразрядными информационными словами (старший значащий байт игнорируется). Размер принимаемых данных задается программно. Буфера FIFO приемника и передатчика автоматически не очищаются даже в случае, если бит SSE установлен в 0. Это позволяет заполнить буфер передатчика необходимой информацией заблаговременно, перед разрешением работы модуля.

Назначение разрядов регистра SSPDR описано в таблице 317.

Таблица 317 – Формат регистра DR

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31…0 | DATA | Принимаемые данные (чтение).<br>Передаваемые данные (запись).<br>В случае, если выбран размер информационного слова менее 32 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 |

### 20.20.4 *SR*

*Регистр состояния*

Регистр состояния доступен только для чтения и содержит информацию о состоянии буферов FIFO приемника и передатчика и занятости модуля SSP.

Назначение бит регистра SSPCPSR представлено в таблице 318.

Таблица 318 – 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:<br>0 – модуль SSP не активен;<br>1 – модуль SSP в настоящее время передает и/или принимает данные, либо буфер FIFO передатчика не пуст |
| 3 | RFF | Буфер FIFO приемника заполнен:<br>0 – не заполнен;<br>1 – заполнен |
| 2 | RNE | Буфер FIFO приемника не пуст:<br>0 – пуст;<br>1 – не пуст |
| 1 | TNF | Буфер FIFO передатчика не заполнен:<br>0 – заполнен;<br>1 – не заполнен |

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 0 | TFE | Буфер FIFO передатчика пуст:<br>0 – не пуст;<br>1 – пуст |

### 20.20.5 *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 319 отображает назначение бит регистра SSPSR.

Таблица 319 – 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 |

### 20.20.6 *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 320.

Таблица 320 – Регистр IMSC

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31…7 | | Резерв. Не модифицируйте. При чтении выдаются нули |
| 6 | TNBSYIM | Маска прерывания по пустоте сдвигового регистра передатчика.<br>1 – не маскирована.<br>0 – маскирована |
| 5 | TFEIM | Маска прерывания по пустоте FIFO передатчика.<br>1 – не маскирована.<br>0 – маскирована |
| 4 | RNEIM | Маска прерывания по наличию данных в FIFO приемника.<br>1 – не маскирована.<br>0 – маскирована |
| 3 | TXIM | Маска прерывания по заполнению на 50% и менее буфера FIFO передатчика.<br>1 – не маскирована.<br>0 – маскирована |
| 2 | RXIM | Маска прерывания по заполнению на 50% и менее буфера FIFO приемника.<br>1 – не маскирована.<br>0 – маскирована |

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 1 | RTIM | Маска прерывания по таймауту приемника (буфер FIFO приемника не пуст и не было попуток его чтения в течение времени таймаута).<br>1 – не маскирована.<br>0 – маскирована |
| 0 | RORIM | Маска прерывания по переполнению буфера приемника.<br>1 – не маскирована.<br>0 – маскирована |

### 20.20.7 *RIS*

*Регистр состояния прерываний*

Этот регистр доступен только для чтения и содержит текущее состояние прерываний без учета маскирования. Данные, записываемые в регистр, игнорируются.

Таблица 321 отображает назначение бит в регистре RIS.

Таблица 321 – Регистр RIS

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31… 7 | | Резерв. Не модифицируйте. При чтении выдаются нули |
| 6 | TNBSYRIS | Состояние до маскирования прерывания SSPTNBSYINTR |
| 5 | TFERIS | Состояние до маскирования прерывания SSPTFEINTR |
| 4 | RNERIS | Состояние до маскирования прерывания SSPRNEINTR |
| 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 |

### 20.20.8 *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 322.

Таблица 322 – Регистр MIS

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31…7 | | Резерв. Не модифицируйте. При чтении выдаются нули |
| 6 | TNBSYMIS | Состояние маскированного прерывания SSPTNBSYINTR |
| 5 | TFEMIS | Состояние маскированного прерывания SSPTFEINTR |
| 4 | RNEMIS | Состояние маскированного прерывания SSPRNEINTR |
| 3 | TXMIS | Состояние маскированного прерывания SSPTXINTR |
| 2 | RXMIS | Состояние маскированного прерывания SSPRXINTR |
| 1 | RTMIS | Состояние маскированного прерывания SSPRTINTR |
| 0 | RORMIS | Состояние маскированного прерывания SSPRORINTR |

### 20.20.9 *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 323.

Таблица 323 – 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 |

### 20.20.10 *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 324.

Таблица 324 – 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 |

## 20.21 Описание регистров контроллеров UART UART_CNTR

Модуль универсального асинхронного приемопередатчика (UART – Universal Synchronous Asynchronous Receiver Transmitter) является периферийным устройством 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).

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) устанавливаются в значение 0 после сброса по включению питания или системного сброса.

Столбец «Тип» (таблица 325) определяет режим доступа к регистру в соответствии с обозначениями:

- RW – чтение и запись;
- RO – только чтение;
- WO – только запись.

Таблица 325 – Регистры контроллеров UART

| Смещение | Наименование | Тип | Значение после сброса | Размер, бит | Описание |
|---|---|---|---|---|---|
| 0x4008_D000 | UART0_CNTR | | | | Контроллер UART0 |
| 0x4008_E000 | UART1_CNTR | | | | Контроллер UART1 |
| 0x4008_F000 | UART2_CNTR | | | | Контроллер UART2 |
| 0x4009_0000 | UART3_CNTR | | | | Контроллер UART3 |
| | | | | | |
| 0x000 | DR | RW | 0x--- | 13/9 | Регистр данных |
| 0x004 | RSR_ECR | RW | 0x0 | 4/0 | Регистр состояния приемника / Сброс ошибки приемника |
| 0x008–0x014 | - | - | – | - | Зарезервировано |
| 0x018 | FR | RO | 0b-10010--- | 9 | Регистр флагов |
| 0x01C | - | - | – | - | Зарезервировано |
| 0x020 | - | - | – | - | Зарезервировано |
| 0x024 | IBRD | RW | 0x0000 | 16 | Целая часть делителя скорости обмена данными |
| 0x028 | FBRD | RW | 0x00 | 6 | Дробная часть делителя скорости обмена данными |
| 0x02C | LCR_H | RW | 0x00 | 9 | Регистр управления линией |
| 0x030 | CR | RW | 0x0300 | 16 | Регистр управления |
| 0x034 | IFLS | RW | 0x12 | 6 | Регистр порога прерывания по заполнению буфера FIFO |
| 0x038 | IMSC | RW | 0x0000 | 14 | Регистр маски прерывания |
| 0x03C | RIS | RO | 0x100- | 14 | Регистр состояния прерываний |
| 0x040 | MIS | RO | 0x000- | 14 | Регистр состояния прерываний с маскированием |
| 0x044 | ICR | WO | – | 11 | Регистр сброса прерывания |
| 0x048 | DMACR | RW | 0x00 | 3 | Регистр управления DMA |

### 20.21.1 *DR*

*Регистр данных*

**В ходе передачи данных:**

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

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

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

**В ходе приема данных:**

Если буфер FIFO приемника разрешен, байт данных и четыре бита состояния (разрыв, ошибка формирования кадра, четность, переполнение) сохраняются в 13-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 326 – Формат регистра UARTDR

| № бита | Сигнал | Назначение |
|---|---|---|
| 15…13 | | Резерв |
| 12 | 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 |
| 11 | 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 |
| 10 | 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. |
| 9 | 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 |
| 8…0 | DATA | Принимаемые данные (чтение).<br>Передаваемые данные (запись) |

Примечание – Необходимо запрещать работу приемопередатчика перед любым перепрограммированием его регистров управления. Если приёмопередатчик переводится в отключенное состояние во время передачи или приема символа, то перед остановкой он завершает выполняемую операцию.

### 20.21.2 *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 327 показывает назначение бит регистра UARTRSR/UARTECR.

Таблица 327 – 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 был перезаписан 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/

### 20.21.3 *FR*

*Регистр флагов*

После сброса биты регистра флагов TXFF, RXFF и BUSY устанавливаются в 0, а биты TXFE и RXFE – в 1. Таблица 328 представляет информацию о назначении бит регистра UARTFR.

Таблица 328 – 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 |

### 20.21.4 *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.

Требуемое значение коэффициента деления для формирования сигнала IrLPBaud16 вычисляется по формуле:

$$ILPDVSR = \frac{F_{UARTCLK}}{F_{IrLPBaud16}},$$

где номинальное значение частоты $F_{IrLPBaud16}$ составляет 1,8432 МГц.

Коэффициент деления должен быть установлен таким образом, чтобы выполнялось соотношение

$$1.42\ \text{МГц} < F_{IrLPBaud16} < 2.12\ \text{МГц},$$

что, в свою очередь, гарантирует формирование кодеком импульсов данных с длительностью 1,41 ÷ 2,11 мкс (в три раза длиннее периода сигнала IrLPBaud16).

Таблица 329 – 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.<br>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.

### 20.21.5 *IBRD*

*Регистр целой части делителя скорости передачи данных*

Назначение бит регистра UARTBIRD показано ниже (Таблица 330).

Таблица 330 – 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 |

### 20.21.6 *FBRD*

*Регистр дробной части делителя скорости передачи данных*

Таблица 331 – 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:

$$BAUDDIV = \frac{F_{UARTCLK}}{16 \cdot Baud_rate},$$

Где $F_{UARTCLK}$ – тактовая частота контроллера UART,

Baud_rate – требуемая скорость передачи данных.

Коэффициент BAUDDIV состоит из целой и дробной частей – BAUDDIV_INT и BAUDDIV_FRAC, соответственно.

Примечания:

1 Изменения содержимого регистров UARTIBRD и UARTFBRD вступают в силу только после завершения передачи и приема текущего символа данных.
2 Минимальный допустимый коэффициент деления – 1, максимальный 65535 ($2^{16}$ - 1). Таким образом, значение UARTIBRD, равное 0, является недопустимым, при этом значение регистра UARTFBRD игнорируется.
3 Аналогично, при UARTIBRD равном 65535 (0xFFFF), значение UARTFBRD не может быть больше нуля. Невыполнение этого условия приведет к прерыванию приема или передачи.

Далее приведен пример вычисления коэффициента деления.

*Пример. Вычисление коэффициента деления*

Пусть требуемая скорость передачи данных составляет 230400 бит/с, частота тактового сигнала UARTCLK равна 4 МГц. Тогда

$$\text{Коэффициент деления} = (4 \cdot 10^6)/(16 \cdot 230400) = 1{,}085.$$

Таким образом, BRD = 1, FBRD = 0,085.

Следовательно, значение, записываемое в регистр UARTBFRD, равно

$$m = integer((0{,}085 \cdot 64) + 0{,}5) = 5.$$

Реальное значение коэффициента деления = $1 + 5/64 = 1{,}078$.

Реальная скорость передачи данных = $(4 \cdot 10^6)/(16 \cdot 1.078) = 231911$ бит/с.

Ошибка установки скорости = $\frac{231911-230400}{230400} \cdot 100\% = 0{,}656\%$.

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.

Таблица 332 представляет 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 332 – 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 |

Таблица 333 содержит значения коэффициента деления для типичных скоростей передачи данных при частоте UARTCLK = 4 МГц.

Таблица 333 – Коэффициенты деления при частоте UARTCLK = 4МГц

| 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 |

### 20.21.7 *LCR_H*

Регистр управления линией

Данный регистр обеспечивает доступ к разрядам с 30 по 22 регистра UARTLCR. При сбросе все биты регистра UARTLCR_H обнуляются.

Таблица 334 показывает назначение разрядов регистра UARTLCR_H.

Таблица 334 – Регистр UARTLCR_H

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 15…9 | | Резерв. Не модифицируйте. При чтении выдаются нули |
| 8 | WLEN[2] | Длина слова – количество передаваемых или принимаемых информационных бит в кадре:<br>0 – длина слова определяется WLEN[1:0]<br>1 – 9 бит |
| 7 | SPS | Передача бита четности с фиксированным значением.<br>0 – запрещена;<br>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[1:0] | Длина слова – количество передаваемых или принимаемых информационных бит в кадре:<br>0b11 – 8 бит<br>0b10 – 7 бит<br>0b01 – 6 бит<br>0b00 – 5 бит |
| 4 | FEN | Разрешение работы буфера FIFO приемника и передатчика.<br>0 – запрещено;<br>1 – разрешено |
| 3 | STP2 | Режим передачи двух стоповых бит.<br>0 – один стоповый бит;<br>1 – два стоповых бита.<br>Приемник не проверяет наличие дополнительного стопового бита в кадре |
| 2 | EPS | Четность/нечетность.<br>0 – бит четности дополняет количество единиц в информационной части кадра до нечетного;<br>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.<br>0 – кадр не содержит бита четности;<br>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 31-разрядный регистр UARTLCR, который обновляется по стробу, формируемому при записи в UARTLCR_H. Для того, чтобы изменение параметров коэффициента деления частоты обмена данными вступило в силу, после изменения значения регистров

UARTIBRD и/или UARTFBRD необходимо осуществить запись данных в регистр UARTLCR_H.

Примечания:

1 Изменение значений трех регистров можно осуществить корректно двумя способами:
   - запись UARTIBRD, запись UARTFBRD, запись UARTLCR_H;
   - запись UARTFBRD, запись UARTIBRD, запись UARTLCR_H.
2 Для того чтобы изменить значение лишь одного из регистров (UARTIBRD или UARTFBRD) необходимо выполнить следующий шаг:
   - запись UARTIBRD (или UARTFBRD), запись UARTLCR_H.

Таблица 335 содержит таблицу истинности для бит управления контролем четности PEN, EPS и SPS регистра управления линией UARTLCR_H.

Таблица 335 – Управление режимом контроля четности

| PEN | EPS | SPS | Бит контроля четности |
|---|---|---|---|
| 0 | X | X | Не передается, не проверяется |
| 1 | 1 | 0 | Проверка четности слова данных |
| 1 | 0 | 0 | Проверка нечетности слова данных |
| 1 | 0 | 1 | Бит четности постоянно равен 1 |
| 1 | 1 | 1 | Бит четности постоянно равен 0 |

*Примечания*

1 Регистры UARTLCR_H, UARTIBRD и UARTFBRD не должны изменяться:
   - при разрешенной работе приемопередатчика;
   - во время завершения приема или передачи данных в процессе остановки (перевода в запрещенное состояние) приемопередатчика.
2 Целостность данных в буферах FIFO не гарантируется в следующих случаях:
   - после установки бита разрыва линии BRK;
   - если программное обеспечение произвело остановку приемопередатчика при наличии данных в буферах FIFO после его повторного перевода в разрешенное состояние.

### 20.21.8 *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 ниже (Таблица 336).

Таблица 336 – 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.<br>1 – разрешено, данные передаются в линию только при активном значении сигнала nUARTCTS. |
| 14 | RTSEn | Разрешение управления потоком данных по RTS.<br>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.<br>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.<br>В случае перевода приемопередатчик в запрещенное состояние в ходе передачи данных, он завершает передачу текущего символа: перед остановкой |
| 7 | LBE | 0 – запрещено;<br>1 – шлейф разрешен.<br>В режиме разрешенного шлейфа:<br>Если установлены бит SIREN=1 и бит регистра управления тестированием UARTTCRSIRTEST=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.<br>После окончания тестирования по шлейфу бит SIRTEST должен быть установлен в 0.<br>Если бит SIRTEST=0, то выходная линия передатчика UARTTXD коммутируется на вход приемника UARTRXD.<br>Как в режиме SIR, так и в режиме UART, выходные линии состояния модема коммутируются на соответствующие входные линии.<br>После сброса бит устанавливается в 0 |
| 6…3 | | Резерв. Не модифицируйте. При чтении выдаются нули |

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 2 | SIRLP | Выбор режима ИК обмена с пониженным энергопотреблением:<br>0 – длительность импульсов данных равна 3/16 длительности передачи бита;<br>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 IrDASIR:<br>0 – запрещено. Сигнал nSIROUT находится в низком состоянии, данные на входе SIRIN не обрабатываются.<br>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.<br>В случае, если UARTEN=0 значение бита не играет роли |
| 0 | UARTEN | Разрешение работы приемопередатчика:<br>0 – работа запрещена. Перед остановкой завершается прием и/или передача обрабатываемого в текущий момент символа.<br>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, биты TXE и UARTEN необходимо установить в логическую 1. Аналогично, для разрешения приема данных необходимо установить в 1 биты RXE и UARTEN.

Примечание – Рекомендуется следующая последовательность действий для программирования регистров управления:

1 остановите работу приемопередатчика;
2 дождитесь окончания приема и/или передачи текущего символа данных;
3 сбросьте буфер передатчика путем установки бита FEN регистра UARTLCR_H в 0;
4 измените настройки регистра UARTCR;
5 возобновите работу приемопередатчика.

### 20.21.9 *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 ниже (Таблица 337).

Таблица 337 – 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:<br>b000 = буфер заполнен на 1/8<br>b001 = буфер заполнен на 1/4<br>b010 = буфер заполнен на 1/2<br>b011 = буфер заполнен на 3/4<br>b100 = буфер заполнен на 7/8<br>b101-b111 = резерв |

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 2…0 | TXIFLSEL | Порог прерывания по заполнению буфера передатчика:<br>b000 = буфер заполнен на 1/8<br>b001 = буфер заполнен на 1/4<br>b010 = буфер заполнен на 1/2<br>b011 = буфер заполнен на 3/4<br>b100 = буфер заполнен на 7/8<br>b101-b111 = резерв |

### 20.21.10 *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 ниже (Таблица 338).

Таблица 338 – Регистр UARTIMSC

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31…14 | | Зарезервировано.<br>Не модифицируйте. При чтении выдаются нули |
| 13 | TNBSYIM | Маска прерывания по отсутствию данных в сдвиговом регистре передатчика UARTTNBSYINTR:<br>1 – установлена;<br>0 – сброшена |
| 12 | TFEIM | Маска прерывания по отсутствию данных в FIFO передатчика UARTTFEINTR:<br>1 – установлена;<br>0 – сброшена |
| 11 | RNEIM | Маска прерывания по наличию данных в FIFO приемника UARTRNEINTR:<br>1 – установлена;<br>0 – сброшена |
| 10 | OEIM | Маска прерывания по переполнению буфера UARTOEINTR:<br>1 – установлена;<br>0 – сброшена |
| 9 | BEIM | Маска прерывания по разрыву линии UARTBEINTR:<br>1 – установлена;<br>0 – сброшена |
| 8 | PEIM | Маска прерывания по ошибке контроля четности UARTPEINTR:<br>1 – установлена;<br>0 – сброшена |
| 7 | FEIM | Маска прерывания по ошибке в структуре кадра UARTFEINTR:<br>1 – установлена;<br>0 – сброшена |
| 6 | RTIM | Маска прерывания по таймауту приема данных UARTRTINTR:<br>1 – установлена;<br>0 – сброшена |
| 5 | TXIM | Маска прерывания от передатчика UARTTXINTR.<br>1 – установлена;<br>0 – сброшена |
| 4 | RXIM | Маска прерывания от приемника UARTRXINTR.<br>1 – установлена;<br>0 – сброшена |
| 3 | DSRMIM | Маска прерывания UARTDSRINTR по изменению состояния линии nUARTDSR:<br>1 – установлена;<br>0 – сброшена |

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 2 | DCDMIM | Маска прерывания UARTDCDINTR по изменению состояния линии nUARTDCD:<br>1 – установлена;<br>0 – сброшена |
| 1 | CTSMIM | Маска прерывания UARTCTSINTR по изменению состояния линии nUARTCTS:<br>1 – установлена;<br>0 – сброшена |
| 0 | RIMIM | Маска прерывания UARTRIINTR по изменению состояния линии nUARTRI:<br>1 – установлена;<br>0 – сброшена |

### 20.21.11 *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) и бита прерывания по отсутствию данных в FIFO передатчика (бит 12), устанавливаются в 0. Значение бит прерывания по состоянию модема после сброса не определено.

Назначение бит регистра UARTRIS представлено ниже (Таблица 339).

Таблица 339 – Регистр UARTRIS

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31…14 | | Зарезервировано.<br>Не модифицируйте. При чтении выдаются нули |
| 13 | TNBSYRIS | Состояние прерывания по отсутствию данных в сдвиговом регистре передатчика UARTTNBSYINTR |
| 12 | TFERIS | Состояние прерывания по отсутствию данных в FIFO передатчика UARTTFEINTR |
| 11 | RNERIS | Состояние прерывания по наличию данных в FIFO приемника UARTRNEINTR |
| 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.<br>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 |

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

### 20.21.12 *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 ниже (Таблица 340).

Таблица 340 – Регистр UARTMIS

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31…14 | | Зарезервировано.<br>Не модифицируйте. При чтении выдаются нули |
| 13 | TNBSYMIS | Маскированное состояние прерывания по отсутствию данных в сдвиговом регистре передатчика UARTTNBSYINTR |
| 12 | TFEMIS | Маскированное состояние прерывания по отсутствию данных в FIFO передатчика UARTTFEINTR |
| 11 | RNEMIS | Маскированное состояние прерывания по наличию данных в FIFO приемника UARTRNEINTR |
| 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 |

### 20.21.13 *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 ниже (Таблица 341).

Таблица 341 – Регистр UARTICR

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31…11 | | Зарезервировано.<br>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 |

Примечание – Для организации надежного сброса статусных флагов прерываний необходимо учитывать требования, изложенные в пункте 21.1.2 «Обработчики исключений».

### 20.21.14 *DMACR*

*Регистр управления прямым доступом к памяти*

Регистр доступен по чтению и записи. После сброса, все биты регистра обнуляются. Назначение бит регистра UARTDMACR представлено ниже (Таблица 342).

Таблица 342 – Регистр UARTDMACR

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31…3 | | Зарезервировано.<br>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 |

## 20.22 Описание регистров контроллера I2C I2C_CNTR

Таблица 343 – Регистры контроллера I2C

| **Базовый Адрес** | **Название** | **Описание** |
|---|---|---|
| 0x400B_2000 | I2C | Контроллер I2C |
| **Смещение** | | |
| 0x00 | PRL | Младшая часть предделителя частоты |
| 0x04 | PRH | Старшая часть предделителя частоты |
| 0x08 | CTR | Управление контроллером I2C |
| 0x0C | RXD | Принятые данные по I2C |
| 0x10 | STA | Статус I2C |
| 0x14 | TXD | Передаваемые данные по I2C |
| 0x18 | CMD | Управление I2C |

### 20.22.1 *I2C->PRL*

Таблица 344 – Регистр PRL

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

Таблица 345 – 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 |

### 20.22.2 *I2C->PRH*

Таблица 346 – Регистр PRH

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

Таблица 347 - 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 |

### 20.22.3 *I2C->CTR*

Таблица 348 – Регистр CTR

| **Номер** | 31…8 | 7 | 6 | 5…0 |
|---|---|---|---|---|
| **Доступ** | U | R/W | R/W | U |
| **Сброс** | 0 | 0 | 0 | 0 |
| | **-** | **EN_I2C** | **EN_INT** | **-** |

Таблица 349 – 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:<br>0 – выключен;<br>1 – включен |
| 6 | EN_INT | Влияния не оказывает |
| 5…0 | - | Зарезервировано |

### 20.22.4 *I2C->RXD*

Таблица 350 – Регистр RXD

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

Таблица 351 – 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 |

### 20.22.5 *I2C->STA*

Таблица 352 – 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 353 – 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/NACK:<br>0 – получен ACK;<br>1 – получен NACK.<br>Бит устанавливается в 0 при потере арбитража |
| 6 | BUSY | Состояние шины I2C:<br>0 – после получения сигнала STOP;<br>1 – после получения сигнала START |
| 5 | LOST_ARB | Потеря арбитража:<br>0 – нет потери арбитража;<br>1 – потерян арбитраж.<br>Этот бит выставляется, если:<br>– получен сигнал STOP, но он не был инициирован этим контроллером;<br>– Если контроллер пытается выставить SDA в высокий уровень, но SDA остается в низком уровне.<br>Бит устанавливается в 0 после получения сигнала START |
| [4:2] | - | Зарезервировано |
| 1 | TR_PROG | Процесс передачи:<br>0 – передача завершена;<br>1 – передаются данные.<br>Бит является объединением битов RD и WR регистра CMD |
| 0 | INT | Флаг прерывания.<br>0 – нет прерывания;<br>1 – есть прерывание.<br>Флаг выставляется, если:<br>– завершена передача байта;<br>– был потерян арбитраж.<br>Прерывание для процессора выставляется, если установлен бит EN_INT |

### 20.22.6 *I2C->TXD*

Таблица 354 – Регистр TXD

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

Таблица 355 - 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.<br>При передаче первого байта адреса нулевой бит определяет режим передачи:<br>0 – запись в ведомое устройство;<br>1 – чтение из ведомого устройства |

### 20.22.7 *I2C->CMD*

Таблица 356 – 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 357 – Описание бит регистра CMD

| **№ бита** | **Функциональное имя бита** | **Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений** |
|---|---|---|
| [31:8] | - | Зарезервировано |
| 7 | START | Отправить сигнал START:<br>0 – нет действия;<br>1 – начать отправку сигнала START.<br>Передача START выполняется только при установленных битах WR или RD.<br>Сбрасывается автоматически при потере арбитража и после завершения передачи |
| 6 | STOP | Отправить сигнал STOP.<br>0 – нет действия;<br>1 – начать отправку сигнала STOP.<br>Сбрасывается автоматически при потере арбитража и после завершения передачи |
| 5 | RD | Чтение из ведомого устройства:<br>0 – нет действия;<br>1 – начать чтение.<br>Сбрасывается автоматически при потере арбитража и после завершения передачи. |
| 4 | WR | Запись в ведомое устройство:<br>0 – нет действия;<br>1 – начать запись.<br>Сбрасывается автоматически при потере арбитража и после завершения передачи. |
| 3 | ACK | Отправить ACK при чтении:<br>0 – отправить ACK;<br>1 – отправить NACK.<br>После завершения отправки NACK автоматически в 0 не сбрасывается, а очищается записью нуля. |
| [2:1] | - | Зарезервировано |
| 0 | CLR_INT | Очистить флаг INT.<br>Запись 1 очищает флаг |

## 20.23 Описание регистров контроллеров МКПД MIL_CNTR

Таблица 358 – Регистры контроллеров МКПД

| **Базовый адрес** | **Смещение** | **Название** | **Состояние после сброса** | **Описание** |
|---|---|---|---|---|
| 0x4008_7000 | | MILCNR1 | | Контроллер MIL1 управление |
| 0x4009_1000 | | MILCNR0 | | Контроллер MIL0 управление |

| Базовый адрес | Смещение | Название | Состояние после сброса | Описание |
|---|---|---|---|---|
| | | | | |
| | 0x0000 | CONTROL | | Регистр управления контроллером |
| | 0x0004 | STATUS | | Регистр состояния контроллера |
| | 0x0008 | ERROR | | Регистр ошибок контроллера |
| | 0x000C | CommandWord1 | | Регистр командного слова 1 |
| | 0x0010 | CommandWord2 | | Регистр командного слова 2 |
| | 0x0014 | ModeData | | Слово данных команды управления |
| | 0x0018 | StatusWord1 | | Регистр ответного слова 1 |
| | 0x001C | StatusWord2 | | Регистр ответного слова 2 |
| | 0x0020 | INTEN | | Регистр разрешения прерываний |
| | 0x0024 | MSG | | Регистр декодирования сообщений |
| | 0x1000-0x1FFC | DATA | | Регистр данных |

### 20.23.1 *CONTROL*

Таблица 359 – Регистр управления

| Base ADDR= | | | | | Offset= | | 0x0000 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | RT_HW | INPINV | AUTOTUNE | ENFILTER | INVTR | RERR | DIV[6:5] | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DIV[4:0] | | | | | RTA4 | RTA3 | RTA2 | RTA1 | RTA0 | TRB | TRA | RTMODE | BCMODE | BCSTART | MR |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| [31:24] | | | Зарезервировано |
| 23 | RT_HW | | Бит аппаратной поддержки ГОСТ 51765-2001 для режима ОУ.<br>1 – аппаратная поддержка разрешена<br>0 – аппаратная поддержка запрещена<br>При установке этого бита достаточно программной реализация циклического подадреса, установки и сброса признаков абонент занят, неисправность абонента, неисправность ОУ, запрос на обслуживание, а также задание адреса ОУ и бита паритета посредством внешнего соединения. |

| | | | |
|---|---|---|---|
| 22 | INPINV | | Бит инверсии входов приёмника PRMx+ и PRMx-<br>1 – инверсия разрешена;<br>0 – инверсия запрещена.<br>Вход PRMx+ инвертируется и коммутируется на отрицательный вход декодера манчестерского кода. Вход PRMx- инвертируется и коммутируется на положительный вход декодера манчестерского кода. Применение этого бита актуально для приёмопередатчиков с принудительной установкой выхода приемника в состояние логическая «1», например, 5559ИН74Т (HI1574) |
| 21 | AUTOTUNE | | Бит автоматической подстройки середины битовых интервалов (с ревизии 3)<br>0 – автоподстройка разрешена<br>1 – автоподстройка запрещена |
| 20 | ENFILTER | | Бит разрешения фильтрации потока NRZ (с ревизии 3)<br>1 – фильтрация разрешена<br>0 – фильтрация запрещена<br>В случае применения драйверов с некорректной скважностью и длительностью импульсов NRZ кода, таких как 5559ИН67Т, необходимо устанавливать бит в единицу для корректного приёма. В этом случае контроль длительностей импульсов NRZ не осуществляется |
| 19 | INVTR | | Разрешение инверсии сигналов (с ревизии 3)<br>PRD_PRMA, PRD_PRMB, PRD_PRMC, PRD_PRMD<br>1 – инверсия<br>0 – прямой выход |
| 18 | RERR | | Сброс ошибок в режиме ОУ<br>1 – ошибки могут быть сброшены только битом MR<br>0 – сброс ошибок происходит автоматически, после установки бита IDLE |
| 17..11 | DIV[6:0] | | Делитель частоты PCLK блока МКИО до 1 МГц<br>Содержит значение, на которое необходимо поделить частоту PCLK, чтобы получить 1 МГц. Частота PCLK обязательно должна быть не менее 24 МГц, не более 120 МГц и кратна 8. Если PCLK не кратна 8, то DIV[6:3] = (PCLK/8)+1, DIV[2:0] = 0, но стабильность приёма не гарантируется |
| [10:6] | RTA4-RTA0 | | Адрес оконечного устройства<br>Содержит адрес, который присвоен устройству, если контроллер работает в режиме оконечного устройства RTMODE=1; BCMODE=0 |
| 5 | TRB | | Блокировка передатчика резервного канала.<br>1 – передатчик разблокирован<br>0 – передатчик заблокирован |
| 4 | TRA | | Блокировка передатчика основного канала.<br>1 – передатчик разблокирован<br>0 – передатчик заблокирован |
| [3:2] | RTMODE<br>BCMODE | | Выбор режима работы контроллера<br>10 – режим оконечного устройства<br>01 – режим контроллера шины<br>11 – режим неадресуемого монитора |
| 1 | BCSTART | | Инициирует передачу сообщения в канал в режиме КШ.<br>1 – старт сообщения<br>0 – стоп сообщения<br>Сбрасывается в ноль автоматически по завершению сообщения |
| 0 | MR | | Сброс контроллера.<br>1 – контроллер сбрасывается в исходное состояние<br>0 – разрешение работы контроллера |

### 20.23.2 *STATUS*

Таблица 360 – Регистр состояния

| Base ADDR= | | | | | Offset= | | 0x0004 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| зарезервировано | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| зарезервировано | | | | | RCVB_stick | RCVA_stick | ERR_stick | VALMESS_stick | RFLAGN_stick | RCVB | RCVA | ERR | VALMESS | RFLAGN | IDLE |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| [31:11] | - | | Зарезервировано |
| 10 | RCVB_stick | | Признак активности резервного канала<br>0 – канал неактивен<br>1 – канал активен<br>Устанавливается при установке бита RCVB. Сбрасывается программной записью нуля, если RCVB=0.<br>В установленном состоянии не формирует прерывание. Необходимо следить за своевременным сбросом, иначе бит может относиться не к своему пакету |
| 9 | RCVA_stick | | Признак активности основного канала<br>0 – канал неактивен<br>1 – канал активен<br>Устанавливается при установке бита RCVA. Сбрасывается программной записью нуля, если RCVA=0.<br>В установленном состоянии не формирует прерывание. Необходимо следить за своевременным сбросом, иначе бит может относиться не к своему пакету |
| 8 | ERR_stick | | Ошибка в сообщении.<br>0 – нет ошибок<br>1 – в последней транзакции возникла ошибка<br>Устанавливается при установке бита ERR. Сбрасывается программной записью нуля, если ERR=0.<br>В установленном состоянии не формирует прерывание и не влияет на приём и передачу пакетов. Необходимо следить за своевременным сбросом, иначе бит может относится не к своему пакету |
| 7 | VALMESS_stick | | Успешное завершение транзакции в канале.<br>0 – транзакция завершена с ошибкой, либо транзакции нет в канале<br>1 – транзакция завершена успешно<br>Устанавливается при установке бита VALMESS. Сбрасывается программной записью нуля, если VALMESS =0.<br>В установленном состоянии не формирует прерывание. Необходимо следить за своевременным сбросом, иначе бит может относиться не к своему пакету |

| | | | |
|---|---|---|---|
| 6 | RFLAGN_stick | | Получено достоверное слово из канала.<br>0 – нет достоверных слов в канале<br>1 – в режиме КШ получено достоверное ответное слово<br>1 – в режиме ОУ или М получено достоверное командное слово, ответное слово или слово данных в команде управления<br>Устанавливается при установке бита RFLAGN. Сбрасывается программной записью нуля, если RFLAGN =0.<br>В установленном состоянии не формирует прерывание.<br>Необходимо следить за своевременным сбросом, иначе бит может относиться не к своему пакету |
| 5 | RCVB | | Признак активности резервного канала<br>0 – канал неактивен<br>1 – канал активен |
| 4 | RCVA | | Признак активности основного канала<br>0 – канал неактивен<br>1 – канал активен |
| 3 | ERR | | Ошибка в сообщении.<br>0 – нет ошибок<br>1 – в последней транзакции возникла ошибка<br>В режиме ОУ и М, если сброшен бит RERR, автоматически сбрасывается не менее чем через 4 мкс после установки |
| 2 | VALMESS | | Успешное завершение транзакции в канале.<br>0 – транзакция завершена с ошибкой, либо транзакции нет в канале<br>1 – транзакция завершена успешно<br>В режиме ОУ и М автоматически сбрасывается не менее чем через 4 мкс после установки |
| 1 | RFLAGN | | Получено достоверное слово из канала.<br>0 – нет достоверных слов в канале<br>1 – в режиме КШ получено достоверное ответное слово<br>1 – в режиме ОУ или М получено достоверное командное слово, ответное слово или слово данных в команде управления<br>Между сообщениями бит автоматически сбрасывается в ноль |
| 0 | IDLE | | Состояние контроллера.<br>1 – контроллер в неактивном состоянии<br>0 – контроллер в состоянии обмена сообщениями |

Примечание – Для организации надежного сброса статусных флагов прерываний необходимо учитывать требования, изложенные в пункте 21.1.2 «Обработчики исключений».

### 20.23.3 *ERROR*

Таблица 361 – Регистр ошибок

| Base ADDR= | | | | | Offset= | | 0x0008 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| зарезервировано | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| зарезервировано | | | | | | | | | PROERR | CONERR | GAPERR | CSYCERR/ SEQERR | DSYCERR/ SYNCERR | MANERR | NORCV |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31..7 | | | Зарезервировано |
| 6 | PROERR | | Ошибка в протоколе.<br>1 – недопустимое слово обнаружено на шине во время обмена сообщениями<br>0 – нет ошибок |
| 5 | CONERR | | Ошибка непрерывности сообщения.<br>1 – передача сообщения не непрерывная<br>0 – нет ошибок |
| 4 | GAPERR | | Недопустимая активность на шине.<br>1 – обнаружена активность на шине в интервале 4 мкс после успешного завершения сообщения<br>0 – нет ошибок |
| 3 | CSYCERR/ SEQERR | | Ошибка синхронизации команды в режиме КШ (CSYCERR).<br>1 – ожидался синхроимпульс команды, а получен синхроимпульс данных<br>0 – ошибок нет<br>Ошибка после приёма команды в режиме ОУ и М (SEQERR).<br>1 – обнаружена пауза после приёма командного слова с битом 10 равным нулю или обнаружены слова данных после приёма командного слова с битом 10 равным единице<br>0 – ошибок нет |
| 2 | DSYCERR/ SYNCERR | | Ошибка синхронизации данных в режиме КШ (DSYCERR).<br>1 – ожидался синхроимпульс данных, а получен синхроимпульс команды<br>0 – ошибок нет<br>Ошибка синхронизации в режиме ОУ и М (SYNCERR).<br>1 – ожидался синхроимпульс команды, а получен синхроимпульс данных или наоборот<br>0 – ошибок нет |
| 1 | MANERR | | Ошибка декодирования NRZ кода.<br>1 – ошибка в количестве принятых бит или ошибка в бите контроля чётности<br>0 – ошибок нет |
| 0 | NORCV | | Ошибка приёма.<br>1 – не получено ответное слово в интервале 14 мкс или не получены ожидаемые данные<br>0 – ошибок нет |

### 20.23.4 *CommandWord1*

Таблица 362 – Регистр команды 1

| Base ADDR= | | | | | Offset= | | 0x000C | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Адрес ОУ | | | | | Приём/ Передача | Подадрес/ Режим управления | | | | | Число СД/ Код команды | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31..16 | - | | Зарезервировано |
| 15..11 | Адрес ОУ | | Адрес оконечного устройства, которому предназначено командное слово |
| 10 | Приём/передача | | Бит приёма/передачи<br>1 – режим работы ОУ-КШ<br>0 – режим работы КШ-ОУ |
| 9..5 | Подадрес / Режим управления | | Содержит подадрес, по которому в памяти располагаются принимаемые или передаваемые СД. В случае передачи команды, содержит код 00000 или 11111 |
| 4..0 | Число СД / Код команды | | Содержит количество принимаемых или передаваемых слов данных. В случае передачи команды содержит код команды из Таблицы 1 ГОСТ Р52070-2003 |

Примечание – В режиме ОУ и М регистр доступен только на чтение, в режиме КШ – только на запись.

### 20.23.5 *CommandWord2*

Таблица 363 – Регистр команды 2

| Base ADDR= | | | | | Offset= | | 0x0010 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Адрес ОУ | | | | | 1 | Подадрес/<br>Режим управления | | | | | Число СД/<br>Код команды | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31..16 | | | Зарезервировано |
| 15..11 | Адрес ОУ | | Адрес оконечного устройства, которому предназначено командное слово |
| 10 | Приём/передача | | Бит приёма/передачи<br>1 – режим работы ОУ-ОУ<br>0 – командное слово не используется |
| 9..5 | Подадрес | | Содержит подадрес, по которому в памяти располагаются принимаемые или передаваемые СД |
| 4..0 | Число СД | | Содержит количество принимаемых или передаваемых слов данных |

Примечание – В режиме ОУ и М регистр доступен только на чтение и содержит второе командное слово транзакции ОУ-ОУ. В режиме КШ регистр доступен только на запись и используется для транзакции ОУ-ОУ, если установлен в единицу бит 10.

### 20.23.6 *ModeData*

Таблица 364 – Слово данных команды управления

| Base ADDR= | | | | | Offset= | | 0x0014 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Слово данных команды управления | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31..16 | | | Зарезервировано |
| 15..0 | ModeData | | Содержит принятое или передаваемое слово данных в команде управления |

### 20.23.7 *StatusWord1*

Таблица 365 – Ответное слово 1

| Base ADDR= | | | | | Offset= | | 0x0018 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Адрес ОУ | Ошибка в сообщ. | Пер.ОС | Запр. на обсл. | - | - | - | Прин. ГК | Абон. зан. | Неисп. абон. | Прин. упр. инт. | Неисп. ОУ |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31..16 | | | Зарезервировано |
| 15..11 | Адрес ОУ | | Адрес ОУ, от которого принято ответное слово в режиме КШ. Адрес ОУ, которое передаёт ответное слово в режиме ОУ |
| 10 | Ошибка в сообщ. | | Ошибка в сообщении |
| 9 | Пер.ОС | | Передача ответного слова |
| 8 | Запр. на обсл. | | Запрос на обслуживание |
| 7-5 | | | Зарезервировано |
| 4 | Прин. ГК | | Принята групповая команды |
| 3 | Абон. зан. | | Абонент занят |
| 2 | Неисп. абон. | | Неисправность абонента |
| 1 | Прин. упр. инт. | | Принято управление интерфейсом |
| 0 | Неисп. ОУ | | Неисправность ОУ |

Примечание – В режиме ОУ регистр по записи содержит предназначенное для передачи КШ ответное слово, а по чтению содержит ОС, полученное от передающего ОУ в транзакции ОУ-ОУ. В режиме КШ и М регистр доступен только на чтение и содержит принятое от ОУ ответное слово.

### 20.23.8 *StatusWord2*

Таблица 366 – Ответное слово 2

| Base ADDR= | | | | | Offset= | | 0x001C | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Адрес ОУ | | | | | Ошибка в сообщ. | Пер. ОС | Запр. на обсл. | | | | Прин. ГК | Абонзан. | Неисп. абон. | Прин упр. инт. | Неисп ОУ |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31..16 | | | Зарезервировано |
| 15..11 | Адрес ОУ | | Адрес ОУ, передающего данные в транзакции ОУ-ОУ |
| 10 | Ошибка в сообщ. | | Ошибка в сообщении |
| 9 | Пер. ОС | | Передача ответного слова |
| 8 | Запр. на обсл. | | Запрос на обслуживание |
| 7-5 | | | Зарезервировано |
| 4 | Прин. ГК | | Принята групповая команды |
| 3 | Абон. зан. | | Абонент занят |
| 2 | Неисп. абон. | | Неисправность абонента |
| 1 | Прин. упр. инт. | | Принято управление интерфейсом |
| 0 | Неисп. ОУ | | Неисправность ОУ |

Примечание – В режиме КШ и М регистр доступен только на чтение и содержит ОС, передающего ОУ в транзакции ОУ-ОУ. В режиме ОУ регистр не используется.

### 20.23.9 *INTEN*

Таблица 367 – Регистр разрешения прерываний

| Base ADDR= | | | | | Offset= | | 0x0020 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | ERRIE | VALMESSIE | RFLAGNIE | IDLEIE |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31..4 | | | Зарезервировано |
| 3 | ERRIE | | Прерывание при возникновении ошибки в сообщении.<br>0 – прерывание маскировано<br>1 – прерывание разрешено, это позволяет генерировать прерывание при возникновении ошибок в сообщении |
| 2 | VALMESSIE | | Прерывание при успешном завершении транзакции в канале.<br>0 – прерывание маскировано<br>1 – прерывание разрешено, это позволяет генерировать прерывание при успешном завершении обмена данными в канале |
| 1 | RFLAGNIE | | Прерывание при приёме достоверного слова.<br>0 – прерывание маскировано<br>1 – прерывание разрешено, это позволяет генерировать прерывание при приёме достоверного ОС в режиме КШ или достоверного КС, ОС или слова данных команды управления в режиме ОУ |
| 0 | IDLEIE | | Прерывание неактивности контроллера.<br>0 – прерывание маскировано<br>1 – прерывание разрешено, это позволяет генерировать прерывание по переходу контроллера в неактивное состояние |

### 20.23.10 *MSG*

Таблица 368 – Регистр декодирования сообщений

| Base ADDR= | | | | | Offset= | | 0x0024 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Код сообщения | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…14 | - | | Зарезервировано |
| 13…0 | Код сообщения | | Регистр содержит код сообщения, полученного в режиме ОУ или М, и доступен только на чтение. В режиме КШ регистр не используется. Регистр обновляется каждый раз при получении нового достоверного КС |

Таблица 369 – Коды сообщений регистра MSG

| Код сообщения | CommandWord1 | CommandWord2 |
|---|---|---|
| ***Команды обмена данными*** | 15:11 10 9:5 4:0 | 15:11 10 9:5 4:0 |
| 0001 Команда приёма КШ-ОУ, не групповая | RTA 0 00001-11110 XXXXX | |
| 0080 Команда приёма КШ-ОУ, групповая | 11111 0 00001-11110 XXXXX | |
| 0004 Команда приёма ОУ-ОУ, не групповая | RTA 0 00001-11110 XXXXX | XXXXX 1 00001-11110 XXXXX |
| 0100 Команда приёма ОУ-ОУ, групповая | 11111 0 00001-11110 XXXXX | не RTA 1 00001-11110 XXXXX |
| 0402 Команда передачи ОУ-КШ | RTA 1 00001-11110 XXXXX | |
| 1008 Команда передачи ОУ-ОУ, не групповая | не F 0 00001-11110 XXXXX | не F 0 00001-11110 XXXXX |
| 0200 Команда передачи ОУ-ОУ, групповая | 11111 0 00001-11110 XXXXX | RTA 1 00001-11110 XXXXX |
| ***Команда управления*** | | |
| 0410 Код 0-15 K=1 нет данных, не групповая | RTA 1 00000 11111 0XXXX | |
| 0400 Код 0-15 K=1 нет данных, групповая | 11111 1 00000 11111 0XXXX | |
| 2420 Код 16-31 K=1 с данными, не групповая | RTA 1 00000 11111 1XXXX | |
| 0040 Код 16-31 K=0 с данными, не групповая | RTA 0 00000 11111 1XXXX | |
| 0800 Код 16-31 K=0 с данными, групповая | 1111 0 00000 11111 1XXXX | |

### 20.23.11 *DATA*

Таблица 370 – Память принимаемых/передаваемых данных

| Base ADDR= | | | | | Offset= | | 0x1000-0x1FFC | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA[15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…16 | - | | Зарезервировано |
| 15…0 | DATA | | Данные читаются из памяти или записываются в память в соответствии с подадресом (биты с 9 по 5) достоверного командного слова. Каждому подадресу соответствует 32x16 ячеек памяти на приём и 32x16 ячеек памяти на передачу. Общий объём памяти данных 2Kx16 в рамках двух буферов приёмника и передатчика (по размеру 1Kx16 на каждый буфер).<br>Когда производится запись в регистр DATA, то работа ведётся с буфером передатчика.<br>Когда производится считывание регистра DATA, то работа ведётся с буфером приёмника. |

## 20.24 Описание регистров контроллеров формирования опорных напряжений и токов ADC_BG_CNTR и ANA_BG_CNTR

### 20.24.1 *ADC_BG_CNTR*

Адрес: 0x400A_702C

Таблица 371 – Регистр ADC_BG_CNTR

| **Номер** | 31…22 | 21 | 20 | 19 | 18 | 17, 16 |
|---|---|---|---|---|---|---|
| **Доступ** | R/W | R/W | R/W | R/W | R/W | R/W |
| **Сброс** | 0 | 0 | 0 | 0 | 0 | 0 |
| | - | ADC_BG_ZMODE | ADC_BG_VRECMODE | ADC_BG_IRECMODE | ADC_BG_EXTMODE | ADC_BG_SWMODE[5:4] |

| **Номер** | 15…12 | 11…8 | 7 | 6…1 | 0 |
|---|---|---|---|---|---|
| **Доступ** | R/W | R/W | R/W | R/W | R/W |
| **Сброс** | 0 | 0 | 0 | 0 | 0 |
| | ADC_BG_SWMODE[3:0] | ADC_BG_BFEXT[5:2] | ADC_BG_BGEN | ADC_BG_BFEN[5:0] | ADC_BG_IREFEN |

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:22] | - | Зарезервировано |
| 21 | ADC_BG_ZMODE | Отключение выхода BG от выходных буферов<br>0 – Выход BG подключен (нормальный режим работы);<br>1 – Выход BG отключен (Z состояние) |
| 20 | ADC_BG_VRECMODE | Включение аварийного источника напряжения<br>0 – нормальная работа (нормальный режим работы);<br>1 – аварийная работа |
| 19 | ADC_BG_IRECMODE | Включение аварийного источника тока<br>0 – нормальная работа (нормальный режим работы);<br>1 – аварийная работа |
| 18 | ADC_BG_EXTMODE | Переключение ядра блока на внешнюю опору<br>0 – выход генерируется ядром BG (нормальный режим работы);<br>1 – выход берется с внешней опоры |
| [17:12] | ADC_BG_SWMODE[5:0] | Управление режимом работы буферов<br>0 – 1,25 В;<br>1 – 2,5 В |
| [11:8] | ADC_BG_BFEXT[5:2] | Подключение внешних буферов<br>0 – внешний буфер отключен (нормальный режим работы);<br>1 – внешний буфер подключен |
| 7 | ADC_BG_BGEN | Включение блока BG<br>0 – выключен<br>1 – включен |
| [6:1] | ADC_BG_BFEN[5:0] | Включение буферов<br>0 – выключен<br>1 – включен |
| 0 | ADC_BG_IREFEN | Сигнал разрешения работы встроенного источника опорного тока:<br>0 –выключен;<br>1 –включен |

### 20.24.2 *ANA_BG_CNTR*

Адрес: 0x400AE008

Таблица 372 – Регистр ANA_BG_CNTR

| **Номер** | 31…22 | 21 | 20 | 19 | 18 | 17, 16 |
|---|---|---|---|---|---|---|
| **Доступ** | R/W | R/W | R/W | R/W | R/W | R/W |
| **Сброс** | 0 | 0 | 0 | 0 | 0 | 0 |
| | - | ANA_BG_ ZMODE | ANA_BG_ VRECMODE | ANA_BG_ IRECMODE | ANA_BG_ EXTMODE | ANA_BG_ SWMODE[5:4] |

| **Номер** | 15…12 | 11…8 | 7 | 6…1 | 0 |
|---|---|---|---|---|---|
| **Доступ** | R/W | R/W | R/W | R/W | R/W |
| **Сброс** | 0 | 0 | 0 | 0 | 0 |
| | ANA_BG_ SWMODE[3:0] | ANA_BG_ BFEXT[5:2] | ANA_BG_ BGEN | ANA_BG_ BFEN[5:0] | ANA_BG_ IREFEN |

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:22] | - | Зарезервировано |
| 21 | ANA_BG_ZMODE | Отключение выхода BG от выходных буфером<br>0 – Выход BG подключен (нормальный режим работы);<br>1 – Выход BG отключен (Z состояние); |
| 20 | ANA_BG_VRECMODE | Включение аварийного источника напряжения<br>0 – нормальная работа (нормальный режим работы);<br>1 – аварийная работа |

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 19 | ANA_BG_IRECMODE | Включение аварийного источника тока<br>0 – нормальная работа (нормальный режим работы);<br>1 – аварийная работа |
| 18 | ANA_BG_EXTMODE | Переключение ядра блока на внешнюю опору<br>0 – выход генерируется ядром BG (нормальный режим работы);<br>1 – выход берется с внешней опоры. |
| [17:12] | ANA_BG_SWMODE[5:0] | Управление режимом работы буферов<br>0 – 1,25В;<br>1 – 2,5В. |
| [11:8] | ANA_BG_BFEXT[5:2] | Подключение внешних буферов<br>0 – внешний буфер отключен (нормальный режим работы);<br>1 – внешний буфер подключен |
| 7 | ANA_BG_BGEN | Включение блока BG<br>0 – выключен<br>1 – включен |
| [6:1] | ANA_BG_BFEN[5:0] | Включение буферов<br>0 – выключен<br>1 – включен |
| 0 | ANA_BG_IREFEN | Сигнал разрешения работы встроенного источника опорного тока:<br>0 – выключен;<br>1 – включен |

## 20.25 Описание регистров блока контроллера компаратора COMP_CNTR

### 20.25.1 *COMP_CNTR*

**Адреса:**
**COMP0_CNTL – 0x400AE000**
**COMP1_CNTL – 0x400AF000**
**COMP2_CNTL – 0x400B0000**
**COMP3_CNTL – 0x400B1000**

| Номер | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | 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 | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | Зарезервировано | | | | | | | | | | | | | | | |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |

| | Зарезервировано | CMP_CLREN | CMP_SELP | CMP_SELN | INV | CMP_ON | CMP_IE | CMP_PE |
|---|---|---|---|---|---|---|---|---|

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:5] | Зарезервировано | Зарезервировано |
| 8 | CMP_CLREN | Бит разрешения очистки регистра прерываний при чтении.<br>0 - не разрешено;<br>1 – разрешено |
| [7:6] | CMP_SELP | Выбор источника канала P<br>00 – канал 1;<br>01 – канал 2;<br>10 – канал 3;<br>11 – выход ЦАП (DACx -> COMPx) |
| [5:4] | CMP_SELN | Выбор источника канала N<br>00 – канал 1;<br>01 – канал 2;<br>10 – канал 3;<br>11 – выход ЦАП (DACx -> COMPx) |
| 3 | INV | Инверсия сигнала сравнения от компаратора перед дальнейшей обработкой:<br>0 – прямой сигнал;<br>1 – инвертированный |
| 2 | CMP_ON | Включение компаратора<br>0 – компаратор выключен;<br>1 – компаратор включен |
| 1 | CMP_IE | Разрешение прерывания от компаратора<br>0 – запрещено;<br>1 – разрешено |
| 0 | CMP_PE | Разрешение сигнала ошибки ШИМ от компаратора<br>0 – запрещено;<br>1 – разрешено |

### 20.25.2 *COMP_EVNT*

**Адреса:**

**COMP0_EVNT – 0x400AE004**

**COMP1_EVNT– 0x400AF004**

**COMP2_EVNT – 0x400B0004**

**COMP3_EVNT – 0x400B1004**

| Номер | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | 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 | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | Зарезервировано | | | | | | | | | | | | | | | |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |

| | зарезервировано | RST | CMP_RES_IS2 | CMP_RES_IS | CMP_RES_I |
|---|---|---|---|---|---|

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:4] | Зарезервировано | Зарезервировано |
| 3 | RST | Бит очистки прерываний от компаратора.<br>При записи 1 очищаются прерывания CMP_RES_IS2.<br>Читается 0 |
| 2 | CMP_RES_IS2 | Результат сравнения компаратора зафиксированный в триггере 2 |
| 1 | CMP_RES_IS | Результат сравнения компаратора, подсинхронизированный по синхросигналу PCLK |
| 0 | CMP_RES_I | Результат сравнения компаратора |

## 20.26 Описание регистров контроллера ЦАП DAC_CNTR

Таблица 373 – Регистры контроллера ЦАП

| Базовый Адрес | Название | Описание |
|---|---|---|
| 0x400A_A000 | DAC0_CNTR | Контроллер DAC0 |
| 0x400A_B000 | DAC1_CNTR | Контроллер DAC1 |
| 0x400A_C000 | DAC2_CNTR | Контроллер DAC2 |
| 0x400A_D000 | DAC3_CNTR | Контроллер DAC3 |
| **Смещение** | | |
| 0x00 | DAC_CNTR | Регистр управления блоком DAC |
| 0x04 | FIFO_CNTR | Регистр управления/статуса FIFO DAC |
| 0x08 | DAC_PRD | Регистр периода обновления данных DAC |
| 0x0C | DAC_CNT | Счетчик периода обновления данных DAC |
| 0x10 | DAC_IE | Регистр разрешения прерываний DAC |
| 0x14 | DAC_RE | Регистр разрешения запросов DMA от DAC |
| 0x18 | DAC_STS | Регистр статуса DAC |
| 0x1C | DAC_DATA | Регистр FIFO данных DAC |

### 20.26.1 *DAC_CNTR*

| Base ADDR= | 0x400A_A000<br>0x400A_B000<br>0x400A_C000<br>0x400A_D000 | Offset= | 0x0000_0000 | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | DAC_CNTR | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | | | | | | | | | | | | | | | |
| U | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | | | | | | | DAC_RST | - | DAC_MODE | | | DAC_REF | | DAC_EN_D | DAC_EN_A |
| U | | | | | | | R/W | U | R/W | | | R/W | | R/W | R/W |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…9 | - | - | Зарезервировано |
| 8 | DAC_RST | 0 | Бит сброса регистров цифрового контроллера ЦАП в начальное значение и сброс буфера FIFO.<br>Запись 1 приводит к сбросу регистров цифрового контроллера ЦАП.<br>Возможна только установка в 1.<br>Самоочищающийся бит. |
| 7 | - | - | Зарезервировано |
| 6:4 | DAC_MODE | 2'b00 | Выбор режима работы ЦАП:<br>3'b000 - Режим задания одиночного значения. В этом режиме данные постоянно записываются и считываются из нулевой ячейки FIFO. Обновления значения на выходе блока происходит только при записи нового значения. При этом происходит обновление счетчика периода. Возможна организация одиночных преобразований, при которой при получении прерывания о достижении периода счета DAC в буфер заносится новое значение для преобразования.<br>3'b001 - Режим FIFO с периодическим обновлением. В этом режиме данные записываются в очередь буфера FIFO размером в 32 значения. Данные на выходе ЦАП обновляются при достижении счетчиком периода обновления данных DAC_CNT значения, задаваемого регистром DAC_PRD. При достижении последнего записанного в буфер FIFO значения обновление значений на выходе ЦАП прекращается.<br>3'b010 - Режим FIFO с обновлением данных от внешнего сигнала. В этом режиме данные также записываются в очередь буфера FIFO. Выдача очередного значения происходит по событию фронта сигнала внешней синхронизации DAC_SYNC.<br><br>Режим 3'b011 аналогичен режиму 3'b010. Режимы 3'b1xx аналогичны режимам 3'b0xx с тем отличием, что при достижении последнего записанного в буфер FIFO значения обновление значений на выходе ЦАП начинается с нулевого значения (циклично). |
| 3:2 | DAC_REF | 2'b00 | Вход выбора источника опорного напряжения:<br>00: $V_{CC_DAC}$ и GND;<br>01: VRFP0 и VRFN0;<br>10: VRFP1 и VRFN1;<br>11: VREF (формируется с ANA_BG) и GND. |
| 1 | DAC_EN_D | 0 | Разрешение работы цифрового контроллера ЦАП<br>0 – работа запрещена;<br>1 – работа разрешена. |
| 0 | DAC_EN_A | 0 | Разрешение работы аналогового преобразователя ЦАП<br>0 – работа запрещена;<br>1 – работа разрешена. |

### 20.26.2 *FIFO_CNTR*

| Base ADDR= | 0x400A_A000<br>0x400A_B000<br>0x400A_C000<br>0x400A_D000 | Offset= | 0x0000_0004 | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|

| REG Name: | FIFO_CNTR | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | | | | | | | | | | | | | | | |
| U | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | | FIFO_LVL | | | | | | - | | FIFO_CNT | | | | | |
| U | | R/W | | | | | | U | | RO | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…14 | - | - | Зарезервировано |
| 13…8 | FIFO_LVL | 6'b000000 | Уровень сравнения заполнения буфера FIFO. |
| 7…6 | - | - | Зарезервировано |
| 5…0 | FIFO_CNT | 6'b000000 | Счетчик количества запись в буфере FIFO. |

### 20.26.3 *DAC_PRD*

| Base ADDR= | 0x400A_A000<br>0x400A_B000<br>0x400A_C000<br>0x400A_D000 | Offset= | 0x0000_0008 | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | DAC_PRD | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DAC_PRD[31:16] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DAC_PRD[15:0] | | | | | | | | | | | | | | | |
| R/W | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | DAC_PRD | 0x0000_0000 | Регистр периода обновления данных ЦАП |

### 20.26.4 *DAC_CNT*

| Base ADDR= | 0x400A_A000<br>0x400A_B000<br>0x400A_C000<br>0x400A_D000 | Offset= | 0x0000_000C | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | DAC_CNT | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DAC_CNT[31:16] | | | | | | | | | | | | | | | |
| RO | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DAC_CNT[15:0] | | | | | | | | | | | | | | | |
| RO | | | | | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | DAC_CNT | 0x0000_0000 | Значение счетчика периода обновления данных ЦАП.<br>Счетчик инкрементирует на частоте PCLK от 0 до DAC_PRD. |

### 20.26.5 *DAC_IE*

| Base ADDR= | | 0x400A_A000<br>0x400A_B000<br>0x400A_C000<br>0x400A_D000 | | | Offset= | | 0x0000_0000 | | | Reset= | | | 0x0000_0000 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | DAC_IE | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | | | | | | | | | | | | | | | |
| U | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | | | | | | | | | | | | FIFO_NFULL_IE | FIFO_EMPTY_IE | FIFO_LIM_IE | DAC_INT_IE |
| U | | | | | | | | | | | | R/W | R/W | R/W | R/W |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…4 | - | - | Зарезервировано |
| 3 | FIFO_NFULL_IE | 0 | Разрешение формирования запроса прерывания, если FIFO данных ЦАП не заполнено.<br>0 – запрещено;<br>1 – разрешено. |
| 2 | FIFO_EMPTY_IE | 0 | Разрешение формирования запроса прерывания, если FIFO данных ЦАП пусто.<br>0 – запрещено;<br>1 – разрешено. |
| 1 | FIFO_LIM_IE | 0 | Разрешение формирования запроса прерывания, если FIFO данных ЦАП заполнено ниже уровня сравнения.<br>0 – запрещено;<br>1 – разрешено. |
| 0 | DAC_INT_IE | 0 | Разрешение формирования запроса прерывания INT_DAC по событию DAC_INT.<br>0 – запрещено;<br>1 – разрешено. |

### 20.26.6 *DAC_RE*

| Base ADDR= | | 0x400A_A000<br>0x400A_B000<br>0x400A_C000<br>0x400A_D000 | | | Offset= | | 0x0000_0014 | | | Reset= | | | 0x0000_0000 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | DAC_RE | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | | | | | | | | | | | | | | | |
| U | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | | | | | | | | | | | | FIFO_NFULL_RE | FIFO_EMPTY_RE | FIFO_LIM_RE | CNT_PRD_RE |
| U | | | | | | | | | | | | R/W | R/W | R/W | R/W |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…4 | - | - | Зарезервировано |
| 3 | FIFO_NFULL_RE | 0 | Разрешение формирования запросов к DMA если FIFO данных ЦАП не заполнено.<br>0 – запрещено;<br>1 – разрешено. |
| 2 | FIFO_EMPTY_RE | 0 | Разрешение формирования запросов к DMA если FIFO данных ЦАП пусто.<br>0 – запрещено;<br>1 – разрешено. |
| 1 | FIFO_LIM_RE | 0 | Разрешение формирования запросов к DMA, если FIFO данных ЦАП заполнено ниже уровня сравнения.<br>0 – запрещено;<br>1 – разрешено. |
| 0 | CNT_PRD_RE | 0 | Разрешение формирования запросов к DMA при достижении периода счетчика обновления данных ЦАП.<br>0 – запрещено;<br>1 – разрешено. |

### 20.26.7 *DAC_STS*

| Base ADDR= | 0x400A_A000<br>0x400A_B000<br>0x400A_C000<br>0x400A_D000 | Offset= | 0x0000_0018 | Reset= | 0x0000_0004 |
|---|---|---|---|---|---|

| REG Name: | DAC_STS | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | | | | | | | | | | | | | | | |
| U | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | | | | | | | | | | | | FIFO_FULL | FIFO_EMPTY | FIFO_LIM | DAC_INT |
| U | | | | | | | | | | | | RO | RO | RO | R/W |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…4 | - | - | Зарезервировано |
| 3 | FIFO_FULL | 0 | Флаг заполнения FIFO данных ЦАП. Возникновение флага не зависит от разрешения в регистрах DAC_IE и DAC_RE.<br>0 – нет события;<br>1 – есть событие (FIFO заполнено). |
| 2 | FIFO_EMPTY | 1 | Флаг опустошения FIFO данных DAC. Возникновение флага не зависит от разрешения в регистрах DAC_IE и DAC_RE.<br>0 – нет события;<br>1 – есть событие (FIFO пусто). |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 1 | FIFO_LIM | 0 | Флаг заполнения FIFO данных ЦАП ниже уровня сравнения. Возникновение флага не зависит от разрешения в регистрах DAC_IE и DAC_RE.<br>0 – нет события;<br>1 – есть событие (FIFO_CNT < FIFO_LIM). |
| 0 | DAC_INT | 0 | Флаг события DAC_INT. Зависит от режима работы DAC_MODE и разрешения в регистре DAC_IE.<br>0 – нет события:<br>1 – есть событие.<br>В режиме с обновлением данных от внешнего сигнала событие возникает по фронту внешнего сигнала.<br>При записи 1 в данный бит регистра формируется немаскируемый сигнал запроса на прерывание. |

### 20.26.8 *DAC_DATA*

| Base ADDR= | 0x400A_A000<br>0x400A_B000<br>0x400A_C000<br>0x400A_D000 | Offset= | 0x0000_001C | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|

| REG Name: | DAC_DATA | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| - | | | | | | | | | | | | | | | |
| U | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| - | | | | DAC_DATA | | | | | | | | | | | |
| U | | | | R/W | | | | | | | | | | | |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…12 | - | - | Зарезервировано |
| 11…0 | DAC_DATA | 0 | Данные для преобразователя ЦАП.<br>Пишется по адресу следующей свободной ячейки FIFO.<br>Читается по адресу последней считанной ячейки FIFO.<br>Размер FIFO – 32 значения.<br>В режиме задания одиночного значения данные постоянно записываются и считываются из нулевой ячейки FIFO. |

## 20.27 Описание регистров контроллеров АЦП ADC_CNTR

Таблица 374 – Регистры контроллеров АЦП

| Базовый Адрес | Название | Описание |
|---|---|---|
| 0x400A_7000 | ADC0_BASE | Контроллер АЦП0 |
| 0x400A_8000 | ADC1_BASE | Контроллер АЦП1 |
| 0x400A_9000 | ADC2_BASE | Контроллер АЦП2 |
| **Смещение** | | |
| 0x000 | ADC0_CNTR | Регистр управления контроллером АЦП0 |
| 0x004 | ADC1_CNTR | Регистр управления контроллером АЦП1 |
| 0x008 | ADC0_SCOPE_CNTR | Регистр управления границами преобразований АЦП0 |
| 0x00C | ADC1_SCOPE_CNTR | Регистр управления границами преобразований АЦП1 |
| 0x010 | ADC_SYNC_CNTR | Управление режимом синхронизации преобразований АЦП0 и АЦП1 |
| 0x014 | ADC_BUF_CNTR | Регистр управления режимом записи результатов АЦП0 и АЦП1 |

| | | |
|---|---|---|
| 0x018 | ADC_BUF_STATE | Состояние буферов АЦП0 и АЦП1 |
| 0x01C | ADC_INT_CNTR | Состояние буферов АЦП0 и АЦП1 |
| 0x020 | ADC_DMA_CNTR | Регистр управления АЦП0 |
| 0x024 | - | Зарезервировано |
| 0x028 | ADC_ ANALOG_CNTR | Регистр управления АЦП1 |
| 0x02C | ADC_BG_CNTR | Регистр управления источником опорного напряжения.<br>Доступен только в блоке ADC0 по адресу 0x400A_702C. Настройки задаваемые в регистре применимы ко всей сборке АЦП – ADC0, ADC1, ADC2. |
| 0x030 | - | Зарезервировано |
| 0x040 | ADC0_CHSEL | Выбор входного n- или p- канала из диапазона 0-8 для АЦП0 |
| 0x060 | ADC1_CHSEL | Выбор входного n- или p- канала из диапазона 0-8 для АЦП1 |
| 0x080<br>...<br>0x0FC | ADC0_RESULT0<br>…<br>ADC0_RESULT31 | Адреса результатов преобразований АЦП0:<br>0x0080 – первый результат,<br>…<br>0x00FC – последний результат |
| 0x100<br>…<br>0x17C | ADC1_RESULT0<br>…<br>ADC1_RESULT31 | Адреса результатов преобразований АЦП1:<br>0x0100 – первый результат,<br>…<br>0x017C – последний результат |

### 20.27.1 *ADC0_CNTR*

Таблица 375 – Регистр ADC0_CNTR

| **Номер** | 31..24 | 23..20 | 19 | 18..14 | 13..6 |
|---|---|---|---|---|---|
| **Доступ** | R/W | R/W | - | R/W | R/W |
| **Сброс** | 0 | 0 | - | 0 | 0 |
| **Имя** | DELAY_TIME_ADC0 | ALT_DEL_CHN_P | - | MAXCNV_ADC0 | CONV_TRIG_CNTR_ADC0 |

| **Номер** | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|
| **Доступ** | - | R/W | R/W | - | R/W | R/W |
| **Сброс** | - | 0 | 0 | - | 0 | 0 |
| **Имя** | - | ETRNL_CONV_MODE_ADC0 | STARTofCONV_ADC0 | - | RST_ADC0 | EN_ADC0 |

Таблица 376 – Описание бит регистра ADC0_CNTR

| **№ бита** | **Функциональное имя бита** | **Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений** |
|---|---|---|
| 31..24 | DELAY_TIME_ADC0 | Время заряда емкости для АЦП0 и АЦП1 в тактах частоты ACLK |
| 23..20 | ALT_DEL_CHN_P | Номер канала, при выборе которого будет использовано альтернативное время заряда емкости |
| 19 | - | Зарезервировано |
| 18..14 | MAXCNV_ADC0 | Количество преобразований, выполняемых за одну последовательность преобразований. Нулевое значение выполнит одно преобразование (отсчёт с нуля). |
| 13..6 | CONV_TRIG_CNTR_ADC0 | Выбор источников запуска преобразований для бита STARTofCONV_ADC0. Запись единицы в соответствующий бит разрешает соответствующий источник для запусков цепочек преобразования. Источники запуска преобразований описаны в разделе «Широтно-импульсный модулятор», а табл. 18.24.1.3 и 18.24.2.3 описывают распределение этих источников в зависимости от значения поля CONV_TRIG_CNTR_ADCx. Например, значение 0010_0001 для ADC00 устанавливает в качестсве источника одновременно EPWMxSOCA_1 и ADC_ESOC_0_0. |

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 5 | - | Зарезервировано |
| 4 | ETRNL_CONV_MODE_ADC0 | Управление режимом цепочки преобразования АЦП0:<br>0 – Режим однократного выполнения последовательности преобразований; После завершения цепочки преобразования бит STARTofCONV_ADC0 сбрасывается. Для следующего запуска последовательности перобразований необходимо установить бит STARTofCONV_ADC0.<br>1 – Режим непрерывных преобразований; После завершения цепочки преобразования бит STARTofCONV_ADC0 не сбрасываются и преобразования начинаются с начала |
| 3 | STARTofCONV_ADC0 | Бит начала преобразования АЦП0. Может быть установлен либо программно, либо автоматически. После окончания цепочки преобразования автоматически сбрасывается (кроме режима непрерывных преобразований):<br>0 – Чтение: нет преобразования;<br>Запись: нет эффекта;<br>1 – Чтение: преобразование в процессе;<br>Запись: начало преобразования |
| 2 | - | Зарезервировано |
| 1 | RST_ADC0 | Сброс контроллера АЦП0:<br>0 – Контроллер АЦП в рабочем режиме;<br>1 – Контроллер АЦП в сбросе |
| 0 | EN_ADC0 | Разрешение работы преобразователя и контроллера АЦП0<br>0 – Работа контроллера АЦП запрещена;<br>1 – Работа контроллера АЦП разрешена |

Таблица 377 – Распределение запросов преобразований для АЦП0

| Блок АЦП | Значение поля CONV_TRIG_CNTR_ADCx | Источники начала преобразования |
|---|---|---|
| ADC00 | 1000_0000 | EPWMxSOCA_2 |
| | 0100_0000 | EPWMxSOCB_2 |
| | 0010_0000 | EPWMxSOCA_1 |
| | 0001_0000 | EPWMxSOCB_1 |
| | 0000_1000 | EPWMxSOCA_0 |
| | 0000_0100 | EPWMxSOCB_0 |
| | 0000_0010 | EPWMxSOCA_7 |
| | 0000_0001 | ADC_ESOC_0_0 |
| ADC01 | 1000_0000 | EPWMxSOCA_5 |
| | 0100_0000 | EPWMxSOCA_4 |
| | 0010_0000 | EPWMxSOCA_3 |
| | 0001_0000 | EPWMxSOCA_2 |
| | 0000_1000 | EPWMxSOCA_1 |
| | 0000_0100 | EPWMxSOCA_0 |
| | 0000_0010 | EPWMxSOCA_8 |
| | 0000_0001 | ADC_ESOC_0_1 |
| ADC02 | 1000_0000 | EPWMxSOCA_8 |
| | 0100_0000 | EPWMxSOCA_7 |
| | 0010_0000 | EPWMxSOCA_6 |
| | 0001_0000 | EPWMxSOCB_8 |
| | 0000_1000 | EPWMxSOCB_7 |
| | 0000_0100 | EPWMxSOCB_6 |
| | 0000_0010 | EPWMxSOCA_1 |
| | 0000_0001 | ADC_ESOC_0_2 |

### 20.27.2 *ADC1_CNTR*

Таблица 378 – Регистр ADC1_CNTR

| **Номер** | 31..24 | 23..20 | 19 | 18..14 | 13..6 |
|---|---|---|---|---|---|
| **Доступ** | R/W | R/W | - | R/W | R/W |
| **Сброс** | 0 | 0 | - | 0 | 0 |
| **Имя** | DELAY_TIME_ALT | ALT_DEL_CHN_N | - | MAXCNV_ADC1 | CONV_TRIG_CNTR_ADC1 |

| **Номер** | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|
| **Доступ** | - | R/W | R/W | R/W | R/W | R/W |
| **Сброс** | - | 0 | 0 | 0 | 0 | 0 |
| **Имя** | - | ETRNL_CONV_MODE_ADC1 | STARTofCONV_ADC1 | ADC_BUFEN | RST_ADC1 | EN_ADC1 |

Таблица 379 – Описание бит регистра ADC1_CNTR

| **№ бита** | **Функциональное имя бита** | **Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений** |
|---|---|---|
| 31..24 | DELAY_TIME_ALT | Альтернативное время заряда емкости для АЦП0 и АЦП1 в тактах частоты ACLK |
| 23..20 | ALT_DEL_CHN_N | Номер канала, при выборе которого будет использовано альтернативное время заряда емкости |
| 19 | - | Зарезервировано |
| 18..14 | MAXCNV_ADC1* | |
| 13..6 | CONV_TRIG_CNTR_ADC1* | |
| 5 | - | Зарезервировано |
| 4 | ETRNL_CONV_MODE_ADC1* | |
| 3 | STARTofCONV_ADC1* | |
| 2 | ADC_BUFEN | Включение буфера АЦП (для 2х АЦП сразу)<br>0 – выключено<br>1 – включено |
| 1 | RST_ADC1* | |
| 0 | EN_ADC1* | |

* Описание бит совпадает с описанием бит регистра ADC0_CNTR при замене строк ADC0 на ADC1 и АЦП0 на АЦП1

Таблица 380 – Распределение запросов преобразований для АЦП1

| Блок АЦП | Значение поля CONV_TRIG_CNTR_ADCx | Источники начала преобразования |
|---|---|---|
| ADC10 | 1000_0000 | EPWMxSOCA_5 |
| | 0100_0000 | EPWMxSOCB_5 |
| | 0010_0000 | EPWMxSOCA_4 |
| | 0001_0000 | EPWMxSOCB_4 |
| | 0000_1000 | EPWMxSOCA_3 |
| | 0000_0100 | EPWMxSOCB_3 |
| | 0000_0010 | EPWMxSOCA_7 |
| | 0000_0001 | ADC_ESOC_1_0 |
| ADC11 | 1000_0000 | EPWMxSOCB_5, |
| | 0100_0000 | EPWMxSOCB_4, |
| | 0010_0000 | EPWMxSOCB_3, |
| | 0001_0000 | EPWMxSOCB_2 |
| | 0000_1000 | EPWMxSOCB_1 |
| | 0000_0100 | EPWMxSOCB_0 |
| | 0000_0010 | EPWMxSOCB_8 |
| | 0000_0001 | ADC_ESOC_1_1 |

| Блок АЦП | Значение поля CONV_TRIG_CNTR_ADCx | Источники начала преобразования |
|---|---|---|
| ADC12 | 1000_0000 | EPWMxSOCA_8 |
| | 0100_0000 | EPWMxSOCA_7 |
| | 0010_0000 | EPWMxSOCA_6 |
| | 0001_0000 | EPWMxSOCB_8 |
| | 0000_1000 | EPWMxSOCB_7 |
| | 0000_0100 | EPWMxSOCB_6 |
| | 0000_0010 | EPWMxSOCB_1 |
| | 0000_0001 | ADC_ESOC_1_2 |

### 20.27.3 *ADCx_SCOPE_CNTR*

Таблица 381 – Регистр ADCx_SCOPE_CNTR

| **Номер** | 31..29 | 28..24 | 23..12 | 11..0 |
|---|---|---|---|---|
| **Доступ** | R/W | - | - | R/W |
| **Сброс** | 0 | - | - | 0 |
| **Имя** | SCOPE_MODE_ADCx | - | UPPER_SCOPE_ADCx | LOWER_SCOPE_ADCx |

Таблица 382 – Описание бит регистра ADCx_SCOPE_CNTR

| **№ бита** | **Функциональное имя бита** | **Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений** |
|---|---|---|
| 31..29 | SCOPE_MODE_ADCx | Режим контроля границ АЦПх<br>000 – режим ограничения выдачи результатов выключен;<br>001 – не записывать результаты, попадающие вне заданных границ преобразований включительно;<br>010 – не записывать результаты, попадающие внутрь заданных границ преобразований включительно;<br>011 – не записывать результаты, попадающие вне заданных границ преобразований включительно, выдавать сигнал ошибки по данному событию;<br>100 – не записывать результаты, попадающие внутрь заданных границ преобразований включительно, выдавать сигнал ошибки по данному событию;<br>101 – записывать результаты, выдавать сигнал ошибки, если результат попадает внутрь заданных границ преобразований включительно;<br>110 – записывать результаты, выдавать сигнал ошибки, если результат попадает вне заданных границ преобразований включительно;<br>111 – записывать результаты, выдавать ошибку, если результат совпадает с границами преобразований |
| 28...24 | - | Зарезервировано |
| 23..12 | UPPER_SCOPE_ADCx | Верхняя граница преобразований АЦПх |
| 11..0 | LOWER_SCOPE_ADCx | Нижняя граница преобразований АЦПх |

### 20.27.4 *ADC_SYNC_CNTR*

Таблица 383 – Регистр ADC_SYNC_CNTR

| **Номер** | 31..28 | 27..20 | 19 | 18..16 |
|---|---|---|---|---|
| **Доступ** | - | R/W | - | R/W |
| **Сброс** | - | 0 | - | 0 |
| **Имя** | - | SYNCofCONV_out_DELAY_TIME_ADC1 | - | SYNCofCONV_in_MODE_ADC1 |

| **Номер** | 15..12 | 11..4 | 3 | 2..0 |
|---|---|---|---|---|
| **Доступ** | - | R/W | - | R/W |
| **Сброс** | - | 0 | - | 0 |
| **Имя** | - | SYNCofCONV_out_DELAY_TIME_ADC0 | - | SYNCofCONV_in_MODE_ADC0 |

Таблица 384 – Описание бит регистра ADC_SYNC_CNTR

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31..28 | - | Зарезервировано |
| 27...20 | SYNCofCONV_out_DELAY_TIME_ADC1 | Время выдачи сигнала синхронизации в тактах АЦП1 от начала преобразования.<br>0 – 0 -тактов;<br>1 – 1- такт;<br>2 - … |
| 19 | - | Зарезервировано |
| 18..16 | SYNCofCONV_in_MODE_ADC1 | Режим синхронизации АЦП1<br>000 – Синхронизация отключена;<br>001 – Синхронизация относительно сигнала АЦП0;<br>010 – Синхронизация относительно внешнего сигнала;<br>011 – Синхронный режим работы;<br>100 – Попеременный режим работы; |
| 15..12 | - | Зарезервировано |
| 11..4 | SYNCofCONV_out_DELAY_TIME_ADC0 | Время выдачи сигнала синхронизации в тактах АЦП0 от начала преобразования.<br>0 – Сигнал выдается на один такт через 0 –тактов;<br>1 – Сигнал выдается на один такт через 1- такт;<br>2 - … |
| 3 | - | Зарезервировано |
| 2..0 | SYNCofCONV_in_MODE_ADC0 | Режим синхронизации АЦП0<br>000 – Синхронизация отключена;<br>001 – Синхронизация относительно сигнала АЦП1;<br>010 – Синхронизация относительно внешнего сигнала;<br>011– Синхронный режим работы;<br>100 – Попеременный режим работы; |

### 20.27.5 *ADC_BUF_CNTR*

Таблица 385 – Регистр ADC_BUF_CNTR

| **Номер** | 31..19 | 20 | 19..17 | 16..12 |
|---|---|---|---|---|
| **Доступ** | - | R/W | - | R/W |
| **Сброс** | - | 0 | - | 0 |
| **Имя** | - | BUF_MODE_ADC1 | - | FIFO_LIM_ADC1 |

| **Номер** | 11..9 | 8 | 7..5 | 4..0 |
|---|---|---|---|---|
| **Доступ** | - | R/W | - | R/W |
| **Сброс** | - | 0 | - | 0 |
| **Имя** | - | BUF_MODE_ADC0 | - | FIFO_LIM_ADC0 |

Таблица 386 – Описание бит регистра ADC_BUF_CNTR

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31...19 | - | Зарезервировано |
| 20 | BUF_MODE_ADC1 | Режим работы буфера результатов АЦП1<br>0 – Независимые регистры<br>1 – FIFO |
| 19...17 | - | Зарезервировано |
| 16..12 | FIFO_LIM_ADC1 | Количество заполненных ячеек АЦП1, вызывающее прерывание |
| 11..9 | - | Зарезервировано |
| 8 | BUF_MODE_ADC0 | Режим работы буфера результатов АЦП0<br>0 – Независимые регистры<br>1 – FIFO |

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 7..5 | - | Зарезервировано |
| 4..0 | FIFO_LIM_ADC0 | Количество заполненных ячеек АЦП0, вызывающее прерывание |

### 20.27.6 *ADC_BUF_STATE*

Таблица 387 – Регистр ADC_BUF_STATE

| **Номер** | 31..18 | 17 | 16 | 15 | 14..9 |
|---|---|---|---|---|---|
| **Доступ** | - | R/W | R/W | R/W | R/W |
| **Сброс** | - | 0 | 0 | 0 | 0 |
| **Имя** | - | FIFO_FULL_ ADC1 | FIFOnEMPTY_ADC1 | FIFO_LIM_ADC1 | CELL_BUSY_ADC1 |

| **Номер** | 8 | 7 | 6 | 5..0 |
|---|---|---|---|---|
| **Доступ** | R/W | R/W | R/W | R/W |
| **Сброс** | 0 | 0 | 0 | 0 |
| **Имя** | FIFO_FULL_ADC0 | FIFOnEMPTY_ADC0 | FIFO_LIM_ADC0 | CELL_BUSY_ADC0 |

Таблица 388 – Описание бит регистра ADC_BUF_STATE

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31...18 | - | Зарезервировано |
| 17 | FIFO_FULL_ADC1 | FIFO АЦП1 полностью заполнено |
| 16 | FIFOnEMPTY_ADC1 | В FIFO АЦП1 есть данные |
| 15 | FIFO_LIM_ADC1 | Достигнут установленный лимит заполнения FIFO АЦП1 |
| 14..9 | CELL_BUSY_ADC1 | Количество занятых ячеек FIFO АЦП1 |
| 8 | FIFO_FULL_ADC0 | FIFO АЦП0 полностью заполнено |
| 7 | FIFOnEMPTY_ADC0 | В FIFO АЦП0 есть данные |
| 6 | FIFO_LIM_ADC0 | Достигнут установленный лимит заполнения FIFO АЦП0 |
| 5..0 | CELL_BUSY_ADC0 | Количество занятых ячеек FIFO АЦП0 |

### 20.27.7 *ADC_INT_CNTR*

Таблица 389 – Регистр ADC_INT_CNTR

| **Номер** | 31..8 | 7 | 6 | 5 | 4 |
|---|---|---|---|---|---|
| **Доступ** | - | R/W | R/W | R/W | R/W |
| **Сброс** | - | 0 | 0 | 0 | 0 |
| **Имя** | - | ENDofCONV_ ADC1 | FIFO_LIM_ADC1 _INT_EN | FIFOnEMPTY_ ADC1_INT_EN | FIFO_FULL_ADC1_INT_ EN |

| **Номер** | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| **Доступ** | R/W | R/W | R/W | R/W |
| **Сброс** | 0 | 0 | 0 | 0 |
| **Имя** | ENDofCONV_ ADC0 | FIFO_LIM_ADC0 _INT_EN | FIFOnEMPTY_ADC0_INT _EN | FIFO_FULL_ADC0_INT _EN |

Таблица 390 – Описание бит регистра ADC_INT_CNTR

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31..8 | - | Зарезервировано |
| 7 | ENDofCONV_ADC1 | Разрешение флага прерывания по завершению последовательности преобразования АЦП1* |
| 6 | FIFO_LIM_ADC1_INT_EN | Разрешение флага прерывания по достижению лимита данных в буфере АЦП1* |

| | | |
|---|---|---|
| 5 | FIFOnEMPTY_ADC1_INT_EN | Разрешение флага прерывания по наличию данных в буфере АЦП1* |
| 4 | FIFO_FULL_ADC1_INT_EN | Разрешение флага прерывания по переполнению буфера АЦП1* |
| 3 | ENDofCONV_ADC0 | Разрешение флага прерывания по завершению последовательности преобразования АЦП0* |
| 2 | FIFO_LIM_ADC0_INT_EN | Разрешение флага прерывания по достижению лимита данных в буфере АЦП0* |
| 1 | FIFOnEMPTY_ADC0_INT_EN | Разрешение флага прерывания по наличию данных в буфере АЦП0* |
| 0 | FIFO_FULL_ADC0_INT_EN | Разрешение флага прерывания по переполнению буфера АЦП0* |

* 0 – запрещено, 1 – разрешено

### 20.27.8 *ADC_DMA_CNTR*

Таблица 391 – Регистр ADC_ DMA_CNTR

| **Номер** | 31..8 | 7 | 6 |
|---|---|---|---|
| **Доступ** | - | R/W | R/W |
| **Сброс** | - | 0 | 0 |
| **Имя** | - | ENDofCONV_ADC1_DMA_EN | FIFO_LIM_ADC1_DMA_EN |

| **Номер** | 5 | 4 | 3 |
|---|---|---|---|
| **Доступ** | R/W | R/W | R/W |
| **Сброс** | 0 | 0 | 0 |
| **Имя** | FIFOnEMPTY_ADC1_DMA_EN | FIFO_FULL_ADC1_DMA_EN | ENDofCONV_DMA_EN_ADC0 |

| **Номер** | 2 | 1 | 0 |
|---|---|---|---|
| **Доступ** | R/W | R/W | R/W |
| **Сброс** | 0 | 0 | 0 |
| **Имя** | FIFO_LIM_ADC0_DMA_EN | FIFOnEMPTY_ADC0_DMA_EN | FIFO_FULL_ADC0_DMA_EN |

Таблица 392 – Описание бит регистра ADC_DMA_CNTR

| **№ бита** | **Функциональное имя бита** | **Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений** |
|---|---|---|
| 31...8 | - | Зарезервировано |
| 7 | ENDofCONV_ADC1_DMA_EN | Разрешение запроса DMA по завершению последовательности преобразования АЦП1* |
| 6 | FIFO_LIM_ADC1_DMA _EN | Разрешение запроса DMA по достижению лимита данных в буфере АЦП1* |
| 5 | FIFOnEMPTY_ADC1_ DMA _EN | Разрешение запроса DMA по наличию данных в буфере АЦП1* |
| 4 | FIFO_FULL_ADC1_ DMA _EN | Разрешение запроса DMA по переполнению буфера АЦП1* |
| 3 | ENDofCONV_ DMA _EN_ADC0 | Разрешение запроса DMA по завершению последовательности преобразования АЦП0* |
| 2 | FIFO_LIM_ADC0_ DMA _EN | Разрешение запроса DMA по достижению лимита данных в буфере АЦП0* |
| 1 | FIFOnEMPTY_ADC0_ DMA _EN | Разрешение запроса DMA по наличию данных в буфере АЦП0* |
| 0 | FIFO_FULL_ADC0_ DMA _EN | Разрешение запроса DMA по переполнению буфера АЦП0* |

* 0 – запрещено, 1 – разрешено

### 20.27.9 *ADC_ANALOG_CNTR*

Таблица 393 – Регистр ADC_ ANALOG_CNTR

| **Номер** | 31..23 | 22 | 21 | 20 | 19..12 |
|---|---|---|---|---|---|
| **Доступ** | - | R/W | R/W | R/W | R/W |
| **Сброс** | - | 0 | 0 | 0 | 0 |
| **Имя** | - | ADC1_CALDONE | MODE_ADC1 | OFFSET_MODE_ADC1 | OFFSET _ADC1 |

| **Номер** | 11 | 10 | 9 | 8 | 7..0 |
|---|---|---|---|---|---|
| **Доступ** | - | R/W | R/W | R/W | R/W |
| **Сброс** | - | 0 | 0 | 0 | 0 |
| **Имя** | - | OFFSET _ADC1 | MODE_ADC0 | OFFSET_MODE_ADC0 | OFFSET _ADC0 |

Таблица 394 – Описание бит регистра ADC_ ANALOG_CNTR

| № бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31..23 | - | Зарезервировано |
| 22 | ADC1_CALDONE | Флаг завершения калибровки АЦП1<br>0 – не завершена<br>1 – завершена |
| 21 | MODE_ADC1 | Режим работы АЦП<br>0 – Однополярный<br>1 – Дифференциальный |
| 20 | OFFSET_MODE_ADC1 | Режим работы АЦП<br>0 – Ручной<br>1 – Автоматический |
| 19..12 | OFFSET _ADC1 | Величина смещения АЦП1 |
| 11 | - | Зарезервировано |
| 10 | ADC0_CALDONE | Флаг завершения калибровки АЦП0<br>0 – не завершена<br>1 – завершена |
| 9 | MODE_ADC0 | Режим работы АЦП<br>0 – Однополярный<br>1 – Дифференциальный |
| 8 | OFFSET_MODE_ADC0 | Режим работы АЦП<br>0 – Ручной<br>1 – Автоматический |
| 7..0 | OFFSET _ADC0 | Величина смещения АЦП0 |

### 20.27.10 *ADC_BG_CNTR*

Таблица 395 – Регистр ADC_BG_CNTR

| **Номер** | 31..22 | 21 | 20 | 19 | 18 |
|---|---|---|---|---|---|
| **Доступ** | - | R/W | R/W | R/W | R/W |
| **Сброс** | - | 0 | 0 | 0 | 0 |
| **Имя** | - | ADC_BG_ZMODE | ADC_BG_VRECMODE | ADC_BG_IRECMODE | ADC_BG_EXTMODE |

| **Номер** | 17..12 | 11..8 | 7 | 6..1 | 0 |
|---|---|---|---|---|---|
| **Доступ** | R/W | R/W | R/W | R/W | R/W |
| **Сброс** | 0 | 0 | 0 | 0 | 0 |
| **Имя** | ADC_BG_SWMODE[5:0] | ADC_BG_BFEXT[5:2] | ADC_BG_BGEN | ADC_BG_BFEN[5:0] | ADC_BG_IREFEN |

Таблица 396 – Описание бит регистра ADC_BG_CNTR

| Биты | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31..22 | - | Зарезервировано |
| 21 | ADC_BG_ZMODE | Отключение выхода BG от выходных буферов:<br>0 – Выход BG подключен (нормальный режим работы),<br>1 – Выход BG отключен (Z состояние). |
| 20 | ADC_BG_VRECMODE | Включение аварийного источника напряжения:<br>0 – нормальная работа (нормальный режим работы),<br>1 – аварийная работа. |
| 19 | ADC_BG_IRECMODE | Включение аварийного источника тока:<br>0 – нормальная работа (нормальный режим работы),<br>1 – аварийная работа. |
| 18 | ADC_BG_EXTMODE | Переключение ядра блока на внешнюю опору 1,2В:<br>0 – выход генерируется ядром (нормальный режим работы),<br>1 – выход берется с внешней опоры 1,2В. |

| | | |
|---|---|---|
| 17..12 | ADC_BG_SWMODE[5:0]* | Управление режимом выходного напряжения одного из шести ([5:0]) буферов:<br>0 – 1,25 В,<br>1 – 2,5 В. |
| 11..8 | ADC_BG_BFEXT[5:2]* | Подключение одного из четырех ([5:2]) внешних буферов:<br>0 – внешний буфер отключен (нормальный режим работы),<br>1 – внешний буфер подключен. |
| 7 | ADC_BG_BGEN | Включение блока BG:<br>0 – выключен,<br>1 – включен. |
| 6..1 | ADC_BG_BFEN[5:0] | Включение одного из шести ([5:0]) буферов:<br>0 – выключен,<br>1 – включен. |
| 0 | ADC_BG_IREFEN | Сигнал разрешения работы встроенного источника опорного тока:<br>0 – выключен,<br>1 – включен. |

* См. раздел «Блоки формирования опорных напряжений» и «Таблица выводов микросхемы».

### 20.27.11 *ADCx_CHSEL*

Таблица 397 – Регистр ADCx_CHSEL

| **Номер** | 255..252 | 251..248 | ... | 15..8 | 7..0 |
|---|---|---|---|---|---|
| **Доступ** | R/W | R/W | R/W | R/W | R/W |
| **Сброс** | 0 | 0 | 0 | 0 | 0 |
| **Имя** | CONV31 | CONV30 | ... | CONV1 | CONV0 |

Таблица 398 – Описание бит регистра ADCx_CHSEL

| **№ бита** | **Функциональное имя бита** | **Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений** |
|---|---|---|
| 255..252 | CONV31 | Выбор канала для 31-го преобразования:<br>старшие четыре бита отвечают за n-канал,<br>младшие четыре бита отвечают за p-канал. |
| 251..248 | CONV30 | -//- |
| … | … | … |
| 15..8 | CONV01 | -//- |
| 7..0 | CONV00 | Выбор канала для 0-го преобразования:<br>старшие четыре бита отвечают за n-канал,<br>младшие четыре бита отвечают за p-канал. |

### 20.27.12 *ADC0_RESULT_i*

Таблица 399 – Регистр ADC0_RESULT_i

| **Номер** | 31..29 | 28..24 | 23..16 | 15..12 | 11..0 |
|---|---|---|---|---|---|
| **Доступ** | - | R/W | R/W | - | R/W |
| **Сброс** | - | 0 | 0 | - | 0 |
| **Имя** | - | CONV_NUMBER | CONV_TIME | - | RESULT_ADC0[11:0] |

Таблица 400 – Описание бит регистра ADC0_RESULT_i

| **№ бита** | **Функциональное имя бита** | **Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений** |
|---|---|---|
| 31..29 | - | Зарезервировано |
| 28..24 | CONV_NUMBER | Номер преобразования |
| 23...16 | CONV_TIME | Время преобразования в тактах рабочей частоты АЦП |
| 15..12 | - | Зарезервировано |
| 11..0 | RESULT_ADC0[11:0] | Результат преобразований |

### 20.27.13 *ADC1_RESULT_i*

Таблица 401 – Регистр ADC1_RESULT_i

| **Номер** | 31..29 | 28..24 | 23..16 | 15..12 | 11..0 |
|---|---|---|---|---|---|
| **Доступ** | - | R/W | R/W | - | R/W |
| **Сброс** | - | 0 | 0 | - | 0 |
| **Имя** | - | CONV_NUMBER | CONV_TIME | - | RESULT_ADC1[11:0] |

Таблица 402 – Описание бит регистра ADC1_RESULT_i

| **№ бита** | **Функциональное имя бита** | **Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений** |
|---|---|---|
| 31..29 | - | Зарезервировано |
| 28..24 | CONV_NUMBER | Номер преобразования |
| 23...16 | CONV_TIME | Время преобразования в тактах рабочей частоты АЦП |
| 15..12 | - | Зарезервировано |
| 11..0 | RESULT_ADC1[11:0] | Результат преобразований |

## 20.28 Описание регистров контроллеров ШИМ EPWM_CNTR

Таблица 403 – Регистры контроллеров ШИМ

| Базовый Адрес | Название | Описание |
|---|---|---|
| 0x4009_E000<br>0x4009_F000<br>0_400A_0000<br>0_400A_1000<br>0_400A_2000<br>0_400A_3000<br>0_400A_4000<br>0_400A_5000<br>0_400A_6000 | ePWM | Контроллер ePWM0<br>Контроллер ePWM1<br>Контроллер ePWM2<br>Контроллер ePWM3<br>Контроллер ePWM4<br>Контроллер ePWM5<br>Контроллер ePWM6<br>Контроллер ePWM7<br>Контроллер ePWM8 |
| **Смещение** | | |
| **Регистры блока счетчика таймера (Time-Base Submodule Registers)** | | |
| 0x00 | **TBPRD** | Регистр периода счета основного счетчика таймера (Time-Base Period Register) |
| 0x08 | **TBPHS** | Регистр значения фазового сдвига (Time-Base Phase Register) |
| 0x0C | **TBCTR** | Регистр основного счетчика ШИМ (Time-Base Counter Register) |
| 0x10 | **TBCTL** | Регистр управления (Time-Base Control Register) |
| 0x14 | **TBSTS** | Регистр статуса (Time-Base Status Register) |
| **Регистры блока сравнения (Counter-Compare Submodule Registers)** | | |
| 0x18 | **CMPA** | Регистр сравнения A (Counter-Compare A Register) |
| 0x1C | **CMPB** | Регистр сравнения B (Counter-Compare B Register) |
| 0x20 | **CMPCTL** | Регистр управления блока сравнения (Counter-Compare Control Register) |
| **Регистры блока анализа событий (Action-Qualifier Submodule Registers)** | | |
| 0x28 | **AQCTLA** | Action-Qualifier Output A Control Register |
| 0x2C | **AQCTLB** | Action-Qualifier Output B Control Register |
| 0x30 | **AQSFRC** | Action-Qualifier Software Force Register |
| 0x34 | **AQCSFRC** | Action-Qualifier Continuous Software Force Register |
| **Регистры блока управления мертвой зоной (Dead-Band Submodule Registers)** | | |
| 0x38 | **DBCTL** | Dead-Band Generator Control Register |
| 0x3C | **DBRED** | Dead-Band Generator Rising Edge Delay Register |
| 0x40 | **DBFED** | Dead-Band Generator Falling Edge Delay Register |
| PWM-Chopper Submodule Control Register | | |
| 0x44 | **PCCTL** | Регистр управления (PWM-Chopper Control Register) |
| **Регистры блока защиты (Trip-Zone Submodule Control and Status Registers)** | | |
| 0x48 | **TZSEL** | Регистр выбора (Trip-Zone Select Register) |
| 0x4C | **TZCTL** | Регистр управления (Trip-Zone Control Register) |
| 0x50 | **TZEINT** | Регистр разрешения прерывания (Trip-Zone Enable Interrupt Register) |
| 0x54 | **TZFLG** | Регистр флагов (Trip-Zone Flag Register) |
| 0x58 | **TZCLR** | Регистр очистки (Trip-Zone Clear Register) |
| 0x5C | **TZFRC** | Регистр установки (Trip-Zone Force Register) |
| **Регистры блока генерации события (Event-Trigger Submodule Registers)** | | |
| 0x60 | **ETSEL** | Регистр выбора события (Event-Trigger Selection Register) |
| 0x64 | **ETPS** | Ресгитр предделителя (Event-Trigger Prescale Register) |
| 0x68 | **ETFLG** | Регистр флага события (Event-Trigger Flag Register) |
| 0x6C | **ETCLR** | Event-Trigger Clear Register |
| 0x78 | **ETFRC** | Event-Trigger Force Register |

## 20.28.1 *Регистры блока счетчика таймера*

### 20.28.1.1 TBPRD

Значение после сброса: 0x0.

Таблица 404 – Регистр TBPRD

| **Номер** | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Доступ** | 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 | R/W |
| **Сброс** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | **TBPRD** | | | | | | | | | | | | | | | |

| **Номер** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Доступ** | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | **TBPRD** | | | | | | | | | | | | | | | |

Таблица 405 – Описание бит регистра TBPRD

| **Номер бита** | **Функциональное имя бита** | **Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений** |
|---|---|---|
| [31:0] | **TBPRD** | Период счета основного счетчика таймера.<br>Определяет частоту выхода ШИМа.<br>По данному адресу распложены регистр периода счета и его дублер. Запись в основной регистр или в регистр дублер определяется битом PRDLD регистра TBCTL.<br>Если запись ведется в регистр дублер, он будет перезаписан в основной регистр только по окончании предыдущего периода ШИМ.<br>Если запись ведется в основной регистр, значение регистра изменит период ШИМ сразу же после записи (в текущем периоде) |

### 20.28.1.2 TBPHS

Значение после сброса: 0x0.

Таблица 406 – Регистр TBPHS

| Номер | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | 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 | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | TBPHS | | | | | | | | | | | | | | | |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TBPHS | | | | | | | | | | | | | | | |

Таблица 407 – Описание бит регистра TBPHS

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:0] | **TBPHS** | Значение фазового сдвига, загружаемое в регистр основного счетчика таймера по событию синхронизации. Разрешение загрузки основного счетчика значением регистра TBPHS определяется битом PHSEN регистра TBCTL. Не рекомендуется устанавливать значение регистра TBPHS большее, чем регистра TBPRD. Данная комбинация может привести к остановке счетчика при возникновении события внешней синхронизации, в случае, если его обработка разрешена |

### 20.28.1.3 TBCTR

Значение после сброса: 0x0.

Таблица 408 – Регистр TBCTR

| Номер | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | 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 | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | TBCTR | | | | | | | | | | | | | | | |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | TBCTR | | | | | | | | | | | | | | | |

Таблица 409 – Описание бит регистра TBCTR

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:0] | **TBCTR** | Основной счетчик ШИМ |

### 20.28.1.4 TBCTL

Значение после сброса: 0x3.

Таблица 410 – Регистр TBCTL

| Номер | 31…22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | Зарезервировано | Зарезервировано | Зарезервировано | Зарезервировано | PHSDIR_EN | SWFSYNC_MODEO | SWFSYNC_MODEI |
|---|---|---|---|---|---|---|---|

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 1 | 1 |
| | Зарезервировано | PHSDIR | CLKDIV | | | | HSPCLKDIV | | | SWFSYNC | SYNCOSEL | | PRDLD | PHSEN | CTRMODE | |

Таблица 411 – Описание бит регистра TBCTL

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:22] | SOC_DELAY | Зарезервировано |
| 21 | PRD_LOAD_MODE | Выбор события загрузки значения периода:<br>Равенство счетчика периоду;<br>Равенство счетчика нулю.<br>В текущей версии микросхемы данный функционал не реализован. Биты зарезервированы. |
| 20 | SYNCI_TCLK_SEL | Выбор источника входного импульса синхронизации:<br>Пересинхронизация на PCLK;<br>Выдача оригинального события.<br>В текущей версии микросхемы данный функционал не реализован. Биты зарезервированы. |
| 19 | SYNCO_TCLK_SEL | Выбор источника выходного импульса синхронизации:<br>Пересинхронизация на PCLK;<br>Выдача оригинального события.<br>В текущей версии микросхемы данный функционал не реализован. Биты зарезервированы. |
| 18 | PHSDIR_EN | Разрешение смены направления счета при получении импульса внешней синхронизации:<br>0 – Запрещено;<br>1 – Разрешено |
| 17 | SWSYNC_MODEO | Режим работы выходного импульса синхронизации:<br>0 – Программный импульс синхронизации объединяется по «ИЛИ» с выходным импульсом синхронизации EPWMxSYNCO;<br>1 – Программный импульс синхронизации не объединяется по «ИЛИ» с выходным импульсом синхронизации EPWMxSYNCO |
| 16 | SWFSYNC_MODEI | Режим работы выходного импульса синхронизации:<br>0 – Программный импульс синхронизации объединяется по «ИЛИ» с входным импульсом синхронизации EPWMxSYNCI;<br>1 – Программный импульс синхронизации не объединяется по «ИЛИ» с входным импульсом синхронизации EPWMxSYNCI |

| | | |
|---|---|---|
| 15..14 | FREE, SOFT | Работа в режиме отладки:<br>00 Останавливаться после очередного декремента или инкремента основного счетчика.<br>01 Останавливаться после того, как счетчик пройдет полный цикл:<br>• Счет вверх: TBCTR = TBPRD;<br>• Счет вниз: TBCTR = 0x0000;<br>• Счет вверх-вниз: TBCTR = 0x0000;<br>1X Без остановок.<br>В текущей версии микросхемы данный функционал не реализован. Биты зарезервированы. |
| 13 | PHSDIR | Бит направления счета в каскадном режиме работы.<br>Используется только при включении основного счетчика в режим счета вверх-вниз.<br>Определяет направление счета основного счетчика после возникновения события синхронизации и загрузки нового значения фазового сдвига.<br>В режимах счета вверх или вниз данный бит игнорируется.<br>0 - Счет вниз после возникновения события синхронизации.<br>1 - Счет вверх после возникновения события синхронизации |
| [12:10] | CLDIV | Основная часть делителя тактовой частоты счетчика.<br>TBCLK = SYSCLKOUT / (HSPCLKDIV × CLKDIV)<br>000 - /1 (по умолчанию)<br>001 - /2<br>010 - /4<br>011 - /8<br>100 - /16<br>101 - /32<br>110 - /64<br>111 - /128 |
| 9..7 | HSPCLKDIV | Дополнительная часть делитель тактовой частоты счетчика.<br>TBCLK = SYSCLKOUT / (HSPCLKDIV × CLKDIV)<br>000 - /1<br>001 - /2 (по умолчанию)<br>010 - /4<br>011 - /6<br>100 - /8<br>101 - /10<br>110 - /12<br>111 - /14 |
| 6 | SWFSYNC | Бит программной установки импульса синхронизации EPWMxSYNC.<br>0 – нет импульса синхронизации.<br>1 – генерирует однократный импульс синхронизации.<br>Событие объединено по ИЛИ со входным импульсом синхронизации EPWMxSYNCI или с выходным EPWMxSYNCO, в зависимости от значения бита SWFSYNC_MODE регистра TBCTL |
| [5:4] | SYNCOSEL | Выбор источника выхода синхронизации EPWMxSYNCO.<br>00 Входной импульс синхронизации EPWMxSYNCI проходит напрямую на выходной импульс синхронизации EPWMxSYNCO (в т.ч. и программный импульс, если разрешено его объединение с входным импульсом по «ИЛИ» битом SWFSYNC_MODEI регистра TBCTL);<br>01 - CTR = zero: Основной счетчик равен 0 (TBCTR = 0x0000);<br>10 - CTR = CMPB: Основной счетчик равен значению сравнения счета (TBCTR = CMPB);<br>11 - Сигнал синхронизации выключен |

| | | |
|---|---|---|
| 3 | PRDLD | Выбор основного или дублирующего регистра периода ШИМ для записи.<br>0 - Регистр периода ШИМ загружается из дублирующего регистра, когда счетчик равен 0.<br>1 - Запись ведется в регистр периода ШИМ. Модуль счета изменяется немедленно после записи |
| 2 | PHSEN | Разрешение загрузки основного регистра счетчика из регистра фазы смещения.<br>0 - Основной регистр счета не загружается значением регистра фазы смещения.<br>1 - Загрузка основного регистра значением регистра фазы при возникновении события синхронизации программного события по установке бита SWFSYNC. |
| [1:0] | CTRMODE | Режим счета.<br>Изменение регистра активизируется на следующем, после его изменения, такте синхросигнала TBCLK<br>Значение поля:<br>00 - Счет вверх;<br>01 - Счет вниз;<br>10 - Счет вверх-вниз;<br>11 - Счет остановлен;<br>В режиме счёт ВВЕРХ-ВНИЗ недостижимы такие события, как:<br>-равенство нулю при счёте вверх (при достижении нуля происходит одновременная инкрементация счётчика и смена направления счёта на ВВЕРХ)<br>-равенство периоду при счёте вниз (при достижении периода происходит одновременная декрементация счётчика и смена направления счёта на ВНИЗ)<br><br>То есть в нормальном режиме работы (без программной записи значения счётчика) равенство нулю достижимо только при счёте ВНИЗ, а равенство периоду - только при счёте ВВЕРХ |

### 20.28.1.5 TBSTS

Значение после сброса: 0x1.

Таблица 412 – Регистр TBSTS

| Номер | 31…16 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | Зарезервировано |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 1 |
| | Зарезервировано | | | | | | | | | | | | | CTRMAX | SYNCI | CTRDIR |

Таблица 413 – Описание бит регистра TBSTS

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:3] | Зарезервировано | Зарезервировано |

| | | |
|---|---|---|
| 2 | CTRMAX | Основной таймер достиг максимального значения.<br>Чтение 0 – максимальное значение не достигнуто.<br>Запись 0 – не влияет на значение данного бита.<br>Чтение 1 – достигнуто максимальное значение счета.<br>Запись 1 – сбрасывает бит |
| 1 | SYNCI | Флаг внешнего события синхронизация.<br>Чтение 0 – внешнего события синхронизации не было.<br>Запись 0 – не влияет на значение данного бита.<br>Чтение 1 – возникновение внешнего события синхронизации.<br>Запись 1 – сбрасывает бит в ноль |
| 0 | CTRDIR | Направление счета основного счетчика.<br>Чтение 0 – Вниз.<br>Чтение 1 – Вверх |

### 20.28.2 *Регистры блока сравнения*

#### 20.28.2.1 CMPA

Значение после сброса: 0x0.

Таблица 414 – Регистр CMPA

| Номер | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | 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 | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | CMPA | | | | | | | | | | | | | | | |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | CMPA | | | | | | | | | | | | | | | |

Таблица 415 – Описание бит регистра CMPA

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:0] | CMPA | Значение регистра CMPA сравнивается со значением основного регистра счета (TBCTR). Когда значения равны, генерируется событие равенства CMPA. Это событие поступает на блок выбора действия, где выбирается возможное действие на выводах EPWMxA или EPWMxB, в зависимости от регистров AQCTLA и AQCTLB. Возможные действия, выбираемые на основе значения регистров AQCTLA и AQCTLB, включают:<br>• Не делать ничего. Событие игнорируется;<br>• Сброс: Переключение EPWMxA и/или EPWMxB в состояние «0»;<br>• Установка: Переключение EPWMxA и/или EPWMxB в состояние «1»;<br>• Переключение выводов EPWMxA и/или EPWMxB в противоположное состояние;<br>Дублирование этого регистра может осуществляться битом SHDWAMODE регистра CMPCTL. По умолчанию запись ведется в дублирующий буфер. |

| | | |
|---|---|---|
| | | • Если бит SHDWAMODE = 0, запись и чтение ведется с регистром дублирования. В этом случае, битом LOADAMODE регистра CMPCTL будет определяться событие, по которому будет производиться обновление основного регистра из дублирующего регистра.<br>• Если бит SHDWAMODE = 1, запись и чтение ведется непосредственно из основного регистра. Любые изменения будут иметь эффект немедленно.<br>Основной и дублирующие регистры имеют один и тот же адрес доступа |

### 20.28.2.2 CMPB

Значение после сброса: 0x0.

Таблица 416 – Регистр CMPB

| Номер | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | 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 | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | CMPB | | | | | | | | | | | | | | | |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | CMPB | | | | | | | | | | | | | | | |

Таблица 417 – Описание бит регистра CMPB

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:0] | CMPB | Значение регистра CMPB сравнивается со значением основного регистра счета (TBCTR). Когда значения равны, генерируется события достижения CMPA. Это событие поступает на блок выбора действия, где выбирается возможное действие на выводах EPWMxA или EPWMxB, в зависимости от регистров AQCTLA и AQCTLB. Возможные действия, выбираемые на основе значения регистров AQCTLA и AQCTLB, включают:<br>• Не делать ничего. Событие игнорируется;<br>• Сброс: Переключение EPWMxA и/или EPWMxB в состояние «0»;<br>• Установка: Переключение EPWMxA и/или EPWMxB в состояние «1»;<br>• Переключение выводов EPWMxA и/или EPWMxB в противоположное состояние;<br>Дублирование этого регистра может осуществляться битом SHDWAMODE регистра CMPCTL. По умолчанию запись ведется в дублирующий буфер.<br>• Если бит SHDWAMODE = 0, запись и чтение ведется с регистром дублирования. В этом случае, битом LOADAMODE регистра CMPCTL будет определяться событие, по которому будет производиться обновление основного регистра из дублирующего регистра.<br>• Если бит SHDWAMODE = 1, запись и чтение ведется непосредственно из основного регистра. Любые изменения будут иметь эффект немедленно.<br>Основной и дублирующие регистры имеют один и тот же адрес доступа. |

### 20.28.2.3 CMPCTL

Значение после сброса: 0xc00.

Таблица 418 – Регистр CMPCTL

| Номер | 31…16 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | Зарезервировано |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Сброс | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | Зарезервировано | | | | SHDWBEMPTY | SHDWAEMPTY | SHDWBFULL | SHDWAFULL | Зарезервировано | SHDWBMODE | Зарезервировано | SHDWAMODE | LOADBMODE | | LOADMODE | |

Таблица 419 – Описание бит регистра CMPCTL

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:12] | Зарезервировано | Зарезервировано |
| 11 | SHDWBEMPTY | Флаг FIFO дублирующих регистров сравнения B (CMPB) пусто. Бит очищается автоматически после загрузки очередного значения из FIFO в регистр CMPB.<br>0 – FIFO не пусто<br>1 – FIFO пусто |
| 10 | SHDWAEMPTY | Флаг FIFO дублирующих регистров сравнения A (CMPA) пусто. Бит очищается автоматически после загрузки очередного значения из FIFO в регистр CMPA.<br>0 – FIFO не пусто<br>1 – FIFO пусто |
| 9 | SHDWBFULL | Флаг FIFO дублирующих регистров сравнения B (CMPB) заполнено.<br>Бит очищается автоматически после загрузки очередного значения из FIFO в регистр CMPB.<br>0 – FIFO не заполнено.<br>1 – FIFO заполнено |
| 8 | SHDWAFULL | Флаг FIFO дублирующих регистров сравнения A (CMPA) заполнено.<br>Бит очищается автоматически после загрузки очередного значения из FIFO в регистр CMPA.<br>0 – FIFO не заполнено.<br>1 – FIFO заполнено |
| 7 | Зарезервировано | Зарезервировано |
| 6 | SHDWBMODE | Режим записи регистра CMPB.<br>0 - Режим дублирования. Запись ведется в дублирующее FIFO.<br>1 - Режим записи в основной регистр |
| 5 | Зарезервировано | Зарезервировано |
| 4 | SHDWAMODE | Режим записи регистра CMPA.<br>0 - Режим дублирования. Запись ведется в дублирующее FIFO.<br>1 - Режим записи в основной регистр |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 3..2 | LOADBMODE | Режим загрузки значений из дублирующего буфера регистра CMPB в основной регистр. Поле игнорируется в режиме записи в основной регистр.<br>00 – Загрузка по событию CTR = Zero: (TBCTR = 0x0000);<br>01 – Загрузка по событию CTR = PRD: (TBCTR = TBPRD);<br>10 – Загрузка по событиям CTR = Zero и CTR = PRD;<br>11 – Значения из буфера не загружаются в основной регистр |
| 1..0 | LOADAMODE | Режим загрузки значений из дублирующего буфера регистра CMPA в основной регистр. Поле игнорируется в режиме записи в основной регистр.<br>00 – Загрузка по событию CTR = Zero: (TBCTR = 0x0000);<br>01 – Загрузка по событию CTR = PRD: (TBCTR = TBPRD);<br>10 – Загрузка по событиям CTR = Zero и CTR = PRD;<br>11 – Значения из буфера не загружаются в основной регистр |

### 20.28.3 *Регистры блока анализа событий*

#### 20.28.3.1 AQCTLA

Значение после сброса: 0x0.

Таблица 420 – Регистр AQCTLA

| Номер | 31…24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|
| 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 |
| | Зарезервировано | EVNT1_ B_SEL | | EVNT2_ B_SEL | | EVNT3_ B_SEL | | EVNT4_ B_SEL | |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | EVNT1U_ B_ MODE | | EVNT1U_ B_ MODE | | EVNT2U_ B_ MODE | | EVNT2D_ B_ MODE | | EVNT3U_ B_ MODE | | EVNT3D_ B_ MODE | | EVNT4U_ B_ MODE | | EVNT4D_ B_MODE | |

Таблица 421 – Описание бит регистра AQCTLA

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:24] | Зарезервировано | Зарезервировано |
| [23:22] | EVNT1_ A_SEL | Поле выбора источника для события 1.<br>00 – равенства таймера значению CMPB (CTReqCMPB);<br>01 – равенства таймера значению CMPA (CTReqCMPA);<br>10 – равенства таймера нулю (CTReqZERO);<br>11 – равенства таймера периоду счета (CTReqPRD) |
| [21:20] | EVNT2_ A_SEL | Поле выбора источника для события 2.<br>00 – равенства таймера значению CMPB (CTReqCMPB);<br>01 – равенства таймера значению CMPA (CTReqCMPA);<br>10 – равенства таймера нулю (CTReqZERO);<br>11 – равенства таймера периоду счета (CTReqPRD) |
| [19:18] | EVNT3_ A_SEL | Поле выбора источника для события 3.<br>00 – равенства таймера значению CMPB (CTReqCMPB);<br>01 – равенства таймера значению CMPA (CTReqCMPA);<br>10 – равенства таймера нулю (CTReqZERO);<br>11 – равенства таймера периоду счета (CTReqPRD) |
| [17:16] | EVNT4_ A_SEL | Поле выбора источника для события 4.<br>00 – равенства таймера значению CMPB (CTReqCMPB);<br>01 – равенства таймера значению CMPA (CTReqCMPA);<br>10 – равенства таймера нулю (CTReqZERO);<br>11 – равенства таймера периоду счета (CTReqPRD) |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [15:14] | EVNT1U_A_MODE | Возможные действия, по событию 1, в случае, если счетчик инкрементируется:<br>00 - Не делать ничего. Событие игнорируется;<br>01 - Сброс: Переключение EPWMxA в состояние «0»;<br>10 - Установка: Переключение EPWMxA в состояние «1»;<br>11 - Переключение выводов EPWMxA в состояние, определяемое полем I_MODEA регистра AQSFRC |
| [13:12] | EVNT1D_A_MODE | Возможные действия, по событию 1, в случае, если счетчик декрементируется:<br>00 - Не делать ничего. Событие игнорируется;<br>01 - Сброс: Переключение EPWMxA в состояние «0»;<br>10 - Установка: Переключение EPWMxA в состояние «1»;<br>11 - Переключение выводов EPWMxA в состояние, определяемое полем I_MODEA регистра AQSFRC |
| [11:10] | EVNT2U_A_MODE | Возможные действия, по событию 2, в случае, если счетчик инкрементируется:<br>00 - Не делать ничего. Событие игнорируется;<br>01 - Сброс: Переключение EPWMxA в состояние «0»;<br>10 - Установка: Переключение EPWMxA в состояние «1»;<br>11 - Переключение выводов EPWMxA в состояние, определяемое полем I_MODEA регистра AQSFRC |
| [9:8] | EVNT2D_A_MODE | Возможные действия, по событию 2, в случае, если счетчик декрементируется:<br>00 - Не делать ничего. Событие игнорируется;<br>01 - Сброс: Переключение EPWMxA в состояние «0»;<br>10 - Установка: Переключение EPWMxA в состояние «1»;<br>11 - Переключение выводов EPWMxA в состояние, определяемое полем I_MODEA регистра AQSFRC. |
| [7:6] | EVNT3U_A_MODE | Возможные действия, по событию 3, в случае, если счетчик инкрементируется:<br>00 - Не делать ничего. Событие игнорируется;<br>01 - Сброс: Переключение EPWMxA в состояние «0»;<br>10 - Установка: Переключение EPWMxA в состояние «1»;<br>11 - Переключение выводов EPWMxA в состояние, определяемое полем I_MODEA регистра AQSFRC |
| [5:4] | EVNT3D_A_MODE | Возможные действия, по событию 3, в случае, если счетчик декрементируется:<br>00 - Не делать ничего. Событие игнорируется;<br>01 - Сброс: Переключение EPWMxA в состояние «0»;<br>10 - Установка: Переключение EPWMxA в состояние «1»;<br>11 - Переключение выводов EPWMxA в состояние, определяемое полем I_MODEA регистра AQSFRC |
| [3:2] | EVNT4U_A_MODE | Возможные действия, по событию 4, в случае, если счетчик инкрементируется:<br>00 - Не делать ничего. Событие игнорируется;<br>01 - Сброс: Переключение EPWMxA в состояние «0»;<br>10 - Установка: Переключение EPWMxA в состояние «1»;<br>11 - Переключение выводов EPWMxA в состояние, определяемое полем I_MODEA регистра AQSFRC |
| [1:0] | EVNT4D_A_MODE | Возможные действия, по событию 4, в случае, если счетчик декрементируется:<br>00 - Не делать ничего. Событие игнорируется;<br>01 - Сброс: Переключение EPWMxA в состояние «0»;<br>10 - Установка: Переключение EPWMxA в состояние «1»;<br>11 - Переключение выводов EPWMxA в состояние, определяемое полем I_MODEA регистра AQSFRC |

### 20.28.3.2 AQCTLB

Значение после сброса: 0x0.

Таблица 422 – Регистр AQCTLB

| Номер | 31…24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|
| 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 |
| | Зарезервировано | EVNT1_ B_SEL | | EVNT2_ B_SEL | | EVNT3_ B_SEL | | EVNT4_ B_SEL | |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | EVNT1U_ B_ MODE | | EVNT1U_ B_ MODE | | EVNT2U_ B_ MODE | | EVNT2D_ B_ MODE | | EVNT3U_ B_ MODE | | EVNT3D_ B_ MODE | | EVNT4U_ B_ MODE | | EVNT4D_ B_MODE | |

Таблица 423 – Описание бит регистра AQCTLB

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:24] | Зарезервировано | Зарезервировано |
| [23:22] | EVNT1_ B_SEL | Поле выбора источника для события 1.<br>00 – равенства таймера значению CMPB (CTReqCMPB);<br>01 – равенства таймера значению CMPA (CTReqCMPA);<br>10 – равенства таймера нулю (CTReqZERO);<br>11 – равенства таймера периоду счета (CTReqPRD) |
| [21:20] | EVNT2_ B_SEL | Поле выбора источника для события 2.<br>00 – равенства таймера значению CMPB (CTReqCMPB);<br>01 – равенства таймера значению CMPA (CTReqCMPA);<br>10 – равенства таймера нулю (CTReqZERO);<br>11 – равенства таймера периоду счета (CTReqPRD) |
| [19:18] | EVNT3_ B_SEL | Поле выбора источника для события 3.<br>00 – равенства таймера значению CMPB (CTReqCMPB);<br>01 – равенства таймера значению CMPA (CTReqCMPA);<br>10 – равенства таймера нулю (CTReqZERO);<br>11 – равенства таймера периоду счета (CTReqPRD) |
| [17:16] | EVNT4_ B_SEL | Поле выбора источника для события 4.<br>00 – равенства таймера значению CMPB (CTReqCMPB);<br>01 – равенства таймера значению CMPA (CTReqCMPA);<br>10 – равенства таймера нулю (CTReqZERO);<br>11 – равенства таймера периоду счета (CTReqPRD) |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [15:14] | EVNT1U_ B_MODE | Возможные действия, по событию 1, в случае, если счетчик инкрементируется:<br>00 - Не делать ничего. Событие игнорируется;<br>01 - Сброс: Переключение EPWMxB в состояние «0»;<br>10 - Установка: Переключение EPWMxB в состояние «1»;<br>11 - Переключение выводов EPWMxB в состояние, определяемое полем I_MODEB регистра AQSFRC |
| [13:12] | EVNT1D_ B_MODE | Возможные действия, по событию 1, в случае, если счетчик декрементируется:<br>00 - Не делать ничего. Событие игнорируется;<br>01 - Сброс: Переключение EPWMxB в состояние «0»;<br>10 - Установка: Переключение EPWMxB в состояние «1»;<br>11 - Переключение выводов EPWMxB в состояние, определяемое полем I_MODEB регистра AQSFRC |
| [11:10] | EVNT2U_ B_MODE | Возможные действия, по событию 2, в случае, если счетчик инкрементируется:<br>00 - Не делать ничего. Событие игнорируется;<br>01 - Сброс: Переключение EPWMxB в состояние «0»;<br>10 - Установка: Переключение EPWMxB в состояние «1»;<br>11 - Переключение выводов EPWMxB в состояние, определяемое полем I_MODEB регистра AQSFRC |
| [9:8] | EVNT2D_ B_MODE | Возможные действия, по событию 2, в случае, если счетчик декрементируется:<br>00 - Не делать ничего. Событие игнорируется;<br>01 - Сброс: Переключение EPWMxB в состояние «0»;<br>10 - Установка: Переключение EPWMxB в состояние «1»;<br>11 - Переключение выводов EPWMxB в состояние, определяемое полем I_MODEB регистра AQSFRC |
| [7:6] | EVNT3U_ B_MODE | Возможные действия, по событию 3, в случае, если счетчик инкрементируется:<br>00 - Не делать ничего. Событие игнорируется;<br>01 - Сброс: Переключение EPWMxB в состояние «0»;<br>10 - Установка: Переключение EPWMxB в состояние «1»;<br>11 - Переключение выводов EPWMxB в состояние, определяемое полем I_MODEB регистра AQSFRC |
| [5:4] | EVNT3D_ B_MODE | Возможные действия, по событию 3, в случае, если счетчик декрементируется:<br>00 - Не делать ничего. Событие игнорируется;<br>01 - Сброс: Переключение EPWMxB в состояние «0»;<br>10 - Установка: Переключение EPWMxB в состояние «1»;<br>11 - Переключение выводов EPWMxB в состояние, определяемое полем I_MODEB регистра AQSFRC |
| [3:2] | EVNT4U_ B_MODE | Возможные действия, по событию 4, в случае, если счетчик инкрементируется:<br>00 - Не делать ничего. Событие игнорируется;<br>01 - Сброс: Переключение EPWMxB в состояние «0»;<br>10 - Установка: Переключение EPWMxB в состояние «1»;<br>11 - Переключение выводов EPWMxB в состояние, определяемое полем I_MODEB регистра AQSFRC |
| [1:0] | EVNT4D_B_MODE | Возможные действия, по событию 4, в случае, если счетчик декрементируется:<br>00 - Не делать ничего. Событие игнорируется;<br>01 - Сброс: Переключение EPWMxB в состояние «0»;<br>10 - Установка: Переключение EPWMxB в состояние «1»;<br>11 - Переключение выводов EPWMxB в состояние, определяемое полем I_MODEB регистра AQSFRC |

### 20.28.3.3 AQSFRC

Значение после сброса: 0x0.

Таблица 424 – Регистр AQSFRC

| Номер | 31…20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 |
| | Зарезервировано | I_MODEA | | I_MODEB | |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | Зарезервировано | | | | | | | | RLDCSF | | OTSFB | ACTSFB | | OTSFA | ACTSFA | |

Таблица 425 – Описание бит регистра AQSFRC

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:20] | Зарезервировано | Зарезервировано |
| [19:18] | I_MODEA | Действие, осуществляемое с выводом EPWMxA по событиям, в соответствии с конфигурацией регистра AQCTLA<br>00 – на вывод назначается инверсия вывода ~EPWMxA;<br>01 – на вывод назначается инверсия вывода ~EPWMxB;<br>10 – на вывод назначается текущее значение вывода EPWMxA;<br>11 – на вывод назначается текущее значение вывода EPWMxB |
| [17:16] | I_MODEB | Действие, осуществляемое с выводом EPWMxB по событиям, в соответствии с конфигурацией регистра AQCTLB<br>00 – на вывод назначается инверсия вывода ~EPWMxB;<br>01 – на вывод назначается инверсия вывода ~EPWMxA;<br>10 – на вывод назначается текущее значение вывода EPWMxB;<br>11 – на вывод назначается текущее значение вывода EPWMxA |
| [15:8] | Зарезервировано | Зарезервировано |
| [7:6] | RLDCSF | Выбор события загрузки регистра AQCSFRC из дублирующих регистров<br>00 – Счетчик равен 0;<br>01 – Счетчик равен периоду (TBPRD);<br>10 – Счетчик равен 0 или периоду (TBPRD);<br>11 – Не использовать регистры дублирования (регистр AQCSFRC записывается непосредственно) |
| 5 | OTSFB | Программное управление выходом B.<br>Запись 0 – нет эффекта;<br>Запись 1 – вывод устанавливается в положении в соответствии со значением поля ACTSFB |
| [4:3] | ACTSFB | Выбор действия при установке бита программного управления выходом B OTSFB:<br>00 – Нет действия;<br>01 – Установить в низкий уровень;<br>10 – Установить в высокий уровень;<br>11 – Действие определяется полем I_MODEB |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 2 | OTSFA | Программное управление выходом A.<br>Запись 0 – нет эффекта;<br>Запись 1 – вывод устанавливается в положении в соответствии со значением поля ACTSFA |
| [1:0] | ACTSFA | Выбор действия при установке бита программного управления выходом A OTSFA:<br>00 – Нет действия;<br>01 – Установить в низкий уровень;<br>10 – Установить в высокий уровень;<br>11 – Действие определяется полем I_MODEA |

#### 20.28.3.4 AQCSFRC

Значение после сброса: 0x0.

Таблица 426 – Регистр AQCSFRC

| Номер | 31…16 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | Зарезервировано |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | Зарезервировано | | | | | | | | | | | | CSFB | | CSFA | |

Таблица 427 – Описание бит регистра ADCSFRC

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:4] | Зарезервировано | Зарезервировано |
| [3:2] | CSFB | Программное переключение выхода B в предопределенное состояние. Значение поля действует по следующему за записью фронту сигнала TBCLK в режиме записи в основной регистр CMPB. В режиме записи в дублирующий буфер установка вывода будет произведена при следующей загрузке регистра из буфера. Состояния вывода определяются полем RLDCSF регистра AQSFRC:<br>00 – не устанавливать вывод в предопределенное состояние;<br>01 – установить вывод B в состояние «0»;<br>10 – установить вывод B в состояние «1»;<br>11 – не устанавливать вывод в предопределенное состояние |
| [1:0] | CSFA | Программное переключение выхода A в предопределенное состояние. Значение поля действует по следующему за записью фронту сигнала TBCLK в режиме записи в основной регистр CMPA. В режиме записи в дублирующий буфер установка вывода будет произведена при следующей загрузке регистра из буфера. Состояния вывода определяются полем RLDCSF регистра AQSFRC:<br>00 – не устанавливать вывод в предопределенное состояние;<br>01 – установить вывод A в состояние «0»;<br>10 – установить вывод A в состояние «1»;<br>11 – не устанавливать вывод в предопределенное состояние |

### 20.28.4 *Регистры блока управления мертвой зоной*

#### 20.28.4.1 DBCTL

Значение после сброса: 0x0.

Таблица 428 – Регистр DBCTL

| Номер | 31…16 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | Зарезервировано |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | Зарезервировано | | | | | | | | | | IN_MODE | | POLSEL | | OUT_ MODE | |

Таблица 429 – Описание бит регистра DBCTL

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:6] | Зарезервировано | Зарезервировано |
| [5:4] | IN_MODE | Полем выбирается входы блоков задержки фронта и среза.<br>Бит 5 управляет мультиплексором S5.<br>Бит 4 управляет мультиплексором S4.<br>00 – сигнал EPWMxA является входом блоков задержки фронта и среза.<br>01 – сигнал EPWMxB является входом блока задержки фронта.<br>сигнал EPWMxA является входом блока задержки среза.<br>10 – сигнал EPWMxA является входом блока задержки фронта.<br>сигнал EPWMxB является входом блока задержки среза.<br>11 – сигнал EPWMxB является входом блоков задержки фронта и среза |
| [3:2] | POLSEL | Инверсия задержанных сигналов.<br>Бит 5 управляет мультиплексором S3.<br>Бит 4 управляет мультиплексором S2.<br>00 – сигнал EPWMxA – прямой;<br>сигнал EPWMxB – прямой;<br>01 – сигнал EPWMxA – инвертирован;<br>сигнал EPWMxB – прямой;<br>10 – сигнал EPWMxA – прямой;<br>сигнал EPWMxB – инвертирован;<br>11 – сигнал EPWMxA – инвертирован;<br>сигнал EPWMxB – инвертирован |
| [1:0] | OUT_MODE | Выбор источника выходных сигналов.<br>Бит 1 управляет мультиплексором S1.<br>Бит 0 управляет мультиплексором S0.<br>00 – сигнал EPWMxA – незадержанный;<br>сигнал EPWMxB – незадержанный;<br>01 – сигнал EPWMxA – задержанный;<br>сигнал EPWMxB – незадержанный;<br>10 – сигнал EPWMxA – незадержанный;<br>сигнал EPWMxB – задержанный;<br>11 – сигнал EPWMxA – задержанный;<br>сигнал EPWMxB – задержанный |

### 20.28.4.2 DBRED

Значение после сброса: 0x0.

Таблица 430 – Регистр DBRED

| Номер | 31…16 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | DEL |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | DEL | | | | | | | | | | | | | | | |

Таблица 431 – Описание бит регистра DBRED

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:0] | DEL | Время задержки по фронту в тактах синхросигнала TCLK<br>31 – 31 такт;<br>…….<br>2 – 2 такта;<br>1 – 1 такт;<br>0 – 1 такт |

### 20.28.4.3 DBFED

Значение после сброса: 0x0.

Таблица 432 – Регистр DBFED

| Номер | 31…28 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | DEL |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | DEL | | | | | | | | | | | | | | | |

Таблица 433 – Описание бит регистра DBFED

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:0] | DEL | Время задержки по срезу в тактах синхросигнала TCLK<br>31 – 31 такт;<br>…….<br>2 – 2 такта;<br>1 – 1 такт;<br>0 – 1 такт |

## 20.28.5 *Регистры блока модуляции высокочастотного сигнала*

### 20.28.5.1 PCCTL

Значение после сброса: 0x0.

Таблица 434 – Регистр PCCTL

| Номер | 31…16 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | Зарезервировано |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | Зарезервировано | | | | | CFPDUTY | | | CHPFREQ | | | OSHTWTH | | | | CHPEN |

Таблица 435 – Описание бит регистра PCCTL

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:11] | Зарезервировано | Зарезервировано |
| [10:8] | CHPDUTY | Регулировка скважности выходного сигнала:<br>000 - Скважность $T_1/T_0$ = 1/8 (12.5%)<br>001 - Скважность $T_1/T_0$ = 2/8 (25.0%)<br>010 - Скважность $T_1/T_0$ = 3/8 (37.5%)<br>011 - Скважность $T_1/T_0$ = 4/8 (50.0%)<br>100 - Скважность $T_1/T_0$ = 5/8 (62.5%)<br>101 - Скважность $T_1/T_0$ = 6/8 (75.0%)<br>110 - Скважность $T_1/T_0$ = 7/8 (87.5%)<br>111 - Зарезервировано |
| [7:5] | CHPFREQ | Частота выходного сигнала:<br>000 - Деление на 1 (12.5 MHz на 100 MHz системной частоты)<br>001 - Деление на 2 (6.25 MHz на 100 MHz системной частоты)<br>010 - Деление на 3 (4.16 MHz на 100 MHz системной частоты)<br>011 - Деление на 4 (3.12 MHz на 100 MHz системной частоты)<br>100 - Деление на 5 (2.50 MHz на 100 MHz системной частоты)<br>101 - Деление на 6 (2.08 MHz на 100 MHz системной частоты)<br>110 - Деление на 7 (1.78 MHz на 100 MHz системной частоты)<br>111 - Деление на 8 (1.56 MHz на 100 MHz системной частоты) |
| [4:1] | OSHTWTH | Ширина первого импульса:<br>0000 - 1 x SYSCLKOUT /8 (= 80нс на 100 MHz системной частоты)<br>0001 - 2 x SYSCLKOUT /8 (= 160нс на 100 MHz системной частоты)<br>0010 - 3 x SYSCLKOUT /8 (= 240нс на 100 MHz системной частоты)<br>0011 - 4 x SYSCLKOUT /8 (= 320нс на 100 MHz системной частоты)<br>0100 - 5 x SYSCLKOUT /8 (= 400нс на 100 MHz системной частоты)<br>0101 - 6 x SYSCLKOUT /8 (= 480нс на 100 MHz системной частоты)<br>0110 - 7 x SYSCLKOUT /8 (= 560нс на 100 MHz системной частоты)<br>0111 - 8 x SYSCLKOUT /8 (= 640нс на 100 MHz системной частоты)<br>1000 - 9 x SYSCLKOUT /8 (= 720нс на 100 MHz системной частоты)<br>1001 - 10 x SYSCLKOUT /8 (= 800нс на 100 MHz системной частоты)<br>1010 - 11 x SYSCLKOUT /8 (= 880нс на 100 MHz системной частоты)<br>1011 - 12 x SYSCLKOUT /8 (= 960нс на 100 MHz системной частоты)<br>1100 - 13 x SYSCLKOUT /8 (= 1040нс на 100 MHz системной частоты)<br>1101 - 14 x SYSCLKOUT /8 (= 1120нс на 100 MHz системной частоты)<br>1110 - 15 x SYSCLKOUT /8 (= 1200нс на 100 MHz системной частоты) |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| | | 1111 - 16 x SYSCLKOUT /8 (= 1280нс на 100 MHz системной частоты) |
| 0 | CHPEN | Разрешение регулировки скважности:<br>0 - запрещено<br>1 - разрешено |

### 20.28.6 *Регистры блока защиты*

#### 20.28.6.1 TZSEL

Значение после сброса: 0x0.

Таблица 436 – Регистр TZSEL

| Номер | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | 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 | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | Зарезервировано | | OSHT6_ INT | OSHT5_ INT | OSHT4_ INT | OSHT3_ INT | OSHT2_ INT | OSHT1_ INT | Зарезервировано | | CBC6_ INT | CBC5_ INT | CBC4_ INT | CBC3_ INT | CBC2_EXT | CBC1_EXT |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | Зарезервировано | | OSHT6 | OSHT5 | OSHT4 | OSHT3 | OSHT2 | OSHT1 | Зарезервировано | | CBC6 | CBC5 | CBC4 | CBC3 | CBC2 | CBC1 |

Таблица 437 – Описание битов регистра TZSEL

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31 | Зарезервировано | Зарезервировано |
| 30 | Зарезервировано | Зарезервировано |
| 29 | OSHT6_ INT | Выбор сигнала TZ6 для одиночного события ошибки INT<br>0 – Запрет использования сигнала TZ6 в качестве источника одиночной ошибки.<br>1 – Разрешение использования сигнала TZ6 в качестве источника одиночной ошибки |
| 28 | OSHT5_ INT | Выбор сигнала TZ5 для одиночного события ошибки INT<br>0 – Запрет использования сигнала TZ5 в качестве источника одиночной ошибки.<br>1 – Разрешение использования сигнала TZ5 в качестTZTCLве источника одиночной ошибки |
| 27 | OSHT4_ INT | Выбор сигнала TZ4 для одиночного события ошибки INT<br>0 – Запрет использования сигнала TZ4 в качестве источника одиночной ошибки. |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| | | 1 – Разрешение использования сигнала TZ4 в качестве источника одиночной ошибки |
| 26 | OSHT3_ INT | Выбор сигнала TZ3 для одиночного события ошибки INT<br>0 – Запрет использования сигнала TZ3 в качестве источника одиночной ошибки.<br>1 – Разрешение использования сигнала TZ3 в качестве источника одиночной ошибки |
| 25 | OSHT2_ INT | Выбор сигнала TZ2 для одиночного события ошибки INT<br>0 – Запрет использования сигнала TZ2 в качестве источника одиночной ошибки.<br>1 – Разрешение использования сигнала TZ2 в качестве источника одиночной ошибки |
| 24 | OSHT1_ INT | Выбор сигнала TZ1 для одиночного события ошибки INT<br>0 – Запрет использования сигнала TZ1 в качестве источника одиночной ошибки.<br>1 – Разрешение использования сигнала TZ1 в качестве источника одиночной ошибки |
| 23 | Зарезервировано | Зарезервировано |
| 22 | Зарезервировано | Зарезервировано |
| 21 | CBC6_ INT | Выбор сигнала TZ6_ для циклического события ошибки INT<br>0 – Запрет использования сигнала TZ6 в качестве источника циклической ошибки.<br>1 – Разрешение использования сигнала TZ6 в качестве источника циклической ошибки |
| 20 | CBC5_ INT | Выбор сигнала TZ5 для циклического события ошибки INT<br>0 – Запрет использования сигнала TZ5 в качестве источника циклической ошибки.<br>1 – Разрешение использования сигнала TZ5 в качестве источника циклической ошибки |
| 19 | CBC4_ INT | Выбор сигнала TZ4 для циклического события ошибки INT<br>0 – Запрет использования сигнала TZ4 в качестве источника циклической ошибки.<br>1 – Разрешение использования сигнала TZ4 в качестве источника циклической ошибки |
| 18 | CBC3_ INT | Выбор сигнала TZ3 для циклического события ошибки INT<br>0 – Запрет использования сигнала TZ3 в качестве источника циклической ошибки.<br>1 – Разрешение использования сигнала TZ3 в качестве источника циклической ошибки |
| 17 | CBC2_ INT | Выбор сигнала TZ2 для циклического события ошибки INT<br>0 – Запрет использования сигнала TZ2 в качестве источника циклической ошибки.<br>1 – Разрешение использования сигнала TZ2 в качестве источника циклической ошибки. |
| 16 | CBC1_INT | Выбор сигнала TZ1 для циклического события ошибки INT<br>0 – Запрет использования сигнала TZ1 в качестве источника циклической ошибки.<br>1 – Разрешение использования сигнала TZ1 в качестве источника циклической ошибки |
| 15 | Зарезервировано | Зарезервировано |
| 14 | Зарезервировано | Зарезервировано |
| 13 | OSHT6_EXT | Выбор сигнала TZ6 для одиночного события ошибки EXT<br>0 – Запрет использования сигнала TZ6 в качестве источника одиночной ошибки.<br>1 – Разрешение использования сигнала TZ6 в качестве источника одиночной ошибки |
| 12 | OSHT5_EXT | Выбор сигнала TZ5 для одиночного события ошибки EXT<br>0 – Запрет использования сигнала TZ5 в качестве источника одиночной ошибки. |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| | | 1 – Разрешение использования сигнала TZ5 в качестве источника одиночной ошибки |
| 11 | OSHT4_EXT | Выбор сигнала TZ4 для одиночного события ошибки EXT<br>0 – Запрет использования сигнала TZ4 в качестве источника одиночной ошибки.<br>1 – Разрешение использования сигнала TZ4 в качестве источника одиночной ошибки |
| 10 | OSHT3_EXT | Выбор сигнала TZ3 для одиночного события ошибки EXT<br>0 – Запрет использования сигнала TZ3 в качестве источника одиночной ошибки.<br>1 – Разрешение использования сигнала TZ3 в качестве источника одиночной ошибки |
| 9 | OSHT2_EXT | Выбор сигнала TZ2 для одиночного события ошибки EXT<br>0 – Запрет использования сигнала TZ2 в качестве источника одиночной ошибки.<br>1 – Разрешение использования сигнала TZ2 в качестве источника одиночной ошибки |
| 8 | OSHT1_EXT | Выбор сигнала TZ1 для одиночного события ошибки EXT<br>0 – Запрет использования сигнала TZ1 в качестве источника одиночной ошибки.<br>1 – Разрешение использования сигнала TZ1 в качестве источника одиночной ошибки |
| 7 | Зарезервировано | Зарезервировано |
| 6 | Зарезервировано | Зарезервировано |
| 5 | CBC6_EXT | Выбор сигнала TZ6_ для циклического события ошибки EXT<br>0 – Запрет использования сигнала TZ6 в качестве источника циклической ошибки.<br>1 – Разрешение использования сигнала TZ6 в качестве источника циклической ошибки |
| 4 | CBC5_EXT | Выбор сигнала TZ5 для циклического события ошибки EXT<br>0 – Запрет использования сигнала TZ5 в качестве источника циклической ошибки.<br>1 – Разрешение использования сигнала TZ5 в качестве источника циклической ошибки |
| 3 | CBC4_EXT | Выбор сигнала TZ4 для циклического события ошибки EXT<br>0 – Запрет использования сигнала TZ4 в качестве источника циклической ошибки.<br>1 – Разрешение использования сигнала TZ4 в качестве источника циклической ошибки |
| 2 | CBC3_EXT | Выбор сигнала TZ3 для циклического события ошибки EXT<br>0 – Запрет использования сигнала TZ3 в качестве источника циклической ошибки.<br>1 – Разрешение использования сигнала TZ3 в качестве источника циклической ошибки |
| 1 | CBC2_EXT | Выбор сигнала TZ2 для циклического события ошибки EXT<br>0 – Запрет использования сигнала TZ2 в качестве источника циклической ошибки.<br>1 – Разрешение использования сигнала TZ2 в качестве источника циклической ошибки |
| 0 | CBC1_EXT | Выбор сигнала TZ1 для циклического события ошибки EXT<br>0 – Запрет использования сигнала TZ1 в качестве источника циклической ошибки.<br>1 – Разрешение использования сигнала TZ1 в качестве источника циклической ошибки |

### 20.28.6.2 TZCTL

Значение после сброса: 0xffffffff.

Таблица 438 – Регистр TZCTL

| Номер | 31…28 | 27, 26 | 25, 24 | 23, 22 | 21, 20 | 19, 18 | 17, 16 |
|---|---|---|---|---|---|---|---|
| Доступ | - | R/W | R/W | R/W | R/W | R/W | R/W |
| Сброс | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | Зарезервировано | TZB_CW_INT | TZA_CW_INT | TZB_CR_INT | TZA_CR_INT | TZB_OS_INT | TZA_OS_INT |

| Номер | 15…12 | 11, 10 | 9, 8 | 7, 6 | 5 ,4 | 3, 2 | 1, 0 |
|---|---|---|---|---|---|---|---|
| Доступ | - | R/W | R/W | R/W | R/W | R/W | R/W |
| Сброс | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | Зарезервировано | TZB_CW_EXT | TZA_CW_EXT | TZB_CR_EXT | TZA_CR_EXT | TZB_OS_EXT | TZA_OS_EXT |

Таблица 439 – Описание бит регистра TZCTL

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:28] | Зарезервировано | Зарезервировано |
| [27:26] | TZB_CW_INT | Выбор действия для выхода EPWMxB по сигналу цепи циклической ошибки INT.<br>00 – высокий импеданс;<br>01 – высокий уровень;<br>10 – низкий уровень;<br>11 – нет действия |
| [25:24] | TZA_CW_INT | Выбор действия для выхода EPWMxA по сигналу цепи циклической ошибки INT.<br>00 – высокий импеданс;<br>01 – высокий уровень;<br>10 – низкий уровень;<br>11 – нет действия |
| [23:22] | TZB_CR_INT | Выбор действия для выхода EPWMxB по сигналу триггера циклической ошибки INT.<br>00 – высокий импеданс;<br>01 – высокий уровень;<br>10 – низкий уровень;<br>11 – нет действия |
| [21:20] | TZA_CR_INT | Выбор действия для выхода EPWMxA по сигналу триггера циклической ошибки INT.<br>00 – высокий импеданс;<br>01 – высокий уровень;<br>10 – низкий уровень;<br>11 – нет действия |
| [19:18] | TZB_OS_INT | Выбор действия для выхода EPWMxB по сигналу триггера одиночной ошибки INT.<br>00 – высокий импеданс;<br>01 – высокий уровень;<br>10 – низкий уровень;<br>11 – нет действия |
| [17:16] | TZA_OS_INT | Выбор действия для выхода EPWMxA по сигналу триггера одиночной ошибки INT.<br>00 – высокий импеданс;<br>01 – высокий уровень;<br>10 – низкий уровень;<br>11 – нет действия |
| [15:12] | Зарезервировано | Зарезервировано |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [11:10] | TZB_CW_EXT | Выбор действия для выхода EPWMxB по сигналу цепи циклической ошибки EXT.<br>00 – высокий импеданс;<br>01 – высокий уровень;<br>10 – низкий уровень;<br>11 – нет действия |
| [9:8] | TZA_CW_EXT | Выбор действия для выхода EPWMxA по сигналу цепи циклической ошибки EXT.<br>00 – высокий импеданс;<br>01 – высокий уровень;<br>10 – низкий уровень;<br>11 – нет действия |
| [7:6] | TZB_CR_EXT | Выбор действия для выхода EPWMxB по сигналу триггера циклической ошибки EXT.<br>00 – высокий импеданс;<br>01 – высокий уровень;<br>10 – низкий уровень;<br>11 – нет действия |
| [5:4] | TZA_CR_EXT | Выбор действия для выхода EPWMxA по сигналу триггера циклической ошибки EXT.<br>00 – высокий импеданс;<br>01 – высокий уровень;<br>10 – низкий уровень;<br>11 – нет действия |
| [3:2] | TZB_OS_EXT | Выбор действия для выхода EPWMxB по сигналу триггера одиночной ошибки EXT.<br>00 – высокий импеданс;<br>01 – высокий уровень;<br>10 – низкий уровень;<br>11 – нет действия |
| [1:0] | TZA_OS_EXT | Выбор действия для выхода EPWMxA по сигналу триггера одиночной ошибки EXT.<br>00 – высокий импеданс;<br>01 – высокий уровень;<br>10 – низкий уровень;<br>11 – нет действия |

#### 20.28.6.3 TZEINT

Значение после сброса: 0x0.

Таблица 440 – Регистр TZEINT

| Номер | 31…16 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | Зарезервировано |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | Зарезервировано | | | | | | | | | | | | OS_INT_IE | OS_EXT_IE | CR_INT_IE | CR_EXT_IE |

Таблица 441 – Описание бит регистра TZEINT

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:4] | Зарезервировано | Зарезервировано |
| 3 | OS_INT_IE | Разрешение прерываний по возникновению одиночной ошибки INT.<br>0 – запрещено;<br>1 - разрешено |
| 2 | OS_EXT_IE | Разрешение прерываний по возникновению одиночной ошибки EXT.<br>0 – запрещено;<br>1 - разрешено |
| 1 | CR_INT_IE | Разрешение прерываний по возникновению циклической ошибки INT.<br>0 – запрещено;<br>1 - разрешено |
| 0 | CR_EXT_IE | Разрешение прерываний по возникновению циклической ошибки EXT.<br>0 – запрещено;<br>1 - разрешено |

#### 20.28.6.4 TZFLG

Значение после сброса: 0x0.

Таблица 442 – Регистр TZFLG

| Номер | 31…16 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | Зарезервировано |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | Зарезервировано | | | | | | | | | | | | OS_INT_FLG | OS_EXT_ FLG | CR_INT_ FLG | CR_EXT_ FLG |

Таблица 443 – Описание бит регистра TZFLG

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:4] | Зарезервировано | Зарезервировано |
| 3 | OS_INT_FLG | Флаг прерываний по возникновению одиночной ошибки INT.<br>0 – Нет прерывания<br>1 – Есть прерывание. Очищается при помощи установки соответсвующего бита регистра TZCLR |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 2 | OS_EXT_ FLG | Флаг прерываний по возникновению одиночной ошибки EXT.<br>0 – Нет прерывания<br>1 – Есть прерывание. Очищается при помощи установки соответствующего бита регистра TZCLR |
| 1 | CR_INT_ FLG | Флаг прерываний по возникновению циклической ошибки INT.<br>0 – Нет прерывания<br>1 – Есть прерывание. Очищается при помощи установки соответствующего бита регистра TZCLR |
| 0 | CR_EXT_ FLG | Флаг прерываний по возникновению циклической ошибки EXT.<br>0 – Нет прерывания<br>1 – Есть прерывание. Очищается при помощи установки соответствующего бита регистра TZCLR |

### 20.28.6.5 TZCLR

Значение после сброса: 0x0.

Таблица 444 – Регистр TZCLR

| Номер | 31…16 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | Зарезервировано |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | Зарезервировано | | | | | | | | | | | | OS_INT_CLR | OS_EXT_ CLR | CR_INT_ CLR | CR_EXT_ CLR |

Таблица 445 – Описание бит регистра TZCLR

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:4] | Зарезервировано | Зарезервировано |
| 3 | OS_INT_CLR | Очищение прерываний по возникновению одиночной ошибки INT.<br>0 – запрещено;<br>1 - разрешено |
| 2 | OS_EXT_ CLR | Очищение прерываний по возникновению одиночной ошибки EXT.<br>0 – запрещено;<br>1 - Разрешено |
| 1 | CR_INT_ CLR | Очищение прерываний по возникновению циклической ошибки INT.<br>0 – запрещено;<br>1 - разрешено |
| 0 | CR_EXT_ CLR | Очищение прерываний по возникновению циклической ошибки EXT.<br>0 – запрещено;<br>1 - разрешено |

### 20.28.6.6 TZFRC

Значение после сброса: 0x0.

Таблица 446 – Регистр TZFRC

| Номер | 31…16 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | Зарезервировано |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | Зарезервировано | | | | | | | | | | | | OS_INT_FRC | OS_EXT_FRC | CR_INT_ FRC | CR_EXT_FRC |

Таблица 447 – Описание бит регистра TZFRC

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:4] | Зарезервировано | Зарезервировано |
| 3 | OS_INT_FRC | Установка прерываний по возникновению одиночной ошибки INT.<br>0 – запрещено;<br>1 - разрешено |
| 2 | OS_EXT_FRC | Установка прерываний по возникновению одиночной ошибки EXT.<br>0 – запрещено;<br>1 - разрешено |
| 1 | CR_INT_ FRC | Установка прерываний по возникновению циклической ошибки INT.<br>0 – запрещено;<br>1 - разрешено |
| 0 | CR_EXT_ FRC | Установка прерываний по возникновению циклической ошибки EXT.<br>0 – запрещено;<br>1 - разрешено |

## 20.28.7 *Регистры блока генерации событий*

### 20.28.7.1 ETSEL

Значение после сброса: 0x0.

Таблица 448 – Регистр ETSEL

| Номер | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | 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 | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | Зарезервировано | SHDWAEMPTY_DMA_EN | SHDWBEMPTY_DMA_EN | SHDWPRDEMPTY_DMA_EN | Зарезервировано | | | | | | SHDWAFULL_EN | SHDWAEMPTY_EN | SHDWBFULL_EN | SHDWBEMPTY_EN | SHDWPRDFULL_EN | SHDWPRDEMPTY_EN |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | SOCBEN | SOCBSEL | | | SOCAEN | SOCASEL | | | Зарезервировано | | | | INTEN | INTSEL | | |

Таблица 449 - Описание бит регистра ETSEL

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31 | Зарезервировано | Зарезервировано |
| 30 | SHDWAEMPTY_DMA_EN | Бит разрешения выдачи запроса DMA по опустошению буфера CMPA<br>0 – разрешено;<br>1 –запрещено |
| 29 | SHDWBEMPTY_DMA_EN | Бит разрешения выдачи запроса DMA по опустошению буфера CMPB<br>0 – разрешено;<br>1 –запрещено |
| 28 | SHDWPRDEMPTY_DMA_EN | Бит разрешения выдачи запроса DMA по опустошению буфера PRD<br>0 – разрешено;<br>1 –запрещено |
| [27:22] | Зарезервировано | Зарезервировано |
| 21 | SHDWAFULL_EN | Бит разрешения выдачи прерывания по заполнению буфера CMPA<br>0 – разрешено;<br>1 –запрещено |
| 20 | SHDWAEMPTY_EN | Бит разрешения выдачи прерывания по опустошению буфера CMPA<br>0 – разрешено;<br>1 –запрещено |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 19 | SHDWBFULL_EN | Бит разрешения выдачи прерывания по заполнению буфера CMPB<br>0 – разрешено;<br>1 –запрещено |
| 18 | SHDWBEMPTY_EN | Бит разрешения выдачи прерывания по опустошению буфера CMPB<br>0 – разрешено;<br>1 –запрещено |
| 17 | SHDWPRDFULL_EN | Бит разрешения выдачи прерывания по заполнению буфера PRD<br>0 – разрешено;<br>1 –запрещено |
| 16 | SHDWPRDEMPTY_EN | Бит разрешения выдачи прерывания по опустошению буфера PRD<br>0 – разрешено;<br>1 –запрещено; |
| 15 | SOCBEN | Разрешение выдачи импульса синхронизации АЦП EPWMxSOCB.<br>0 – Не разрешено;<br>1 – Разрешено |
| 14..12 | SOCBSEL | Выбор события-источника импульса синхронизации АЦП EPWMxSOCB.<br>000 – Запрещено;<br>001 – Выдача импульса по событию (TBCTR = 0x0000);<br>010 – Выдача импульса по событию (TBCTR = TBPRD);<br>011 – Зарезервировано;<br>100 – Выдача импульса по событию (TBCTR = CMPA) во время инкремента TBCTR;<br>101 – Выдача импульса по событию (TBCTR = CMPA) во время декремента TBCTR;<br>110 – Выдача импульса по событию (TBCTR = CMPB) во время инкремента TBCTR;<br>111 – Выдача импульса по событию (TBCTR = CMPB) во время декремента TBCTR |
| 11 | SOCAEN | Разрешение выдачи импульса синхронизации АЦП EPWMxSOCA.<br>0 – Не разрешено;<br>1 – Разрешено |
| 10..8 | SOCASEL | Выбор события-источника импульса синхронизации АЦП EPWMxSOCA.<br>000 – Запрещено;<br>001 – Выдача импульса по событию (TBCTR = 0x0000);<br>010 – Выдача импульса по событию (TBCTR = TBPRD);<br>011 – Зарезервировано;<br>100 – Выдача импульса по событию (TBCTR = CMPA) во время инкремента TBCTR;<br>101 – Выдача импульса по событию (TBCTR = CMPA) во время декремента TBCTR;<br>110 – Выдача импульса по событию (TBCTR = CMPB) во время инкремента TBCTR;<br>111 – Выдача импульса по событию (TBCTR = CMPB) во время декремента TBCTR |
| 3 | INTEN | Разрешение прерывания EPWMx_INT блока ШИМ<br>0 – Запрещено;<br>1 – Разрешено |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 2..0 | INTSEL | Выбор события-источника прерывания EPWMx_INT.<br>000 – Запрещено;<br>001 – Прерывание по событию (TBCTR = 0x0000);<br>010 – Прерывание по событию (TBCTR = TBPRD);<br>011 – Зарезервировано;<br>100 – Прерывание по событию (TBCTR = CMPA) во время инкремента TBCTR;<br>101 – Прерывание по событию (TBCTR = CMPA) во время декремента TBCTR;<br>110 – Прерывание по событию (TBCTR = CMPB) во время инкремента TBCTR;<br>111 – Прерывание по событию (TBCTR = CMPB) во время декремента TBCTR |

### 20.28.7.2 ETPS

Значение после сброса: 0x0.

Таблица 450 – Регистр ETPS

| Номер | 31…16 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | Зарезервировано |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | SOCBCNT | | SOCBPRD | | SOCACNT | | SOCAPRD | | Зарезервировано | | | | INTCNT | | INTPRD | |

Таблица 451 – Описание бит регистра ETPS

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:16] | Зарезервировано | Зарезервировано |
| [15:14] | SOCBCNT | Счетчик количества событий на выводе EPWMxSOCB.<br>00 – Событий не было;<br>01 – 1 событие;<br>10 – 2 событие;<br>11 – 3 событие |
| [13:12] | SOCBPRD | Поле определяет количество событий, выбранных полем SOCBSEL регистра ETSEL для генерации импульса EPWMxSOCB.<br>00 – Генерация события EPWMxSOCB по первому событию.<br>01 – Генерация события EPWMxSOCB по второму событию.<br>10 – Генерация события EPWMxSOCB по третьему событию.<br>11 – Генерация события EPWMxSOCB по четвертому событию |
| [11:10] | SOCACNT | Счетчик количества событий на выводе EPWMxSOCA.<br>00 – Событий не было;<br>01 – 1 событие;<br>10 – 2 событие;<br>11 – 3 событие |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [9:8] | SOCAPRD | Поле определяет количество событий, выбранных полем SOCASEL регистра ETSEL для генерации импульса EPWMxSOCA.<br>00 – Генерация события EPWMxSOCA по первому событию.<br>01 – Генерация события EPWMxSOCA по второму событию.<br>10 – Генерация события EPWMxSOCA по третьему событию.<br>11 – Генерация события EPWMxSOCA по четвертому событию |
| [7:4] | Зарезервировано | |
| [3:2] | INTCNT | Счетчик количества событий прерывания.<br>00 – Событий не было.<br>01 – 1 событие.<br>10 – 2 событие.<br>11 – 3 событие |
| [1:0] | INTPRD | Поле определяет количество событий, выбранных полем INTSEL регистра ETSEL для генерации прерывания.<br>00 – Счетчик выключен. Прерывания не генерируются.<br>01 – Генерация прерывания по первому событию.<br>10 – Генерация прерывания по второму событию.<br>11 – Генерация прерывания по третьему событию |

### 20.28.7.3 ETFLG

Значение после сброса: 0x150000.

Таблица 452 – Регистр ETFLG

| Номер | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | 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 | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| | Зарезервировано | | | | | | | | | | SHDWAFULL | SHDWAEMPTY | SHDWBFULL | SHDWBEMPTY | SHDWPRDFULL | SHDWPRDEMPTY |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | Зарезервировано | | | | | | | | | | | | SOCB | SOCA | Зарезервировано | INT |

Таблица 453 – Описание бит регистра ETFLG

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:22] | Зарезервировано | Зарезервировано |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 21 | SHDWAFULL | Флаг события SHDWAFULL<br>0 – нет действия;<br>1 – событие было |
| 20 | SHDWAEMPTY | Флаг события SHDWAEMPTY<br>0 – нет действия;<br>1 – событие было |
| 19 | SHDWBFULL | Флаг события SHDWBFULL<br>0 – нет действия;<br>1 – событие было |
| 18 | SHDWBEMPTY | Флаг события SHDWBEMPTY<br>0 – нет действия;<br>1 – событие было |
| 17 | SHDWPRDFULL | Флаг события SHDWPRDFULL<br>0 – нет действия;<br>1 – событие было |
| 16 | SHDWPRDEMPTY | Флаг события SHDWPRDEMPTY<br>0 – нет действия;<br>1 – событие было |
| [15:4] | Зарезервировано | Зарезервировано |
| 3 | SOCB | Флаг события SOCB.<br>0 – нет действия;<br>1 – событие SOCBFLG |
| 2 | SOCA | Флаг события SOCA.<br>0 – нет действия;<br>1 – событие SOCAFLG |
| 1 | Зарезервировано | Зарезервировано |
| 0 | INT | Флаг прерывания.<br>0 – нет прерывания;<br>1 – есть прерывание |

#### 20.28.7.4 ETCLR

Значение после сброса: 0x0.

Таблица 454 – Регистр ETCLR

| Номер | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | 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 | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | Зарезервировано | | | | | | | | | | SHDWAFULL | SHDWAEMPTY | SHDWBFULL | SHDWBEMPTY | SHDWPRDFULL | SHDWPRDEMPTY |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |

| | Зарезервировано | SOCB | SOCA | Зарезервировано | INT |
|---|---|---|---|---|---|

Таблица 455 – Описание бит регистра ETCLR

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:22] | Зарезервировано | Зарезервировано |
| 21 | SHDWAFULL | Флаг события SHDWAFULL<br>0 – Нет действия;<br>1 – Событие было |
| 20 | SHDWAEMPTY | Флаг события SHDWAEMPTY<br>0 – Нет действия;<br>1 – Событие было |
| 19 | SHDWBFULL | Флаг события SHDWBFULL<br>0 – Нет действия;<br>1 – Событие было |
| 18 | SHDWBEMPTY | Флаг события SHDWBEMPTY<br>0 – Нет действия;<br>1 – Событие было |
| 17 | SHDWPRDFULL | Флаг события SHDWPRDFULL<br>0 – Нет действия;<br>1 – Событие было |
| 16 | SHDWPRDEMPTY | Флаг события SHDWPRDEMPTY<br>0 – Нет действия;<br>1 – Событие было |
| [15:4] | Зарезервировано | Зарезервировано |
| 3 | SOCB | Программная очистка SOCB.<br>0 – Нет действия;<br>1 – Очистка флага SOCBFLG |
| 2 | SOCA | Программная очистка SOCA.<br>0 – Нет действия;<br>1 – Очистка флага SOCAFLG |
| 1 | Зарезервировано | Зарезервировано |
| 0 | INT | Программная очистка флага прерывания.<br>0 – Нет действия;<br>1 – Очистка флага прерывания |

#### 20.28.7.5 ETFRC

Значение после сброса: 0x0.

Таблица 456 – Регистр ETFRC

| Номер | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | 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 | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Зарезервировано | | | | | | | | | | | SHDWAFULL | SHDWAEMPTY | SHDWBFULL | SHDWBEMPTY | SHDWPRDFULL | SHDWPRDEMPTY |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | Зарезервировано | | | | | | | | | | | | SOCB | SOCA | Зарезервировано | INT |

Таблица 457 – Описание бит регистра ETFRC

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:22] | Зарезервировано | Зарезервировано |
| 21 | SHDWAFULL | Флаг события SHDWAFULL<br>0 – Нет действия;<br>1 – Событие было |
| 20 | SHDWAEMPTY | Флаг события SHDWAEMPTY<br>0 – Нет действия;<br>1 – Событие было |
| 19 | SHDWBFULL | Флаг события SHDWBFULL<br>0 – Нет действия;<br>1 – Событие было |
| 18 | SHDWBEMPTY | Флаг события SHDWBEMPTY<br>0 – Нет действия;<br>1 – Событие было |
| 17 | SHDWPRDFULL | Флаг события SHDWPRDFULL<br>0 – Нет действия;<br>1 – Событие было |
| 16 | SHDWPRDEMPTY | Флаг события SHDWPRDEMPTY<br>0 – Нет действия;<br>1 – Событие было |
| [15:4] | Зарезервировано | Зарезервировано |
| 3 | SOCB | Программная установка SOCB.<br>Сигнал устанавливается в зависимости от значения бита разрешения.<br>Флаг SOCB регистра ETFLG устанавливается независимо от разрешения.<br>0 – Нет действия;<br>1 – Генерируется импульс на выводе EPWMxSOCB и устанавливается флаг SOCBFLG. Используется для целей тестирования |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 2 | SOCA | Программная установка SOCA.<br>Сигнал устанавливается в зависимости от значения бита разрешения.<br>Флаг SOCA регистра ETFLG устанавливается независимо от разрешения.<br>0 – Нет действия;<br>1 – Генерируется импульс на выводе EPWMxSOCA и устанавливается флаг SOCAFLG. Используется для целей тестирования |
| 1 | Зарезервировано | Зарезервировано |
| 0 | INT | Программная установка прерывания.<br>Сигнал устанавливается в зависимости от значения бита разрешения.<br>Флаг устанавливается независимо от разрешения.<br>0 – Нет действия;<br>1 – Генерируется прерывание |

### 20.28.8 *Регистры блока управления ШИМ высокого разрешения HRPWM_CNTR*

#### 20.28.8.1 HRPWM_CNTR

Значение после сброса: 0x0.

Таблица 458 – Регистр HRPWM_CNTR

| Base ADDR= | **0x4009_E000**<br>**0x4009_F000** | Offset= | **0x0000_007C** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | HRPWM_01_CNTR | | | | |
| | HRPWM_23_CNTR | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | HRPWM1_EN | Зарезервировано | | | HRPWM1_SHIFT | | | | | Зарезервировано | HRPWM1_OE | HRPWM1_MX | |
| r | r | r | rw | r | r | r | rw | rw | rw | rw | rw | r | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | HRPWM0_EN | Зарезервировано | | | HRPWM0_SHIFT | | | | | Зарезервировано | HRPWM0_OE | HRPWM0_MX | |
| r | r | r | rw | r | r | r | rw | rw | rw | rw | rw | r | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…29 | Зарезервировано | 0 | Резерв |
| 28 | HRPWM1_EN | 0 | Включение блока ШИМ высокого разрешения HRPWM1<br>0 – выключен;<br>1 – включен |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 27…25 | Зарезервировано | 0 | Резерв |
| 24…20 | HRPWM1_SHIFT | 0 | Задание значение сдвига фронта выходного сигнала в пределах периода рабочей частоты<br>0-15 – уменьшить скважность<br>16-31 – увеличить скважность |
| 19 | Зарезервировано | 0 | Резерв |
| 18 | HRPWM1_OE | 0 | Включение выходного буфера блока ШИМ высокого разрешения HRPWM1<br>0 – буфер выключен;<br>1 – буфер включен<br>Для работы в режиме HRPWM вывод должен быть сконфигурирован в аналоговый режим |
| 17…16 | HRPWM1_MX | 0 | Выбор режима работы блока HRPWM1<br>00 – на вывод в функции HRPWM1 поступает сигнал цифрового ШИМ;<br>01 – вывод в функции HRPWM1работает в режиме высокоточного ШИМ (основной режим работы блока);<br>10 – вывод в функции HRPWM1 работает в режиме высокоточного инверсного ШИМ;<br>11 – недопустимое значение |
| 15…13 | Зарезервировано | 0 | Резерв |
| 12 | HRPWM0_EN | 0 | Включение блока ШИМ высокого разрешения HRPWM0<br>0 – выключен;<br>1 – включен |
| 11…9 | Зарезервировано | 0 | Резерв |
| 8…4 | HRPWM0_SHIFT | 0 | Задание значение сдвига фронта выходного сигнала в пределах периода рабочей частоты<br>0-15 – уменьшить скважность<br>16-31 – увеличить скважность |
| 3 | Зарезервировано | 0 | Резерв |
| 2 | HRPWM0_OE | 0 | Включение выходного буфера блока ШИМ высокого разрешения HRPWM0<br>0 – буфер выключен;<br>1 – буфер включен<br>Для работы в режиме HRPWM вывод должен быть сконфигурирован в аналоговый режим |
| 1…0 | HRPWM0_MX | 0 | Выбор режима работы блока HRPWM0<br>00 – на вывод в функции HRPWM0 поступает сигнал цифрового ШИМ;<br>01 – вывод в функции HRPWM0 работает в режиме высокоточного ШИМ (основной режим работы блока);<br>10 – вывод в функции HRPWM0 работает в режиме высокоточного инверсного ШИМ;<br>11 – недопустимое значение |

## 20.29 Описание регистров контроллеров таймеров общего назначения TIMER_CNTR

Таблица 459 – Регистры контроллеров таймеров общего назначения

| Базовый адрес | Название | Описание |
|---|---|---|
| 0x4009_4000 | TIMER0_CNTR | Контроллер TIMER0_CNTR |
| 0x4009_5000 | TIMER1_CNTR | Контроллер TIMER1_CNTR |
| 0x4009_6000 | TIMER2_CNTR | Контроллер TIMER2_CNTR |
| 0x4009_7000 | TIMER3_CNTR | Контроллер TIMER3_CNTR |
| **Смещение** | | |
| 0x00 | CNT | Основной счетчик таймера |
| 0x04 | PSG | Делитель частоты TIM_CLK для тактирования основного счетчика |
| 0x08 | ARR | Основание счета основного счетчика |
| 0x0C | CNTRL | Регистр управления основным счетчиком |
| 0x50 | BRKETR_CNTRL | Регистр управления входом BRK и ETR |
| 0x54 | STATUS | Регистр статуса таймера |
| 0x58 | IE | Регистр разрешения прерываний |
| 0x5C | DMA_RE0 | Регистр разрешения формирования запроса DMA – TMRx[0] |
| 0x80 | DMA_RE1 | Регистр разрешения формирования запроса DMA – TMRx[1] |
| 0x84 | DMA_RE2 | Регистр разрешения формирования запроса DMA – TMRx[2] |
| 0x88 | DMA_RE3 | Регистр разрешения формирования запроса DMA – TMRx[3] |
| 0x8C | DMA_RE4 | Регистр разрешения формирования запроса DMA – TMRx[4] |
| Канал 1 | | |
| 0x10 | CH1_CCR | Регистр сравнения/захвата для 1 канала таймера |
| 0x20 | CH1_CNTRL0 | Регистр управления 0 для 1 канала таймера |
| 0x30 | CH1_CNTRL1 | Регистр управления 1 для 1 канала таймера |
| 0x40 | CH1_DTG | Регистр управления DTG для 1 канала таймера |
| 0x60 | CH1_CNTRL2 | Регистр управления 2 для 1 канала таймера |
| 0x70 | CH1_CCR1 | Регистр сравнения/захвата 1 для 1 канала таймера |
| Канал 2 | | |
| 0x14 | CH2_CCR | Регистр сравнения/захвата для 2 канала таймера |
| 0x24 | CH2_CNTRL0 | Регистр управления 0 для 2 канала таймера |
| 0x34 | CH2_CNTRL1 | Регистр управления 1 для 2 канала таймера |
| 0x44 | CH2_DTG | Регистр управления DTG для 2 канала таймера |
| 0x64 | CH2_CNTRL2 | Регистр управления 2 для 2 канала таймера |
| 0x74 | CH2_CCR1 | Регистр сравнения/захвата 1 для 2 канала таймера |
| Канал 3 | | |
| 0x18 | CH3_CCR | Регистр сравнения/захвата для 3 канала таймера |
| 0x28 | CH3_CNTRL0 | Регистр управления 0 для 3 канала таймера |
| 0x38 | CH3_CNTRL1 | Регистр управления 1 для 3 канала таймера |
| 0x48 | CH3_DTG | Регистр управления DTG для 3 канала таймера |
| 0x68 | CH3_CNTRL2 | Регистр управления 2 для 3 канала таймера |
| 0x78 | CH3_CCR1 | Регистр сравнения/захвата 1 для 3 канала таймера |
| Канал 4 | | |
| 0x1C | CH4_CCR | Регистр сравнения/захвата для 4 канала таймера |
| 0x2C | CH4_CNTRL0 | Регистр управления 0 для 4 канала таймера |
| 0x3C | CH4_CNTRL1 | Регистр управления 1 для 4 канала таймера |
| 0x4C | CH4_DTG | Регистр управления DTG для 4 канала таймера |
| 0x6C | CH4_CNTRL2 | Регистр управления 2 для 4 канала таймера |
| 0x7C | CH4_CCR1 | Регистр сравнения/захвата 1 для 4 канала таймера |

### 20.29.1 *CNT*

Таблица 460 – Основной счетчик таймера CNT

| Номер | 31…0 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | CNT[31:0] |

Таблица 461 – Описание бит регистра CNT

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31…0 | CNT[31:0] | Значение основного счетчика таймера |

### 20.29.2 *PSG*

Таблица 462 – Делитель частоты TIM_CLK для счета основного счетчика PSG

| Номер | 31..0 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | PSG[31:0] |

Таблица 463 – Описание бит регистра PSG

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31…0 | PSG[31:0] | Значение предварительного делителя счетчика.<br>Основной счетчик считает на частоте:<br>TIM_CLKd = TIM_CLK/(PSG+1) |

### 20.29.3 *ARR*

Таблица 464 – Основание счета основного счетчика ARR

| Номер | 31…0 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | ARR[31:0] |

Таблица 465 – Описание бит регистра ARR

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31…0 | ARR[31:0] | Основание счета для основного счетчика.<br>CNT = [0…ARR] |

### 20.29.4 *CNTRL*

Таблица 466 – Регистр управления основным счетчиком CNTRL

| Номер | 31…12 | 11…8 |
|---|---|---|
| Доступ | U | R/W |
| Сброс | 0 | 0 |
| | - | EVNT_SEL[3:0] |

| Номер | 7…6 | 5…9 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W |
| Сброс | 00 | 00 | 0 | 0 | 0 | 0 |
| | CNT_MODE[1:0] | FDTS[1:0] | DIR | WR_CMPL | ARRB_EN | CNT_EN |

Таблица 467 – Описание бит регистра CNTRL

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений. |
|---|---|---|
| 31…12 | - | Зарезервировано. |
| 11...8 | EVNT_SEL[3:0] | Биты выбора источника событий:<br>0000 – внутренняя тактовая частота TIM_CLKd (формируется путем деления частоты TIM_CLK);<br>0001 – событие на TMR_EVNT0;<br>0010 – событие на TMR_EVNT1;<br>0011 – событие на TMR_EVNT2;<br>0100 – событие переднего фронта на канале 1, «Режим 1»;<br>0101 – событие переднего фронта на канале 2, «Режим 1»;<br>0110 – событие переднего фронта на канале 3, «Режим 1»;<br>0111 – событие переднего фронта на канале 4, «Режим 1»;<br>1000 – событие переднего фронта на ETR, «Режим 2»;<br>1001 – событие заднего фронта на ETR, «Режим 2»;<br>1010 – событие на TMR_EVNT3;<br>1011-1111 – зарезервировано.<br>Примечание: подключение входов таймера TMR_EVNTx к выходам запуска других таймеров приведено на рисунке |
| 7..6 | CNT_MODE[1:0] | Режим счета основного счетчика:<br>00 – счетчик прямой при DIR = 0;<br>счетчик обратный при DIR = 1;<br>01 – счетчик двунаправленный с автоматическим изменением DIR при CNT == 0 или CNT == ARR;<br>10 – счетчик прямой при DIR = 0;<br>счетчик обратный при DIR = 1;<br>11 – зарезервировано.<br>Режим счета CNT_MODE[1:0] необходимо устанавливать в соответствии со значением в поле EVNT_SEL[3:0]:<br>- EVNT_SEL[3:0] = 0000: CNT_MODE[1:0] = 00 или 01;<br>- EVNT_SEL[3:0] != 0000: CNT_MODE[1:0] = 10 |
| 5...4 | FDTS[1:0] | Делитель тактовой частоты $F_{DTS}$:<br>00 – $F_{DTS}$ = TIM_CLK;<br>01 – $F_{DTS}$ = TIM_CLK/2;<br>10 – $F_{DTS}$ = TIM_CLK/3;<br>11 – $F_{DTS}$ = TIM_CLK/4 |
| 3 | DIR | Направление счета основного счетчика:<br>0 – прямой, от 0 до ARR;<br>1 – обратный, от ARR до 0 |
| 2 | WR_CMPL | Флаг выполнения записи нового значения в регистры CNT, PSG и ARR:<br>0 – новые данные можно записывать;<br>1 – данные не записаны и идет запись |
| 1 | ARRB_EN | Режим обновления регистра ARR:<br>0 – ARR будет перезаписан в момент записи в ARR;<br>1 – ARR будет перезаписан при CNT == ARR |
| 0 | CNT_EN | Разрешение работы таймера:<br>0 – таймер отключен;<br>1 – таймер включен |

### 20.29.5 *CHy_CCR*

Таблица 468 – Регистр сравнения/захвата для ‘y’ канала таймера CHy_CCR

| Номер | 31…0 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | CCR[31:0] |

Таблица 469 – Описание бит регистра CHy_CCR

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31…0 | CCR[31:0] | Режим захвата: значение CNT, при котором произошел факт захвата события.<br>Режим ШИМ: значение CCR, с которым сравнивается CNT |

### 20.29.6 *CHy_CCR1*

Таблица 470 – Регистр сравнения/захвата для ‘y’ канала таймера CHy_CCR1

| Номер | 31…0 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | CCR1[31:0] |

Таблица 471 – Описание бит регистра CHy_CCR1

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31…0 | CCR1[31:0] | Режим захвата: значение CNT, при котором произошел факт захвата события.<br>Режим ШИМ: значение CCR1, с которым сравнивается CNT |

### 20.29.7 *CHy_CNTRL0*

Таблица 472 – Регистр управления для ‘y’ канала таймера CHy_CNTRL0

| Номер | 31…17 | 16 | 15 | 14 | 13 |
|---|---|---|---|---|---|
| Доступ | U | RO | R/W | RO | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 |
| | - | WR_CMPL1 | CAP_NPWM | WR_CMPL | ETR_EN |

| Номер | 12 | 11…9 | 8 | 7…6 | 5…4 | 3…0 |
|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W |
| Сброс | 0 | 000 | 0 | 00 | 00 | 0000 |
| | BRK_EN | OCCM[2:0] | OCCE | CH_PSC[1:0] | CH_SEL[1:0] | CH_FLTR[3:0] |

Таблица 473 – Описание бит регистра CHy_CNTRL0

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31...17 | - | Зарезервировано. |
| 16 | WR_CMPL1 | Флаг выполнения записи нового значения в регистр CHy_CCR1:<br>0 – новые данные можно записывать;<br>1 – данные не записаны и идет запись |
| 15 | CAP_NPWM | Режим работы канала:<br>0 – канал работает в режиме ШИМ;<br>1 – канал работает в режиме захвата |
| 14 | WR_CMPL | Флаг выполнения записи нового значения в регистр CHy_CCR:<br>0 – новые данные можно записывать;<br>1 – данные не записаны и идет запись |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 13 | ETR_EN | Разрешение сброса сигнала REF в «0» при высоком уровне на входе ETR:<br>0 – запрещен;<br>1 – разрешен |
| 12 | BRK_EN | Разрешение сброса сигналов REF и DTG в «0» при низком уровне на входе BRK:<br>0 – запрещен;<br>1 – разрешен |
| 11…9 | OCCM[2:0] | Формат выработки сигнала REF в режиме ШИМ:<br>Если CCR1_EN=0:<br>000 – всегда 0;<br>001 – 1, если CNT==CCR;<br>010 – 0, если CNT==CCR;<br>011 – переключение REF, если CNT==CCR;<br>100 – всегда 0;<br>101 – всегда 1;<br>110 – 1, если DIR=0 (счет прямой), CNT<CCR, иначе 0;<br>0, если DIR=1 (счет обратный), CNT>CCR, иначе 1;<br>111 – 0, если DIR=0 (счет прямой), CNT<CCR, иначе 1;<br>1, если DIR=1 (счет обратный), CNT>CCR, иначе 0.<br>Если CCR1_EN=1:<br>000 – всегда 0;<br>001 – 1, если CNT==CCR или CNT==CCR1;<br>010 – 0, если CNT==CCR или CNT==CCR1;<br>011 – переключение REF, если CNT==CCR или CNT==CCR1;<br>100 – всегда 0;<br>101 – всегда 1;<br>110 – 0, если DIR=0 (счет прямой), CCR≤CNT≤CCR1, иначе 1;<br>0, если DIR=1 (счет обратный), CCR<CNT<CCR1, иначе 1;<br>111 – 1, если DIR=0 (счет прямой), CCR≤CNT≤CCR1, иначе 0;<br>1, если DIR=1 (счет обратный), CCR<CN <CCR1, иначе 0;<br>Необходимо соблюдать условие CCR<CCR1 |
| 8 | OCCE | Разрешение работы ETR:<br>0 – запрещен;<br>1 – разрешен |
| 7…6 | CH_PSC[1:0] | Предварительный делитель входного канала:<br>00 – нет деления;<br>01 – /2;<br>10 – /4;<br>11 – /8 |
| 5…4 | CH_SEL[1:0] | Выбор события по входному каналу CHyi для фиксации значения основного счетчика (регистр CNT) в регистр CHy_CCR:<br>00 – положительный фронт на входном канале CHyi;<br>01 – отрицательный фронт на входном канале CHyi;<br>10 – положительный фронт от других каналов:<br>для первого канала от второго канала;<br>для второго канала от третьего канала;<br>для третьего канала от четвертого канала;<br>для четвертого канала от первого канала;<br>11 – положительный фронт от других каналов:<br>для первого канала от третьего канала;<br>для второго канала от четвертого канала;<br>для третьего канала от первого канала;<br>для четвертого канала от второго канала |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 3…0 | CH_FLTR[3:0] | Конфигурация фильтра на входе канала ‘y’. Выбор частоты выборки Fs и количества выборок N:<br>0000 – нет фильтрации, $F_S = F_{DTS}$;<br>0001 – $F_S$ = TIM_CLK, N = 2;<br>0010 – $F_S$ = TIM_CLK, N = 4;<br>0011 – $F_S$ = TIM_CLK, N = 8;<br>0100 – $F_S = F_{DTS}/2$, N = 6;<br>0101 – $F_S = F_{DTS}/2$, N = 8;<br>0110 – $F_S = F_{DTS}/4$, N = 6;<br>0111 – $F_S = F_{DTS}/4$, N = 8;<br>1000 – $F_S = F_{DTS}/8$, N = 6;<br>1001 – $F_S = F_{DTS}/8$, N = 8;<br>1010 – $F_S = F_{DTS}/16$, N = 5;<br>1011 – $F_S = F_{DTS}/16$, N = 6;<br>1100 – $F_S = F_{DTS}/16$, N = 8;<br>1101 – $F_S = F_{DTS}/32$, N = 5;<br>1110 – $F_S = F_{DTS}/32$, N = 6;<br>1111 – $F_S = F_{DTS}/32$, N = 8 |

### 20.29.8 *CHy_CNTRL1*

Таблица 474 – 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 | 00 | 00 | 0 | 0 | 00 | 00 |
| | - | NINV | NSELO [1:0] | NSELOE [1:0] | - | INV | SELO [1:0] | SELOE [1:0] |

Таблица 475 – Описание бит регистра CHy_CNTRL1

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений. |
|---|---|---|
| 31...13 | - | Зарезервировано. |
| 12 | NINV | Инверсия инверсного выхода nCHy:<br>0 – выход не инвертируется;<br>1 – выход инвертируется |
| 11…10 | NSELO[1:0] | Выбор источника сигнала для инверсного выхода nCHy:<br>00 – на nCHyo выдается 0;<br>01 – на nCHyo выдается 1;<br>10 – на nCHyo выдается сигнал nREF;<br>11 – на nCHyo выдается сигнал с DTG |
| 9…8 | NSELOE[1:0] | Режим работы инверсного выхода nCHy:<br>00 – на nCHyoe выдается 0;<br>01 – на nCHyoe выдается 1;<br>10 – на nCHyoe выдается сигнал nREF;<br>11 – на nCHyoe выдается сигнал с DTG.<br>При nCHyoe = 0 вывод канала в третьем состоянии,<br>при nCHyoe = 1 вывод канала работает в режиме выхода |
| 7…5 | - | Зарезервировано |
| 4 | INV | Инверсия прямого выхода CHy:<br>0 – выход не инвертируется;<br>1 – выход инвертируется |
| 3…2 | SELO[1:0] | Выбор источника сигнала для прямого выхода CHy:<br>00 – на CHyo выдается 0;<br>01 – на CHyo выдается 1;<br>10 – на CHyo выдается сигнал REF;<br>11 – на CHyo выдается сигнал с DTG |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений. |
|---|---|---|
| 1…0 | SELOE[1:0] | Режим работы прямого выхода CHy:<br>00 – на CHyое выдается 0;<br>01 – на CHyое выдается 1;<br>10 – на CHyое выдается сигнал REF;<br>11 – на CHyое выдается сигнал с DTG.<br>При CHyое = 0 вывод канала работает в режиме входа,<br>при CHyое = 1 вывод канала работает в режиме выхода |

### 20.29.9 *CHy_CNTRL2*

Таблица 476 – 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 |
| | - | EVNT_DLY | CCR_RLD | CCR1_EN | CH_SEL1[1:0] |

Таблица 477 – Описание бит регистра CHy_CNTRL2

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31...5 | - | Зарезервировано |
| 4 | EVNT_DLY | Задержка события захвата до обновления регистров CHy_CCR и CHy_CCR1:<br>0 – сигнал события захвата устанавливается в момент обнаружения события, при этом обновление регистров CHy_CCR и CHy_CCR1 выполняется через один такт TIM_CLK;<br>1 – сигнал события захвата устанавливается синхронно с обновлением информации в регистрах CHy_CCR и CHy_CCR1 |
| 3 | CCR_RLD | Режим обновления регистров CHy_CCR и CHy_CCR1:<br>0 – обновление возможно в любой момент времени;<br>1 – обновление будет осуществлено только при CNT == 0 |
| 2 | CCR1_EN | Разрешение работы регистра CHy_CCR1:<br>0 – CHy_CCR1 не используется;<br>1 – CHy_CCR1 используется |
| 1…0 | CH_SEL1[1:0] | Выбор события по входному каналу CHyi для фиксации значения основного счетчика (регистр CNT) в регистр CHy_CCR1:<br>00 – положительный фронт на входном канале CHyi;<br>01 – отрицательный фронт на входном канале CHyi;<br>10 – отрицательный фронт от других каналов:<br>для первого канала от второго канала;<br>для второго канала от третьего канала;<br>для третьего канала от четвертого канала;<br>для четвертого канала от первого канала;<br>11 – отрицательный фронт от других каналов:<br>для первого канала от третьего канала;<br>для второго канала от четвертого канала;<br>для третьего канала от первого канала;<br>для четвертого канала от второго канала |

### 20.29.10 *CHy_DTG*

Таблица 478 – Регистр управления генератором «мертвой зоны» CHy_DTG

| Номер | 31…16 | 15…8 | 7…5 | 4 | 3…0 |
|---|---|---|---|---|---|
| Доступ | U | R/W | U | R/W | R/W |
| Сброс | 0 | 00000000 | 000 | 0 | 0000 |
| | - | DTG[7:0] | - | EDTS | DTGx[3:0] |

Таблица 479 – Описание бит регистра CHy_DTG

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31…16 | - | Зарезервировано |
| 15…8 | DTG[7:0] | Основной делитель частоты DTG.<br>Задержка DTGdel = DTG • (DTGx + 1) |
| 7…5 | - | Зарезервировано |
| 4 | EDTS | Частота работы DTG:<br>0 – TIM_CLK;<br>1 – $F_{DTS}$ |
| 3…0 | DTGx[3:0] | Предварительный делитель частоты DTG |

### 20.29.11 *BRKETR_CNTRL*

Таблица 480 – Регистр BRKETR_CNTRL управления входом BRK и ETR

| Номер | 31…8 | 7…4 | 3…2 | 1 | 0 |
|---|---|---|---|---|---|
| Доступ | U | R/W | R/W | R/W | R/W |
| Сброс | 0 | 0000 | 00 | 0 | 0 |
| | - | ETR_FLTR[3:0] | ETR_PSC[1:0] | ETR_INV | BRK_INV |

Таблица 481 – Описание бит регистра BRKETR_CNTRL

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31...8 | - | Зарезервировано |
| 7…4 | ETR_FLTR[3:0] | Конфигурация фильтра на входе ETR. Выбор частоты выборки Fs и количества выборок N:<br>0000 – нет фильтрации, $F_S = F_{DTS}$;<br>0001 – $F_S$ = TIM_CLK, N = 2;<br>0010 – $F_S$ = TIM_CLK, N = 4;<br>0011 – $F_S$ = TIM_CLK, N = 8;<br>0100 – $F_S = F_{DTS}/2$, N = 6;<br>0101 – $F_S = F_{DTS}/2$, N = 8;<br>0110 – $F_S = F_{DTS}/4$, N = 6;<br>0111 – $F_S = F_{DTS}/4$, N = 8;<br>1000 – $F_S = F_{DTS}/8$, N = 6;<br>1001 – $F_S = F_{DTS}/8$, N = 8;<br>1010 – $F_S = F_{DTS}/16$, N = 5;<br>1011 – $F_S = F_{DTS}/16$, N = 6;<br>1100 – $F_S = F_{DTS}/16$, N = 8;<br>1101 – $F_S = F_{DTS}/32$, N = 5;<br>1110 – $F_S = F_{DTS}/32$, N = 6;<br>1111 – $F_S = F_{DTS}/32$, N = 8 |
| 3…2 | ETR_PSC[1:0] | Асинхронный предделитель частоты со входа ETR:<br>00 – без деления;<br>01 – /2;<br>10 – /4;<br>11 – /8 |
| 1 | ETR_INV | Инверсия входа ETR:<br>0 – без инверсии;<br>1 – инверсия |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 0 | BRK_INV | Инверсия входа BRK:<br>0 – без инверсии;<br>1 – инверсия |

### 20.29.12 *STATUS*

Таблица 482 – Регистр статуса таймера STATUS

| Номер | 31...17 | 16…13 | 12…9 |
|---|---|---|---|
| Доступ | U | R/W | R/W |
| Сброс | 0 | 0 | 0 |
| | - | CCR_CAP1_EVNT[3:0] | CCR_REF_EVNT[3:0] |

| Номер | 8…5 | 4 | 3 |
|---|---|---|---|
| Доступ | R/W | R/W | R/W |
| Сброс | 0 | 0 | 0 |
| | CCR_CAP_EVNT[3:0] | BRK_EVNT | ETR_FE_EVNT |

| Номер | 2 | 1 | 0 |
|---|---|---|---|
| Доступ | R/W | R/W | R/W |
| Сброс | 0 | 0 | 0 |
| | ETR_RE_EVNT | CNT_ARR_EVNT | CNT_ZERO_EVNT |

Таблица 483 – Описание бит регистра STATUS

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31…17 | - | Зарезервировано |
| 16…13 | CCR_CAP1_EVNT[3:0] | Событие записи значения счетчика CNT в регистр CHy_CCR1 по захвату настроенного фронта на входе канала CHyi:<br>0 – нет события;<br>1 – есть событие.<br>Сбрасывается записью «0», если запись происходит одновременно с новым событием, приоритет у нового события.<br>Бит 0 – первый канал;<br>Бит 3 – четвертый канал |
| 12…9 | CCR_REF_EVNT[3:0] | Событие переднего фронта на выходе генератора опорного сигнала REF:<br>0 – нет события;<br>1 – есть событие.<br>Сбрасывается записью «0», если запись происходит одновременно с новым событием, приоритет у нового события.<br>Бит 0 – первый канал;<br>Бит 3 – четвертый канал |
| 8…5 | CCR_CAP_EVNT[3:0] | Событие записи значения счетчика CNT в регистр CHy_CCR по захвату настроенного фронта на входе канала CHyi:<br>0 – нет события;<br>1 – есть событие.<br>Сбрасывается записью «0», если запись происходит одновременно с новым событием, приоритет у нового события.<br>Бит 0 – первый канал;<br>Бит 3 – четвертый канал |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 4 | BRK_EVNT | Событие высокого уровня на входе BRK:<br>0 – нет события;<br>1 – есть событие.<br>Сбрасывается записью «0», при условии наличия низкого уровня на входе BRK |
| 3 | ETR_FE_EVNT | Событие заднего фронта на входе ETR:<br>0 – нет события;<br>1 – есть событие.<br>Сбрасывается записью «0», если запись происходит одновременно с новым событием, приоритет у нового события |
| 2 | ETR_RE_EVNT | Событие переднего фронта на входе ETR:<br>0 – нет события;<br>1 – есть событие.<br>Сбрасывается записью «0», если запись происходит одновременно с новым событием, приоритет у нового события |
| 1 | CNT_ARR_EVNT | Событие совпадения CNT с ARR:<br>0 – нет события;<br>1 – есть событие.<br>Сбрасывается записью «0», если запись происходит одновременно с новым событием совпадения, приоритет у нового события.<br>Если с момента совпадения до момента программного сброса флага регистры CNT и ARR не изменили состояния, то флаг повторно не взводится |
| 0 | CNT_ZERO_EVNT | Событие совпадения CNT с нулем:<br>0 – нет события;<br>1 – есть событие.<br>Сбрасывается записью «0», если запись происходит одновременно с новым событием совпадения, приоритет у нового события.<br>Если с момента совпадения до момента программного сброса флага регистр CNT не изменил состояния, то флаг повторно не взводится |

Примечание – Для организации надежного сброса статусных флагов прерываний необходимо учитывать требования, изложенные в пункте 22.1.2 «Обработчики исключений».

### 20.29.13 *IE*

Таблица 484 – Регистр разрешения прерываний таймера IE

| Номер | 31...17 | 16…13 | 12…9 |
|---|---|---|---|
| Доступ | U | R/W | R/W |
| Сброс | 0 | 0 | 0 |
| | - | CCR_CAP1_EVNT_IE[3:0] | CCR_REF_EVNT_IE[3:0] |

| Номер | 8…5 | 4 | 3 |
|---|---|---|---|
| Доступ | R/W | R/W | R/W |
| Сброс | 0 | 0 | 0 |
| | CCR_CAP_EVNT_IE[3:0] | BRK_EVNT_IE | ETR_FE_EVNT_IE |

| Номер | 2 | 1 | 0 |
|---|---|---|---|
| Доступ | R/W | R/W | R/W |
| Сброс | 0 | 0 | 0 |
| | ETR_RE_EVNT_IE | CNT_ARR_EVNT_IE | CNT_ZERO_EVNT_IE |

Таблица 485 – Описание бит регистра IE

| **Номер бита** | **Функциональное имя бита** | **Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений** |
|---|---|---|
| 31…17 | - | Зарезервировано |
| 16...13 | CCR_CAP1_EVNT_IE[3:0] | Флаг разрешения прерывания по событию CCR_CAP1_EVNT[3:0] в регистре STATUS:<br>0 – прерывание запрещено;<br>1 – прерывание разрешено.<br>Бит 0 – первый канал;<br>Бит 3 – четвертый канал |
| 12…9 | CCR_REF_EVNT_IE[3:0] | Флаг разрешения прерывания по событию CCR_REF_EVNT[3:0] в регистре STATUS:<br>0 – прерывание запрещено;<br>1 – прерывание разрешено.<br>Бит 0 – первый канал;<br>Бит 3 – четвертый канал |
| 8…5 | CCR_CAP_EVNT_IE[3:0] | Флаг разрешения прерывания по событию CCR_CAP_EVNT[3:0] в регистре STATUS:<br>0 – прерывание запрещено;<br>1 – прерывание разрешено.<br>Бит 0 – первый канал;<br>Бит 3 – четвертый канал |
| 4 | BRK_EVNT_IE | Флаг разрешения прерывания по событию BRK_EVNT в регистре STATUS:<br>0 – прерывание запрещено;<br>1 – прерывание разрешено |
| 3 | ETR_FE_EVNT_IE | Флаг разрешения прерывания по событию ETR_FE_EVNT в регистре STATUS:<br>0 – прерывание запрещено;<br>1 – прерывание разрешено |
| 2 | ETR_RE_EVNT_IE | Флаг разрешения прерывания по событию ETR_RE_EVNT в регистре STATUS:<br>0 – прерывание запрещено;<br>1 – прерывание разрешено |
| 1 | CNT_ARR_EVNT_IE | Флаг разрешения прерывания по событию CNT_ARR_EVNT в регистре STATUS:<br>0 – прерывание запрещено;<br>1 – прерывание разрешено |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 0 | CNT_ZERO_EVNT_IE | Флаг разрешения прерывания по событию CNT_ZERO_EVNT в регистре STATUS:<br>0 – прерывание запрещено;<br>1 – прерывание разрешено |

### 20.29.14 *DMA_RE0-DMA_RE4*

Таблица 486 – Регистр DMA_RE0-DMA_RE4 разрешения запроса DMA

| Номер | 31...17 | 16…13 | 12…9 |
|---|---|---|---|
| Доступ | U | R/W | R/W |
| Сброс | 0 | 0 | 0 |
| | - | CCR_CAP1_EVNT_RE[3:0] | CCR_REF_EVNT_RE[3:0] |

| Номер | 8…5 | 4 | 3 |
|---|---|---|---|
| Доступ | R/W | R/W | R/W |
| Сброс | 0 | 0 | 0 |
| | CCR_CAP_EVNT_RE[3:0] | BRK_EVNT_RE | ETR_FE_EVNT_RE |

| Номер | 2 | 1 | 0 |
|---|---|---|---|
| Доступ | R/W | R/W | R/W |
| Сброс | 0 | 0 | 0 |
| | ETR_RE_EVNT_RE | CNT_ARR_EVNT_RE | CNT_ZERO_EVNT_RE |

Таблица 487 – Описание бит регистра DMA_RE0-DMA_RE4

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31...17 | - | Зарезервировано |
| 16…13 | CCR_CAP1_EVNT_RE[3:0] | Флаг разрешения запроса DMA по событию CCR_CAP1_EVNT[3:0]:<br>0 – запрос DMA запрещен;<br>1 – запрос DMA разрешен.<br>Бит 0 – первый канал;<br>Бит 3 – четвертый канал |
| 12…9 | CCR_REF_EVNT_RE[3:0] | Флаг разрешения запроса DMA по событию CCR_REF_EVNT[3:0]:<br>0 – запрос DMA запрещен;<br>1 – запрос DMA разрешен.<br>Бит 0 – первый канал;<br>Бит 3 – четвертый канал |
| 8…5 | CCR_CAP_EVNT_RE [3:0] | Флаг разрешения запроса DMA по событию CCR_CAP_EVNT[3:0]:<br>0 – запрос DMA запрещен;<br>1 – запрос DMA разрешен.<br>Бит 0 – первый канал;<br>Бит 3 – четвертый канал |
| 4 | BRK_EVNT_RE | Флаг разрешения запроса DMA по событию BRK_EVNT:<br>0 – запрос DMA запрещен;<br>1 – запрос DMA разрешен |
| 3 | ETR_FE_EVNT_RE | Флаг разрешения запроса DMA по событию ETR_FE_EVNT:<br>0 – запрос DMA запрещен;<br>1 – запрос DMA разрешен |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 2 | ETR_RE_EVNT_RE | Флаг разрешения запроса DMA по событию ETR_RE_EVNT:<br>0 – запрос DMA запрещен;<br>1 – запрос DMA разрешен |
| 1 | CNT_ARR_EVNT_RE | Флаг разрешения запроса DMA по событию CNT_ARR_EVNT:<br>0 – запрос DMA запрещен;<br>1 – запрос DMA разрешен |
| 0 | CNT_ZERO_EVNT_RE | Флаг разрешения запроса DMA по событию CNT_ZERO_EVNT:<br>0 – запрос DMA запрещен;<br>1 – запрос DMA разрешен |

## 20.30 Описание регистров контроллеров квадратурных декодеров QEP_CNTR

Таблица 488 – Регистры контроллеров квадратурных декодеров

| Базовый Адрес | Название | Описание |
|---|---|---|
| 0x4009_C000 | QEP0_CNTR | Контроллер QEP0. |
| 0x4009_D000 | QEP1_CNTR | Контроллер QEP1. |
| **Смещение** | | |
| 0x00 | QDECCNTR | Регистр управления квадратурным декодером. |
| 0x04 | QEPCNTR | Регистр управления. |
| 0x08 | QPOSCNTR | Регистр управления компаратором положения. |
| 0x0C | QCAPCNTR | Регистр управления захватом фронтов. |
| 0x10 | QPOSCNT | Регистр счетчика положения. |
| 0x14 | QPOSINIT | Регистр инициализации счетчика положения. |
| 0x18 | QPOSMAX | Регистр максимального значения счетчика положения. |
| 0x1C | QPOSCMP | Регистр сравнения положения. |
| 0x20 | QPOSILAT | Регистр сохраненного значения счетчика положения по событию индекса. |
| 0x24 | QPOSSLAT | Регистр сохраненного значения счетчика положения по событию стробирования. |
| 0x28 | QPOSLAT | Регистр сохраненного значения счетчика положения по событию временных отсчетов. |
| 0x2C | QUTMR | Регистр таймера временных отсчетов. |
| 0x30 | QUPRD | Регистр периода таймера временных отсчетов. |
| 0x34 | QWDTMR | Регистр сторожевого таймера. |
| 0x38 | QEINT | Регистр разрешения прерывания. |
| 0x3C | QFLG | Регистр флагов прерывания. |
| 0x40 | QCLR | Регистр сброса флагов прерывания. |
| 0x44 | QFRC | Регистр установки флагов прерывания. |
| 0x48 | QEPSTS | Регистр статуса. |
| 0x4C | QWDPRD | Регистр периода сторожевого таймера. |
| 0x50 | QCTMR | Регистр таймера захвата фронтов. |
| 0x54 | QCTMR1 | Зарезервировано. |
| 0x58 | QCTMR2 | Зарезервировано. |
| 0x5C | QCTMR3 | Зарезервировано. |
| 0x60 | QCTMR4 | Зарезервировано. |
| 0x64 | QCPRD | Регистр периода таймера захвата фронтов. |
| 0x68 | QCTMRLAT | Регистр сохраненного значения таймера захвата фронтов. |
| 0x6C | QCPRDLAT | Регистр сохраненного значения периода таймера захвата фронтов. |

### 20.30.1 *QDECCNTR*

Таблица 489 – Регистр QDECCNTR

| Номер | 31…16 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | - |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | QSRC | | SOEN | SPSEL | XCR | SWAP | IGATE | QAP | QBP | QIP | QSP | - | | | | |

Таблица 490 - Описание бит регистра QDECCNTR

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:16] | - | Зарезервировано. |
| [15:14] | QSRC | Выбор режима работы блока квадратурного декодера:<br>00 – режим квадратурного счета (QCLK = iCLK, QDIR = iDIR);<br>01 – режим счета/направления (QCLK = xCLK, QDIR = xDIR);<br>10 – режим счета вверх (QCLK = xCLK, QDIR = 1);<br>11 – режим счета вниз (QCLK = xCLK, QDIR = 0). |
| 13 | SOEN | Разрешение выдачи сигнала синхронизации по событию совпадения положения:<br>0 – разрешение выдачи сигнала синхронизации по событию совпадения положения;<br>1 – запрет выдачи сигнала синхронизации по событию совпадения положения. |
| 12 | SPSEL | Выбор источника выходного сигнала синхронизации:<br>0 – для сигнала синхронизации используется выход индексации;<br>1 – для сигнала синхронизации используется выход стробирования. |
| 11 | XCR | Скорость счета по фронтам сигнала EQEPA квадратурного декодера в режиме счета/направления, счета вверх и счета вниз:<br>0 – удвоенная, считаются передние и задние фронты;<br>1 – одиночная, считаются только передние фронты. |
| 10 | SWAP | Смена квадратурных входов (изменяет направление счета счетчика положения при тех же входных значениях):<br>0 – квадратурные входы в прямом режиме;<br>1 – квадратурные входы заменены местами. |
| 9 | IGATE | Объединение входов индекса и стробирования на выходе QI блока квадратурного декодера:<br>0 – запрет объединения по «И» сигнала индекса (QEPI) и стробирования (QEPS) на выходе QI блока квадратурного декодера;<br>1 – объединение по «И» входов импульса (QEPI) и стробирования (QEPS) на выходе QI блока квадратурного декодера. |
| 8 | QAP | Полярность входного сигнала EQEPx_AIN:<br>0 – сигнал не инвертирован;<br>1 – инвертирует сигнал на входе EQEPx_AIN. |
| 7 | QBP | Полярность входного сигнала EQEPx_BIN:<br>0 – сигнал не инвертирован;<br>1 – инвертирует сигнал на входе EQEPx_BIN. |
| 6 | QIP | Полярность входного сигнала EQEPx_IIN:<br>0 – сигнал не инвертирован;<br>1 – Инвертирует сигнал на входе EQEPx_IIN. |
| 5 | QSP | Полярность входного сигнала EQEPx_SIN:<br>0 – сигнал не инвертирован; |

| | | |
|---|---|---|
| | | 1 – инвертирует сигнал на входе EQEPx_SIN. |
| [4:0] | - | Зарезервировано. |

### 20.30.2 *QEPCNTR*

Таблица 491 – Регистр QEPCNTR

| Номер | 31…16 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | - |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | - | | PCRM | | SEI | | IEI | | SWI | SEL | IEL | | QPEN | QCLM | UTE | WDE |

Таблица 492 – Описание бит регистра QEPCNTR

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:14] | - | Зарезервировано. |
| [13:12] | PCRM | Выбор режима сброса счетчика положения:<br>00 – счетчик положения сбрасывается по маркеру индексации;<br>01 – счетчик положения сбрасывается по достижении максимального значения QPOSMAX;<br>10 – счетчик положения сбрасывается по первому событию полного оборота QI, далее сбрасывается по достижении максимального значения QPOSMAX;<br>11 – счетчик положения сбрасывается по событию блока временных отсчетов (QUTMR = QUPRD). |
| [11:10] | SEI | Выбор режима инициализации счетчика положения значением регистра QPOSINIT по импульсу стробирования QS:<br>00 – счетчик не инициализируется по событиям QS;<br>01 – счетчик не инициализируется по событиям QS;<br>10 – инициализация счетчика положения по фронту сигнала стробирования QS;<br>11 – инициализация счетчика положения по переднему фронту сигнала стробирования QS при счете вверх (QDIR = 1) и по заднему фронту сигнала стробирования QS при счете вниз (QDIR = 0). |
| [9:8] | IEI | Выбор режима инициализации счетчика положения значением регистра QPOSINIT по событию полного оборота:<br>00 – счетчик не инициализируется по событиям QI;<br>01 – счетчик не инициализируется по событиям QI;<br>10 – инициализация счетчика положения по фронту сигнала инициализации QI;<br>10 – инициализация счетчика положения по срезу сигнала инициализации QI. |
| 7 | SWI | Программная инициализация счетчика положения значением регистра QPOSINIT:<br>Запись 0 – не влияет на работу устройства.<br>Запись 1 – инициализация счетчика положения значением регистра. Бит не очищается автоматически.<br>Чтение – текущее состояние бита, соответствует ранее записанному значению. |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 6 | SEL | Выбор режима сохранения счетчика положения в регистр QPOSSLAT по событию стробирования:<br>0 – значение счетчика положения сохраняется в регистр QPOSSLAT по переднему фронту сигнала стробирования QS;<br>Стробирование по заднему фронту может быть реализовано путем инверсии входного сигнала стробирования битом QSP регистра QDECCNTR;<br>1 – значение счетчика положения сохраняется в регистр QPOSSLAT по переднему фронту сигнала QS в случае вращения в прямом направлении (QDIR = 1) и по заднему фронту сигнала QS в случае вращения в обратном направлении (QDIR = 0). |
| [5:4] | IEL | Выбор режима сохранения счетчика положения в регистр QPOSILAT по событию полного оборота QI:<br>00 – значение счетчика положения по сигналу QI на сохраняется;<br>01 – сохранение значения счетчика положения в регистр QPOSILAT по фронту сигнала QI;<br>10 – сохранение значения счетчика положения в регистр QPOSILAT по срезу сигнала QI;<br>11 – сохранение значения счетчика положения в регистр QPOSILAT по маркеру индексации.<br>При PCRM = 00 сохранение по событию полного оборота всегда осуществляется по маркеру индексации независимо от выбранного режима сохранения IEL. |
| 3 | QPEN | Разрешение работы контроллера квадратурных преобразователей:<br>0 – контроллер находится в сбросе;<br>1 – контроллер находится в рабочем режиме. |
| 2 | QCLM | Режим сохранения счетчика захвата фронтов:<br>0 – сохранение по чтению процессором счетчика положения;<br>1 – сохранение по сигналу от таймера временных отсчетов. |
| 1 | UTE | Разрешение работы блока временных отсчетов UTIME:<br>0 – блок находится в сбросе;<br>1 – блок находится в рабочем режиме. |
| 0 | WDE | Разрешение работы блока сторожевого таймера QWDOG:<br>0 – блок находится в сбросе;<br>1 – блок находится в рабочем режиме. |

### 20.30.3 *QPOSCNTR*

Таблица 493 – Регистр QPOSCNTR

| Номер | 31…16 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | - |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | PCSHDW | PCLOAD | PCPOL | PCE | PCSPW | | | | | | | | | | | |

Таблица 494 - Описание бит регистра QPOSCNTR

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:16] | - | Зарезервировано. |
| 15 | PCSHDW | Управление режимом отложенной записи во внутренний регистр сравнения положения:<br>0 – внутренний регистр сравнения не используется, значение сравнение соответствует значению регистра QPOSCMP;<br>1 –значение сравнение соответствует значению внутреннего регистра сравнения, загружаемому значением регистра QPOSCMP по событиям, определяемым PCLOSD. |
| 14 | PCLOSD | Выбор условия для отложенной записи во внутренний регистр сравнения положения из регистра QPOSCMP:<br>0 – запись по событию QPOSCNT = 0;<br>1 – запись по событию QPOSCNT = QPOSCMP. |
| 13 | PCPOL | Полярность выходного сигнала синхронизации:<br>0 – активный 1;<br>1 – активный 0 |
| 12 | PCE | Разрешение/запрет работы компаратора счетчика положения:<br>0 – компаратор находится в сбросе;<br>1 – компаратор находится в рабочем режиме. |
| [11:0] | PCSPW | Выбор ширины сигнала выхода импульса синхронизации по равенству положений (в циклах частоты QEP_CLK):<br>0x000 – $1*4*T_{QEP_CLK}$;<br>0x001 – $2*4*T_{QEP_CLK}$;<br>…<br>0xFFF – $4096*4*T_{QEP_CLK}$. |

### 20.30.4 *QCAPCNTR*

Таблица 495 – Регистр QCAPCNTR

| Номер | 31…16 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | - |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | CEN | BM | - | | | | | | | CCPS | | | UPPS | | | |

Таблица 496 – Описание бит регистра QCAPCNTR

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:16] | - | Зарезервировано. |
| 15 | CEN | Разрешение работы блока захвата фронтов:<br>0 – запрещено;<br>1 – разрешено. |
| 14 | BM | Разрешение режима захвата нескольких фронтов:<br>0 – запрещено;<br>1 – разрешено. Применимо при UPPS ≤ 1. |
| [13:7] | - | Зарезервировано. |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [6:4] | CCPS | Делитель синхросигнала счетчиков блока захвата фронтов:<br>000 – $F_{CAP_CLK} = F_{QEP_CLK}/1$;<br>001 – $F_{CAP_CLK} = F_{QEP_CLK}/2$;<br>010 – $F_{CAP_CLK} = F_{QEP_CLK}/4$;<br>011 – $F_{CAP_CLK} = F_{QEP_CLK}/8$;<br>100 – $F_{CAP_CLK} = F_{QEP_CLK}/16$;<br>101 – $F_{CAP_CLK} = F_{QEP_CLK}/32$;<br>110 – $F_{CAP_CLK} = F_{QEP_CLK}/64$;<br>111 – $F_{CAP_CLK} = F_{QEP_CLK}/128$. |
| [3:0] | UPPS | Делитель частоты возникновения событий сброса счетчиком блока захвата фронтов:<br>0000 – $F_{UPEVNT} = F_{QCLK}/1$;<br>0001 – $F_{UPEVNT} = F_{QCLK}/2$;<br>…<br>1010 – $F_{UPEVNT} = F_{QCLK}/1024$;<br>1011 – $F_{UPEVNT} = F_{QCLK}/2048$;<br>11xx – $F_{UPEVNT} = F_{QCLK}/1$. |

### 20.30.5 *QPOSCNT*

Таблица 497 – Регистр QPOSCNT

| Номер | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | 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 |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | QPOSCNT | | | | | | | | | | | | | | |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | QPOSCNT | | | | | | | | | | | | | | | |

Таблица 498 - Описание бит регистра QPOSCNT

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:0] | QPOSCNT | 32-х разрядный счетчик положения. |

### 20.30.6 *QPOSINIT*

Таблица 499 – Регистр QPOSINIT

| Номер | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | 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 | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | QPOSINIT | | | | | | | | | | | | | | | |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | QPOSINIT | | | | | | | | | | | | | | | |

Таблица 500 – Описание бит регистра QPOSINIT

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 15...0 | QPOSINIT | Регистр начального значения положения. Регистр QPOSCNT инициализируется значением QPOSINIT по событиям индексации, стробирования или программной инициализации. |

### 20.30.7 *QPOSMAX*

Таблица 501 – Регистр QPOSMAX

| Номер | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | 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 | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | QPOSMAX | | | | | | | | | | | | | | | |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | QPOSMAX | | | | | | | | | | | | | | | |

Таблица 502 – Описание бит регистра QPOSMAX

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31...0 | QPOSMAX | Максимальное значение счетчика положения. |

### 20.30.8 *QPOSCMP*

Таблица 503 – Регистр QPOSCMP

| Номер | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | 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 | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | QPOSCMP | | | | | | | | | | | | | | | |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | QPOSCMP | | | | | | | | | | | | | | | |

Таблица 504 – Описание бит регистра QPOSCMP

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:0] | QPOSCMP | Значение счетчика положения для сравнения. По достижении счетчиком положения QPOSCNT значения регистра QPOSCMP компаратором счетчика положения может быть выработан сигнал синхронизации или прерывания совпадения положения. |

### 20.30.9 *QPOSILAT*

Таблица 505 – Регистр QPOSILAT

| Номер | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | 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 | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | QPOSILAT | | | | | | | | | | | | | | | |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | QPOSILAT | | | | | | | | | | | | | | | |

Таблица 506 – Описание бит регистра QPOSILAT

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:0] | QPOSILAT | Значение регистра счетчика положения QPOSCNT записывается в регистр QPOSILAT по событию индексации (полного оборота). |

### 20.30.10 *QPOSSLAT*

Таблица 507 – Регистр QPOSSLAT

| Номер | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | 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 | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | QPOSSLAT | | | | | | | | | | | | | | | |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | QPOSSLAT | | | | | | | | | | | | | | | |

Таблица 508 – Описание бит регистра QPOSSLAT

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:0] | QPOSSLAT | Значение регистра счетчика положения QPOSCNT записывается в регистр QPOSSLAT по событию стробирования. |

### 20.30.11 *QPOSLAT*

Таблица 509 – Регистр QPOSLAT

| Номер | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | 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 | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | QPOSLAT | | | | | | | | | | | | | | | |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | QPOSLAT | | | | | | | | | | | | | | | |

Таблица 510 – Описание бит регистра QPOSLAT

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:0] | QPOSLAT | Значение регистра счетчика положения QPOSCNT записывается в регистр QPOSLAT по событию временных отсчетов блока таймера временных отсчетов. |

### 20.30.12 *QUTMR*

Таблица 511 – Регистр QUTMR

| Номер | 31…16 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | QUTMR |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | QUTMR | | | | | | | | | | | | | | | |

Таблица 512 – Описание бит регистра QUTMR

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:0] | QUTMR | Регистр временных отчетов блока таймера временных отсчетов. |

### 20.30.13 *QUPRD*

Таблица 513 – Регистр QUPRD

| Номер | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | 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 | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | QUPRD | | | | | | | | | | | | | | | |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | QUPRD | | | | | | | | | | | | | | | |

Таблица 514 - Описание бит регистра QUPRD

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:0] | QUPRD | Регистр периода временных отчетов блока таймера временных отсчетов. |

### 20.30.14 *QWDTMR*

Таблица 515 – Регистр QWDTMR

| Номер | 31…16 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | - |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | QWDTMR | | | | | | | | | | | | | | | |

Таблица 516 – Описание бит регистра QWDTMR

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:16] | - | Зарезервировано. |
| [15:0] | QWDTMR | Регистр сторожевого таймера. |

### 20.30.15 *QWDPRD*

Таблица 517 – Регистр QWDPRD

| Номер | 31…16 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | - |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | QWDPRD | | | | | | | | | | | | | | | |

Таблица 518 – Описание бит регистра QWDPRD

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:16] | - | Зарезервировано. |
| [15:0] | QWDPRD | Регистр периода сторожевого таймера. Обновляется программно. |

### 20.30.16 *QEINT*

Таблица 519 – Регистр QEINT

| Номер | 31…16 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | - |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | - | | | | UTO | IEL | SEL | PCM | PCR | PCO | PCU | WTO | QDC | PHE | PCE | - |

Таблица 520 – Описание бит регистра QEINT

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:12] | - | Зарезервировано. |
| 11 | UTO | Разрешение формирования прерывания по событию достижения счетчиком QUTMR таймера временных отсчетов UTIME значения периода QUPRD:<br>0 – запрещено;<br>1 – разрешено. |
| 10 | IEL | Разрешение формирования прерывания по событию сохранения значения QPOSCNT счетчика таймера PCCU в регистр QPOSILAT по событию полного оборота:<br>0 – запрещено;<br>1 – разрешено. |
| 9 | SEL | Разрешение формирования прерывания по событию сохранения значения QPOSCNT счетчика таймера PCCU в регистр QPOSSLAT по фронту сигнала стробирования QS:<br>0 – запрещено;<br>1 – разрешено. |
| 8 | PCM | Разрешение формирования прерывания по событию успешного сравнения значения QPOSCNT счетчика таймера PCCU с значением сравнения QPOSCMP (QPOSCNT = QPOSCMP):<br>0 – запрещено;<br>1 – разрешено. |
| 7 | PCR | Разрешение формирования прерывания по готовности к отложенной записи значения сравнения из регистра QPOSCMP компаратора положения:<br>0 – запрещено;<br>1 – разрешено. |
| 6 | PCO | Разрешение формирования прерывания по событию переполнения счетчика положения PCCU (QPOSCNT ≥ QPOSMAX) при счете вверх:<br>0 – запрещено;<br>1 – разрешено. |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 5 | PCU | Разрешение формирования прерывания по событию переполнения счетчика положения PCCU (QPOSCNT = 0) при счете вниз:<br>0 – запрещено;<br>1 – разрешено. |
| 4 | WTO | Разрешение формирования прерывания по достижении счетчиком QWDTMR сторожевого таймера QWDOG значения периода QWDPRD:<br>0 – запрещено;<br>1 – разрешено. |
| 3 | QDC | Разрешение формирования прерывания по событию смены направления движения счетчика положения PCCU:<br>0 – запрещено;<br>1 – разрешено. |
| 2 | PHE | Разрешение формирования прерывания по событию ошибки фазы входных сигналов квадратурного декодера QDU:<br>0 – запрещено;<br>1 – разрешено. |
| 1 | PCE | Разрешение формирования прерывания по событию ошибки значения счетчика положения PCCU при сбросе по событию полного оборота:<br>0 – запрещено;<br>1 – разрешено. |
| 0 | - | Зарезервировано. |

### 20.30.17 *QFLG*

Таблица 521 – Регистр QFLG

| Номер | 31…16 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | - |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | - | | | | UTO | IEL | SEL | PCM | PCR | PCO | PCU | WTO | QDC | PHE | PCE | INT |

Таблица 522 – Описание бит регистра QFLG

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:12] | - | Зарезервировано. |
| 11 | UTO | Флаг события достижения счетчиком QUTMR таймера временных отсчетов UTIME значения периода QUPRD:<br>0 – нет события;<br>1 – есть событие. |
| 10 | IEL | Флаг события сохранения значения QPOSCNT счетчика таймера PCCU в регистр QPOSILAT по событию полного оборота:<br>0 – нет события;<br>1 – есть событие. |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 9 | SEL | Флаг события сохранения значения QPOSCNT счетчика таймера PCCU в регистр QPOSSLAT по фронту сигнала стробирования QS:<br>0 – нет события;<br>1 – есть событие. |
| 8 | PCM | Флаг события успешного сравнения значения QPOSCNT счетчика таймера PCCU с значением сравнения QPOSCMP (QPOSCNT = QPOSCMP):<br>0 – нет события;<br>1 – есть событие. |
| 7 | PCR | Флаг готовности к отложенной записи значения сравнения из регистра QPOSCMP компаратора положения:<br>0 – нет события;<br>1 – есть событие. |
| 6 | PCO | Флаг события переполнения счетчика положения PCCU (QPOSCNT ≥ QPOSMAX) при счете вверх:<br>0 – нет события;<br>1 – есть событие. |
| 5 | PCU | Флаг события переполнения счетчика положения PCCU (QPOSCNT = 0) при счете вниз:<br>0 – нет события;<br>1 – есть событие. |
| 4 | WTO | Флаг достижения счетчиком QWDTMR сторожевого таймера QWDOG значения периода QWDPRD:<br>0 – нет события;<br>1 – есть событие. |
| 3 | QDC | Флаг события смены направления движения счетчика положения PCCU:<br>0 – нет события;<br>1 – есть событие. |
| 2 | PHE | Флаг события ошибки фазы входных сигналов квадратурного декодера QDU:<br>0 – нет события;<br>1 – есть событие. |
| 1 | PCE | Флаг события ошибки значения счетчика положения PCCU при сбросе по событию полного оборота:<br>0 – нет события;<br>1 – есть событие. |
| 0 | INT | Флаг активного запроса прерывания:<br>0 – нет события;<br>1 – есть событие. |

### 20.30.18 *QCLR*

Таблица 523 – Регистр QCLR

| Номер | 31…16 |
|---|---|
| Доступ | WO |
| Сброс | 0 |
| | - |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | - | | | | UTO | IEL | SEL | PCM | PCR | PCO | PCU | WTO | QDC | PHE | PCE | INT |

Таблица 524 – Описание бит регистра QCLR

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:12] | - | Зарезервировано. |
| 11 | UTO | Сброс флага события достижения счетчиком QUTMR таймера временных отсчетов UTIME значения периода QUPRD:<br>0 – нет воздействия;<br>1 – сброс флага. |
| 10 | IEL | Сброс флага события сохранения значения QPOSCNT счетчика таймера PCCU в регистр QPOSILAT по событию полного оборота:<br>0 – нет воздействия;<br>1 – сброс флага. |
| 9 | SEL | Сброс флага события сохранения значения QPOSCNT счетчика таймера PCCU в регистр QPOSSLAT по фронту сигнала стробирования QS:<br>0 – нет воздействия;<br>1 – сброс флага. |
| 8 | PCM | Сброс флага события успешного сравнения значения QPOSCNT счетчика таймера PCCU с значением сравнения QPOSCMP (QPOSCNT = QPOSCMP):<br>0 – нет воздействия;<br>1 – сброс флага. |
| 7 | PCR | Сброс флага готовности к отложенной записи значения сравнения из регистра QPOSCMP компаратора положения:<br>0 – нет воздействия;<br>1 – сброс флага. |
| 6 | PCO | Сброс флага события переполнения счетчика положения PCCU (QPOSCNT ≥ QPOSMAX) при счете вверх:<br>0 – нет воздействия;<br>1 – сброс флага. |
| 5 | PCU | Сброс флага события переполнения счетчика положения PCCU (QPOSCNT = 0) при счете вниз:<br>0 – нет воздействия;<br>1 – сброс флага. |
| 4 | WTO | Сброс флага достижения счетчиком QWDTMR сторожевого таймера QWDOG значения периода QWDPRD:<br>0 – нет воздействия;<br>1 – сброс флага. |
| 3 | QDC | Сброс флага события смены направления движения счетчика положения PCCU:<br>0 – нет воздействия;<br>1 – сброс флага. |
| 2 | PHE | Сброс флага события ошибки фазы входных сигналов квадратурного декодера QDU:<br>0 – нет воздействия;<br>1 – сброс флага. |
| 1 | PCE | Сброс флага события ошибки значения счетчика положения PCCU при сбросе по событию полного оборота:<br>0 – нет воздействия;<br>1 – сброс флага. |
| 0 | INT | Сброс флага активного запроса прерывания:<br>0 – нет воздействия;<br>1 – сброс флага. |

Примечание – Для организации надежного сброса статусных флагов прерываний необходимо учитывать требования, изложенные в пункте 22.1.2 «Обработчики исключений».

### 20.30.19 *QFRC*

Таблица 525 – Регистр QFRC

| Номер | 31…16 |
|---|---|
| Доступ | WO |
| Сброс | 0 |
| | - |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO | WO |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | - | | | | UTO | IEL | SEL | PCM | PCR | PCO | PCU | WTO | QDC | PHE | PCE | - |

Таблица 526 – Описание бит регистра QFRC

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:12] | - | Зарезервировано. |
| 11 | UTO | Установка флага события достижения счетчиком QUTMR таймера временных отсчетов UTIME значения периода QUPRD:<br>0 – нет воздействия;<br>1 – установка флага. |
| 10 | IEL | Установка флага события сохранения значения QPOSCNT счетчика таймера PCCU в регистр QPOSILAT по событию полного оборота:<br>0 – нет воздействия;<br>1 – установка флага. |
| 9 | SEL | Установка флага события сохранения значения QPOSCNT счетчика таймера PCCU в регистр QPOSSLAT по фронту сигнала стробирования QS:<br>0 – нет воздействия;<br>1 – установка флага. |
| 8 | PCM | Установка флага события успешного сравнения значения QPOSCNT счетчика таймера PCCU с значением сравнения QPOSCMP (QPOSCNT = QPOSCMP):<br>0 – нет воздействия;<br>1 – установка флага. |
| 7 | PCR | Установка флага готовности к отложенной записи значения сравнения из регистра QPOSCMP компаратора положения:<br>0 – нет воздействия;<br>1 – установка флага. |
| 6 | PCO | Установка флага события переполнения счетчика положения PCCU (QPOSCNT ≥ QPOSMAX) при счете вверх:<br>0 – нет воздействия;<br>1 – установка флага. |
| 5 | PCU | Установка флага события переполнения счетчика положения PCCU (QPOSCNT = 0) при счете вниз:<br>0 – нет воздействия;<br>1 – установка флага. |
| 4 | WTO | Установка флага достижения счетчиком QWDTMR сторожевого таймера QWDOG значения периода QWDPRD:<br>0 – нет воздействия;<br>1 – установка флага. |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 3 | QDC | Установка флага события смены направления движения счетчика положения PCCU:<br>0 – нет воздействия;<br>1 – установка флага. |
| 2 | PHE | Установка флага события ошибки фазы входных сигналов квадратурного декодера QDU:<br>0 – нет воздействия;<br>1 – установка флага. |
| 1 | PCE | Установка флага события ошибки значения счетчика положения PCCU при сбросе по событию полного оборота:<br>0 – нет воздействия;<br>1 – установка флага. |
| 0 | - | Зарезервировано. |

### 20.30.20 *QEPSTS*

Таблица 527 – Регистр QEPSTS

| Номер | 31…16 |
|---|---|
| Доступ | RO |
| Сброс | 0 |
| | - |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | RO | RO | RO | RO | RO | RO | RO | RO | R/W | RO | RO | RO | R/W | R/W | R/W | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | - | | | | | | | | UPEVNT | FIDF | QDF | QDLF | COEFF | CDEF | FIMF | PCEF |

Таблица 528 – Описание бит регистра QEPSTS

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:8] | - | Зарезервировано. |
| 7 | UPEVNT | Состояние флага захвата нового значения счетчика захвата фронтов в регистр QCPRD:<br>0 – нет события;<br>1 – есть событие.<br>Сбрасывается записью 1. |
| 6 | FIDF | Направление вращения по первому маркеру индексации:<br>0 – прямое направление вращения;<br>1 – обратное направление вращения. |
| 5 | QDF | Направление вращения в текущий момент времени:<br>0 – прямое направление вращения;<br>1 – обратное направление вращения. |
| 4 | QDLF | Направление движения по каждому маркеру индексации:<br>0 – прямое направление вращения;<br>1 – обратное направление вращения. |
| 3 | COEF | Состояние флага переполнения счетчика захвата фронтов:<br>0 – нет переполнения;<br>1 – есть переполнение.<br>Сбрасывается записью 1. |
| 2 | CDEF | Состояние флага изменения направления вращения при захвате фронтов:<br>0 – без изменения направления;<br>1 – ошибка смены направления.<br>Сбрасывается записью 1. |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 1 | FIMF | Состояние флаг захвата первого индексного маркера:<br>0 – нет события;<br>1 – захвачен первый индексный маркер.<br>Сбрасывается записью 1. |
| 0 | PCEF | Состояние флага ошибки значения счетчика положения PCCU при сбросе по событию полного оборота, обновляемый по каждому событию полного оборота:<br>0 – нет ошибки;<br>1 – есть ошибка. |

### 20.30.21 *QCTMR, QCTMR1, QCTMR2, QCTMR3, QCTMR4*

Таблица 529 – Регистры QCTMR, QCTMR1, QCTMR2, QCTMR3, QCTMR4

| Номер | 31…16 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | - |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | QCTMR | | | | | | | | | | | | | | | |

Таблица 530 – Описание бит регистров QCTMR, QCTMR1, QCTMR2, QCTMR3, QCTMR4

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:16] | - | Зарезервировано. |
| [15:0] | QCTMR | Счетчик захвата времени. |

### 20.30.22 *QCPRD*

Таблица 531 – Регистр QCPRD

| Номер | 31…16 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | - |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | QCPRD | | | | | | | | | | | | | | | |

Таблица 532 – Описание бит регистра QCPRD

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:16] | - | Зарезервировано. |

| | | |
|---|---|---|
| [15:0] | QCPRD | Сохраненное значение периода таймера захвата. Обновляется по каждому сигналу QCLK от блока квадратурного декодера. |

### 20.30.23 *QCTMRLAT*

Таблица 533 – Регистр QCTMRLAT

| Номер | 31…16 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | - |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | QCTMRLAT | | | | | | | | | | | | | | | |

Таблица 534 – Описание бит регистра QCTMRLAT

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:16] | - | Зарезервировано. |
| [15:0] | QCTMRLAT | Сохраненное значение регистра QCTMR. Обновляется по каждому событию чтения счетчика POSCNT процессором или окончанию счета таймера временных отсчетов. |

### 20.30.24 *QCPRDLAT*

Таблица 535 – Регистр QCPRDLAT

| Номер | 31…16 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | - |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | QCPRDLAT | | | | | | | | | | | | | | | |

Таблица 536 – Описание бит регистра QCPRDLAT

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| [31:16] | - | Зарезервировано. |
| [15:0] | QCPRDLAT | Сохраненное значение регистра QCPRD. Обновляется по каждому событию чтения счетчика POSCNT процессором или окончанию счета таймера временных отсчетов |

## 20.31 Описание регистров контроллеров захвата CAP_CNTR

Таблица 537 – Регистры контроллеров захвата

| Базовый Адрес | Название | Описание |
|---|---|---|
| 0x4009_8000 | CAP0_CNTR | Контроллер CAP0 |
| 0x4009_9000 | CAP1_CNTR | Контроллер CAP1 |
| 0x4009_A000 | CAP2_CNTR | Контроллер CAP2 |
| 0x4009_B000 | CAP3_CNTR | Контроллер CAP3 |
| **Смещение** | | |
| 0x00 | TSCTR | Регистр счетчика захвата времени |
| 0x04 | CTRPHS | Регистр значения смещения фазы счетчика |
| 0x08 | CAP1 | Регистр захвата 1 |
| 0x0C | CAP2 | Регистр захвата 2 |
| 0x10 | CAP3 | Регистр захвата 3 |
| 0x14 | CAP4 | Регистр захвата 4 |
| 0x18 | ECCTL1 | Регистр контроля захвата 1 |
| 0x1C | ECCTL2 | Регистр контроля захвата 2 |
| 0x20 | ECEINT | Регистр маски источников прерывания |
| 0x24 | ECFLG | Регистр статуса источников прерывания |
| 0x28 | ECCLR | Регистр сброса источников прерывания |
| 0x2C | ECFRC | Регистр установки источников прерывания |

### 20.31.1 *TSCTR*

Таблица 538 – Регистр TSCTR

| Номер | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | 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 | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | TSCTR | | | | | | | | | | | | | | | |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | TSCTR | | | | | | | | | | | | | | | |

Таблица 539 – Описание бит регистра TBPRD

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31..0 | TSCTR | Основной 32 разрядный счетчик |

### 20.31.2 *CTRPHS*

Таблица 540 – Регистр CTRPHS

| Номер | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | 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 | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | |
|---|---|
| | CTRPHS |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | CTRPHS | | | | | | | | | | | | | | | |

Таблица 541 – Описание бит регистра CTRPHS

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31..0 | CTRPHS | Значение сдвига значения счетчика относительно входного импульса синхронизации. Значение этого регистра записывается в регистр основного счетчика CTRPHS по событию внешней или программной синхронизацией |

### 20.31.3 *CAP1*

Таблица 542 – Регистр CAP1

| Номер | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | 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 | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | CAP1 | | | | | | | | | | | | | | | |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | CAP1 | | | | | | | | | | | | | | | |

Таблица 543 – Описание бит регистра CAP1

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31..0 | CAP1 | Регистр загружается записью в регистр, значением таймера TSCTR по событию CEVT1 в режиме захвата, значением регистра CAP3 в режиме ШИМ (APWM). В режиме ШИМ задает периода PRD таймера TSCTR |

### 20.31.4 *CAP2*

Таблица 544 – Регистр CAP2

| Номер | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | 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 | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | CAP2 | | | | | | | | | | | | | | | |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | CAP2 | | | | | | | | | | | | | | | |

Таблица 545 – Описание бит регистра CAP2

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31..0 | CAP2 | Регистр загружается записью в регистр, значением таймера TSCTR по событию CEVT2 в режиме захвата, значением регистра CAP4 в режиме ШИМ (APWM). В режиме ШИМ задает значение сравнения CMP таймера TSCTR |

### 20.31.5 *CAP3*

Таблица 546 – Регистр CAP3

| Номер | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | 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 | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | CAP3 | | | | | | | | | | | | | | | |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | CAP3 | | | | | | | | | | | | | | | |

Таблица 547 – Описание бит регистра CAP3

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31..0 | CAP3 | Регистр загружается записью в регистр и значением таймера TSCTR по событию CEVT3 в режиме захвата. В режиме ШИМ (APWM) используется как регистр отложенной загрузки значения периода CAP1 |

### 20.31.6 *CAP4*

Таблица 548 – Регистр CAP4

| Номер | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | 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 | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | CAP4 | | | | | | | | | | | | | | | |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | CAP4 | | | | | | | | | | | | | | | |

Таблица 549 – Описание бит регистра CAP4

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31..0 | CAP4 | Регистр загружается записью в регистр и значением таймера TSCTR по событию CEVT4 в режиме захвата. В режиме ШИМ (APWM) используется как регистр отложенной загрузки значения сравнения CAP2 |

### 20.31.7 *ECCTL1*

Таблица 550 – Регистр ECCTL1

| Номер | 31…18 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | Зарезервировано |

| Номер | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | SYNCO_CS | SYNCI_CS | Зарезервировано | FREESCALE | | | | | | CAPLDEN | CTRRST4 | CAP4POL | CTRRST3 | CAP3POL | CTRRST2 | CAP2POL | CTRRST1 | CAP1POL |

Таблица 551 – Описание бит регистра ECCTL1

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31...18 | Зарезервировано | Зарезервировано |
| 17 | SYNCO_CS | Выбор синхронизации для сигнала SYNCO:<br>0 – Сигнал работает на частоте блока;<br>1 – Сигнал пересинхронизируется на частоту PCLK |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 16 | SYNCI_CS | Выбор синхронизации для сигнала SYNCI:<br>0 – Сигнал используется со входа блока;<br>1 – Сигнал пересинхронизируется на частоту PCLK |
| 15..14 | Зарезервировано | Зарезервировано |
| 13..9 | PRESCALE | Делитель частоты входного сигнала:<br>00000 Деление на 1;<br>00001 Деление на 2;<br>00010 Деление на 4;<br>00011 Деление на 6;<br>……<br>11110 Деление на 60;<br>11111 Деление на 64 |
| 8 | CAPLDEN | Разрешение загрузки регистров захвата CAP 1-4 по событию захвата.<br>0 – Запрещен захват значений по событиям захвата;<br>1 – Разрешен захват значений по событиям захвата |
| 7 | CTRRST4 | Сброс счетчика по событию захвата 4:<br>0 – Запрещен сброс счетчика по событию захвата 4;<br>1 – Разрешен сброс счетчика по событию захвата 4 |
| 6 | CAP4POL | Выбор полярности сигнала события захвата 4:<br>0 – Захват происходит по фронту сигнала;<br>1 – Захват происходит по срезу сигнала |
| 5 | CTRRST3 | Сброс счетчика по событию захвата 3:<br>0 – Запрещен сброс счетчика по событию захвата 3;<br>1 – Разрешен сброс счетчика по событию захвата 3 |
| 4 | CAP3POL | Выбор полярности сигнала события захвата 3:<br>0 – Захват происходит по фронту сигнала;<br>1 – Захват происходит по срезу сигнала |
| 3 | CTRRST2 | Сброс счетчика по событию захвата 2:<br>0 – Запрещен сброс счетчика по событию захвата 2;<br>1 – Разрешен сброс счетчика по событию захвата 2 |
| 2 | CAP2POL | Выбор полярности сигнала события захвата 2:<br>0 – Захват происходит по фронту сигнала;<br>1 – Захват происходит по срезу сигнала |
| 1 | CTRRST1 | Сброс счетчика по событию захвата 1:<br>0 – Запрещен сброс счетчика по событию захвата 1;<br>1 – Разрешен сброс счетчика по событию захвата 1 |
| 0 | CAP1POL | Выбор полярности сигнала события захвата 1:<br>0 – Захват происходит по фронту сигнала;<br>1 – Захват происходит по срезу сигнала |

### 20.31.8 *ECCTL2*

Таблица 552 – Регистр ECCTL2

| Номер | 31…28 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | Зарезервировано |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 1 | 0 | 0 | 0 | 0 |
| | Зарезервировано | | | | | APWMPOL | CAP/APWM | SWSYNC | SYNCO_SEL | | SYNCI_EN | TSCTRSTOP | REARM | STOP_WRAP | | CONT/ONESHT |

Таблица 553 – Описание бит регистра ECCTL2

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31...11 | Зарезервировано | Зарезервировано |
| 10 | APWMPOL | Выбор полярности сигнала APWM. Доступен в режиме APWM.<br>0 – Активный уровень высокий;<br>1 – Активный уровень низкий |
| 9 | CAP/APWM | Выбор режима работы Захват/ШИМ (CAP/APWM):<br>0 – Блок функционирует в режиме захвата. В этом режиме:<br>Запрещен сброс регистра основного счета TSCTR по событию CTR = PRD;<br>Запрещена загрузка регистров CAP1 и CAP2 дублерами;<br>Возможность загрузки регистров CAP1-4;<br>1 – Блок функционирует в режиме ШИМ |
| 8 | SWSYNC | Программное управление синхронизацией основного счетчика (TSCTR). Этот бит обеспечивает программную возможность синхронизации<br>0 – Запись нуля не влияет на систему. Читается всегда как 0;<br>1 – Запись единицы приводит к загрузке в регистр основного счета таймера TSCTR |
| 7..6 | SYNCO_SEL | Выбор источника выходного сигнала синхронизации<br>00 – событие выходной синхронизации равно событию входной синхронизации;<br>01 – генерация события синхронизации по событию CTR = PRD;<br>10 – на событие выходной синхронизации транслируется вход блока захвата;<br>11 – запрещение выходного сигнала синхронизации |
| 5 | SYNCI_EN | Режим синхронизации основного счетчика TSCTR:<br>0 – Запрет синхронизации основного счетчика по событиям синхронизации;<br>1 –Разрешение загрузки регистра TSCTR значением CTRPHS по событиям внешней или программной синхронизации |
| 4 | TSCTRSTOP | Остановка работы таймера<br>0 – Работает.<br>1 – Остановлен |

| | | |
|---|---|---|
| 3 | REARM | Бит перезапуска преобразований в одиночном режиме<br>0 – Не влияет на работу модуля. Читается всегда как 0;<br>1 – Запускает серию захватов после остановки в одиночном режиме |
| 2..1 | STOP_WRAP | Значение счетчика события, на котором происходит остановка захвата для одиночного режима.<br>00 – Соответствует регистру захвата CAP1. Остановка будет выполнена при записи результата захвата в регистр CAP1.<br>01 – Соответствует регистру захвата CAP2.<br>10 – Соответствует регистру захвата CAP3.<br>11 – Соответствует регистру захвата CAP4 |
| 0 | CONT/ONESHT | Непрерывный или одиночный режим работы<br>0 – непрерывный режим;<br>1 – одиночный режим |

### 20.31.9 *ECEINT*

Таблица 554 – Регистр ECEINT

| Номер | 31…28 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | Зарезервировано |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | Зарезервировано | | | | | | | | CTR=CMP | CTR=PRD | CTROVF | CEVT4 | CEVT3 | CEVT2 | CEVT1 | Зарезервировано |

Таблица 555 – Описание бит регистра ECEINT

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31...8 | Зарезервировано | Зарезервировано |
| 7 | CTR=CMP | Разрешение прерывания по событию CTR=CMP:<br>0 – прерывание запрещено;<br>1 – прерывание разрешено |
| 6 | CTR=PRD | Разрешение прерывания по событию CTR=PRD:<br>0 – прерывание запрещено;<br>1 – прерывание разрешено |
| 5 | CTROVF | Разрешение прерывания по событию CTROVF:<br>0 – прерывание запрещено;<br>1 – прерывание разрешено |
| 4 | CEVT4 | Разрешение прерывания по событию CEVT4:<br>0 – прерывание запрещено;<br>1 – прерывание разрешено |
| 3 | CEVT3 | Разрешение прерывания по событию CEVT3:<br>0 – прерывание запрещено;<br>1 – прерывание разрешено |

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 2 | CEVT2 | Разрешение прерывания по событию CEVT2:<br>0 – прерывание запрещено;<br>1 – прерывание разрешено |
| 1 | CEVT1 | Разрешение прерывания по событию CEVT1:<br>0 – прерывание запрещено;<br>1 – прерывание разрешено |
| 0 | Зарезервировано | Зарезервировано |

### 20.31.10 *ECFLG*

Таблица 556 – Регистр ECFLG

| Номер | 31…28 |
|---|---|
| Доступ | RO |
| Сброс | 0 |
| | Зарезервировано |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO | RO |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | Зарезервировано | | | | | | | | CTR=CMP | CTR=PRD | CTROVF | CEVT4 | CEVT3 | CEVT2 | CEVT1 | Зарезервировано |

Таблица 557 – Описание бит регистра ECFLG

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31...8 | Зарезервировано | Зарезервировано |
| 7 | CTR=CMP | Флаг события прерывания по событию CTR=CMP:<br>0 – событие не произошло;<br>1 – событие произошло |
| 6 | CTR=PRD | Флаг события прерывания по событию CTR=PRD:<br>0 – событие не произошло;<br>1 – событие произошло |
| 5 | CTROVF | Флаг события прерывания по событию CTROVF:<br>0 – событие не произошло;<br>1 – событие произошло |
| 4 | CEVT4 | Флаг события прерывания по событию CEVT4:<br>0 – событие не произошло;<br>1 – событие произошло |
| 3 | CEVT3 | Флаг события прерывания по событию CEVT3:<br>0 – событие не произошло;<br>1 – событие произошло |
| 2 | CEVT2 | Флаг события прерывания по событию CEVT2:<br>0 – событие не произошло;<br>1 – событие произошло |
| 1 | CEVT1 | Флаг события прерывания по событию CEVT1:<br>0 – событие не произошло;<br>1 – событие произошло |
| 0 | Зарезервировано | Зарезервировано |

### 20.31.11 *ECCLR*

Таблица 558 – Регистр ECCLR

| Номер | 31…28 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | Зарезервировано |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | Зарезервировано | | | | | | | | CTR=CMP | CTR=PRD | CTROVF | CEVT4 | CEVT3 | CEVT2 | CEVT1 | Зарезервировано |

Таблица 559 – Описание бит регистра ECCLR

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31...8 | Зарезервировано | Зарезервировано |
| 7 | CTR=CMP | Очистка прерывания по событию CTR=CMP:<br>0 – не влияет на состояние события;<br>1 – очистка события прерывания |
| 6 | CTR=PRD | Очистка прерывания по событию CTR=PRD:<br>0 – не влияет на состояние события;<br>1 – очистка события прерывания |
| 5 | CTROVF | Очистка прерывания по событию CTROVF:<br>0 – не влияет на состояние события;<br>1 – очистка события прерывания |
| 4 | CEVT4 | Очистка прерывания по событию CEVT4:<br>0 – не влияет на состояние события;<br>1 – очистка события прерывания |
| 3 | CEVT3 | Очистка прерывания по событию CEVT3:<br>0 – не влияет на состояние события;<br>1 – очистка события прерывания |
| 2 | CEVT2 | Очистка прерывания по событию CEVT2:<br>0 – не влияет на состояние события;<br>1 – очистка события прерывания |
| 1 | CEVT1 | Очистка прерывания по событию CEVT1:<br>0 – не влияет на состояние события;<br>1 – очистка события прерывания |
| 0 | Зарезервировано | Зарезервировано |

Примечание – Для организации надежного сброса статусных флагов прерываний необходимо учитывать требования, изложенные в пункте 22.1.2 «Обработчики исключений».

### 20.31.12 *ECFRC*

Таблица 560 – Регистр ECFRC

| Номер | 31…28 |
|---|---|
| Доступ | R/W |
| Сброс | 0 |
| | Зарезервировано |

| Номер | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Доступ | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | 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 | 0 | 0 | 0 | 0 | 0 |
| | Зарезервировано | | | | | | | | CTR=CMP | CTR=PRD | CTROVF | CEVT4 | CEVT3 | CEVT2 | CEVT1 | Зарезервировано |

Таблица 561 – Описание бит регистра ECFRC

| Номер бита | Функциональное имя бита | Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений |
|---|---|---|
| 31...8 | Зарезервировано | Зарезервировано |
| 7 | CTR=CMP | Программная установка прерывания по событию CTR=CMP:<br>0 – не влияет на состояние события;<br>1 – установка события |
| 6 | CTR=PRD | Программная установка прерывания по событию CTR=PRD:<br>0 – не влияет на состояние события;<br>1 – установка события |
| 5 | CTROVF | Программная установка прерывания по событию CTROVF:<br>0 – не влияет на состояние события;<br>1 – установка события |
| 4 | CEVT4 | Программная установка прерывания по событию CEVT4:<br>0 – не влияет на состояние события;<br>1 – установка события |
| 3 | CEVT3 | Программная установка прерывания по событию CEVT3:<br>0 – не влияет на состояние события;<br>1 – установка события |
| 2 | CEVT2 | Программная установка прерывания по событию CEVT2:<br>0 – не влияет на состояние события;<br>1 – установка события |
| 1 | CEVT1 | Программная установка прерывания по событию CEVT1:<br>0 – не влияет на состояние события;<br>1 – установка события |
| 0 | Зарезервировано | Зарезервировано |

## 20.32 Описание регистров контроллера тригонометрических преобразований CORDIC_CNTR

### 20.32.1 Регистр управления модулем CRD_CNTR

(смещение: 0x00)
Начальное значение: 0x10100000

Таблица 562 – Регистр CRD_CNTR

| Номер | 31…29 | 28…24 | 23…21 | 20…16 |
|---|---|---|---|---|
| Доступ | R/O | R/W | R/O | R/W |
| Сброс | 0 | 0 | 0 | 0 |
| | Зарезервировано | OUT_LVL_VAL | Зарезервировано | IN_LVL_VAL |

| Номер | 15…9 | 8 | 7 | 6…4 | 3…1 | 0 |
|---|---|---|---|---|---|---|
| Доступ | R/O | R/W | R/O | R/W | R/O | R/W |
| Сброс | 0 | 0 | 0 | 0 | 0 | 0 |
| | Зарезервировано | CRD_EN | Зарезервировано | IN_FORMAT | Зарезервировано | IN_MODE |

Таблица 563 – Описание бит регистра CRD_CNTR

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 0 | IN_MODE | Режим поворота:<br>0 – Поворот входного вектора (x; y) на угол α. На выходе получается новый, повернутый вектор (x; y), угол α обнуляется<br>1 – Поворот входного вектора (x; y) до оси OX. На выходе получается исходный угол вектора α, и длина исходного вектора в координате x (деформированное значение) | RW |
| 3:1 | - | *Зарезервировано* | RO |
| 6:4 | IN_FORMAT | Формат рабочих данных:<br>000 – 32 бита, плавающая точка<br>001 – 16 бит, плавающая точка<br>010 – 32 бита, целочисленный режим<br>Остальные значение – *зарезервировано* | RW |
| 7 | - | *Зарезервировано* | RO |
| 8 | CRD_EN | Включение модуля:<br>1 – включить модуль<br>0 – выключить модуль<br>Включение контролируется по биту CRD_RDY *(см. статусный регистр CRD_STATUS)* | RW |
| 15:9 | - | *Зарезервировано* | RO |

| | | | |
|---|---|---|---|
| 20:16 | IN_LVL_VAL | Уровень заполнения входного FIFO для выработки сигнала IN_FIFO_LVL *(см. статусный регистр CRD_STATUS)*. Диапазон значение [0; 31] | RW |
| 23:21 | - | *Зарезервировано* | RO |
| 28:24 | OUT_LVL_VAL | Уровень заполнения выходного FIFO для выработки сигнала OUT_FIFO_LVL *(см. статусный регистр CRD_STATUS)*. Диапазон значение [0; 31] | RW |
| 31:29 | - | *Зарезервировано* | RO |

### 20.32.2 Регистр статуса модуля CRD_STATUS

(смещение: 0x01)

Таблица 564 – Описание бит регистра *CRD_STATUS*

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 0 | CRD_RDY | Готовность модуля к работе:<br>0 – Модуль не готов к работе, выключен<br>1 – Модуль готов к работе, включен<br>Включение модуля осуществляется битом CRD_EN *(см. контрольный регистр CRD_CNTR)* | RO |
| 1 | IN_PRGR | Флаг наличия данных в обработке, данный флаг равен 1 когда во входном FIFO есть данные или данные обрабатываются в ядре модуля. Как только данные будут обработаны и входное FIFO опустеет флаг становится 0 | RO |
| 2 | OUT_NCLR | Флаг наличия не забранных данных, данный флаг равен 1 когда данные обрабатываются в ядре модуля или выходное FIFO не пусто, по окончанию обработки и выборке всех данных из выходного FIFO флаг становится 0 | RO |
| 3 | - | *Зарезервировано* | RO |
| 4 | IN_FIFO_EMPTY | Входное FIFO пусто, данный флаг находиться в 1, когда в FIFO нет данных | RO |
| 5 | IN_FIFO_FULL | Входное FIFO заполнено, данный флаг находится в 1, когда FIFO полностью заполнено, записываемые данные при выставленном флаге отбрасываются, при попытке записи выставляется флаг IN_FIFO_OVR | RO |
| 6 | IN_FIFO_LVL | Во входном FIFO данных больше заданного количества, данный флаг выставляется в 1 когда количество данных сохраненных в FIFO больше IN_LVL_*VAL (см. контрольный регистр CRD_CNTR)* | RO |
| 7 | IN_FIFO_OVR | Входное FIFO переполнено. Данный флаг выставляется в 1 при попытке записи данных в заполненное FIFO, когда выставлен флаг IN_FIFO_FULL. Данные записываемые в этот момент отбрасываются, и данный флаг выставляется в 1. Флаг снимается при удачной записи в FIFO, то есть при очередной записи в не заполненное FIFO | RO |
| 8 | OUT_FIFO_EMPTY | Выходное FIFO пусто, данный флаг находиться в 1, когда в FIFO нет данных | RO |
| 9 | OUT_FIFO_FULL | Выходное FIFO заполнено, данный флаг находится в 1, когда FIFO полностью заполнено | RO |
| 10 | OUT_FIFO_LVL | В выходном FIFO данных больше заданного количества, данный флаг выставляется в 1 когда количество данных сохраненных в FIFO больше OUT_LVL_VAL *(см. контрольный регистр CRD_CNTR)* | RO |
| 11 | OUT_FIFO_OVR | Попытка чтения из пустого FIFO. Данный флаг выставляется в 1 при попытке чтения данных из пустого FIFO, когда выставлен флаг OUT_FIFO_EMPTY. Из FIFO будут считаны прошлые данные, а флаг будет выставлен в 1. Флаг снимается при удачном чтении из FIFO, то есть при очередном чтении из не пустого FIFO | RO |
| 31:12 | - | *Зарезервировано* | RO |

### 20.32.3 Регистр флагов прерываний CRD_INTF

Смещение: 0x02
Запись 0 не меняет значение бит в данном регистре

Таблица 565 – Описание бит регистра *CRD_INTF*

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 0 | IN_FIFO_EMPTY_IF | Входное FIFO пусто, флаг отражает состояние бита IN_FIFO_EMPTY *(см. статусный регистр CRD_STATUS)*, ставиться и снимается с появление и снятием бита IN_FIFO_EMPTY | RO |
| 1 | IN_FIFO_NLVL_IF | Входное FIFO заполнено не выше указанного уровня, флаг отражает инверсию состояния бита IN_FIFO_LVL *(см. статусный регистр CRD_STATUS)*. Флаг в 1 когда IN_FIFO_LVL в 0 и наоборот | RO |
| 2 | IN_FIFO_NFULL_IF | Входное FIFO заполнено не полностью, флаг отражает инверсию состояния бита IN_FIFO_FULL *(см. статусный регистр CRD_STATUS)*. Флаг в 1 когда IN_FIFO_FULL в 0 и наоборот | RO |
| 3 | IN_FIFO_OVR_IF | Входное FIFO переполнено, флаг выставляется при возникновении данной ситуации, при появлении бита IN_FIFO_OVR *(см. статусный регистр CRD_STATUS),* флаг снимается записью 1 в данный бит | R/W1 |
| 4 | OUT_FIFO_NEMPTY_IF | Выходное FIFO не пусто, флаг отражает инверсию состояния бита OUT_FIFO_EMPTY *(см. статусный регистр CRD_STATUS)*. Флаг в 1 когда OUT_FIFO_EMPTY в 0 и наоборот | RO |
| 5 | OUT_FIFO_LVL_IF | Выходное FIFO заполнено выше заданного уровня, флаг отражает состояние бита OUT_FIFO_LVL *(см. статусный регистр CRD_STATUS)*, ставиться и снимается с появление и снятием бита OUT_FIFO_LVL | RO |
| 6 | OUT_FIFO_FULL_IF | Выходное FIFO заполнено полностью, флаг отражает состояние бита OUT_FIFO_FULL *(см. статусный регистр CRD_STATUS)*, ставиться и снимается с появление и снятием бита OUT_FIFO_FULL | RO |
| 7 | OUT_FIFO_OVR_IF | Произведено чтение из пустого FIFO, флаг выставляется при возникновении данной ситуации, при появлении бита OUT_FIFO_OVR *(см. статусный регистр CRD_STATUS)*, флаг снимается записью 1 в данный бит | R/W1 |
| 8 | PROC_CMPL_IF | Обработка данных завершена, флаг выставляется по возникновению данной ситуации, при снятии бита IN_PRGR *(см. статусный регистр CRD_STATUS)*, флаг снимается записью 1 в данный бит | R/W1 |
| 31:9 | - | *Зарезервировано* | RO |

### 20.32.4 Регистр разрешения прерываний CRD_INTE

(смещение: 0x03)

Начальное значение: 0x00000000

Таблица 566 – Описание бит регистра *CRD_INTE*

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 0 | IN_FIFO_EMPTY_IE | Биты разрешения прерывания, если бит данного регистра в 1 и выставлен соответствующий флаг регистр флагов прерываний CRD_INTF, модуль генерирует запрос прерывания. *(Описание флагов прерываний см. CRD_INTF)* | RW |
| 1 | IN_FIFO_NLVL_IE | | RW |
| 2 | IN_FIFO_NFULL_IE | | RW |
| 3 | IN_FIFO_OVR_IE | | RW |
| 4 | OUT_FIFO_NEMPTY_IE | | RW |
| 5 | OUT_FIFO_LVL_IEN | | RW |
| 6 | OUT_FIFO_FULL_IE | | RW |
| 7 | OUT_FIFO_OVR_IE | | RW |
| 8 | PROC_CMPL_IE | | RW |
| 31:9 | - | *Зарезервировано* | RO |

### 20.32.5 Регистр записи координаты X вектора IN_X

*(смещение: 0x04)*

Начальное значение: 0x00000000

Таблица 567 – Описание бит регистра

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 31:0 | X | Значение координаты X вектора для поворота, интерпретация данных зависит от выбранного формата рабочих данных *(см. контрольный регистр CRD_CNTR).* Данные записанные в этот регистр передаются во входное FIFO после записи в регистр значения угла *(см. регистр записи значения угла α IN_A)* | RW |

### 20.32.6 *Регистр записи координаты Y вектора IN_Y*

*(смещение: 0x05)*

Начальное значение: 0x00000000

Таблица 568 – Описание бит регистра

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 31:0 | Y | Значение координаты Y вектора для поворота, интерпретация данных зависит от выбранного формата рабочих данных *(см. контрольный регистр CRD_CNTR).* Данные записанные в этот регистр передаются во входное FIFO после записи в регистр значения угла *(см. регистр записи значения угла α IN_A)* | RW |

### 20.32.7 *Регистр записи значения угла α IN_A*

*(смещение: 0x06)*

Начальное значение: 0x00000000

Таблица 569 – Описание бит регистра

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 31:0 | A | Значение угла поворота или начального угла. Интерпретация данных зависит от режима поворота *(см. контрольный регистр CRD_CNTR).* При записи в этот регистр (в старший байт при побайтовом доступе) во входное FIFO поступают данные всех входных регистров IN_X, IN_Y, IN_A | RW |

### 20.32.8 *Регистр последовательной записи IN_SEQ*

*(смещение: 0x07)*

При чтении из данного регистра всегда возвращается 0 значение

Таблица 570 – Описание бит регистра

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 31:0 | SEQ | Входные данные для поворота, запись в данный регистр (в старший байт при побайтовом доступе) производит последовательную запись в регистры IN_X, IN_Y, IN_A. Выбор текущего регистра для записи зависит от адреса записи *(см. регистр выбора адреса записи IN_ADDR)* | WO |

### 20.32.9 Регистр чтения координаты X вектора результата OUT_X

(смещение: 0x08)

Таблица 571 – Описание бит регистра

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 31:0 | X | Значение координаты X вектора после поворота или деформированная длина вектора, интерпретация данных зависит от выбранного формата рабочих данных и режима поворота *(см. контрольный регистр CRD_CNTR).* Данные читаются из выходного FIFO, обновление данных происходит после чтения из регистра значения угла *(см. регистр чтения значения угла α OUT_A)* | RO |

### 20.32.10 Регистр чтения координаты Y вектора результата OUT_Y

*(смещение: 0x09)*

Таблица 572 – Описание бит регистра

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 31:0 | Y | Значение координаты Y вектора после поворота, интерпретация данных зависит от выбранного формата рабочих данных *(см. контрольный регистр CRD_CNTR).* Данные читаются из выходного FIFO, обновление данных происходит после чтения из регистра значения угла *(см. регистр чтения значения угла α OUT_A)* | RO |

### 20.32.11 Регистр чтения значения угла α OUT_A

*(смещение: 0x0A)*

Таблица 573 – Описание бит регистра

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 31:0 | A | Ошибка поворота или значение угла вектора. Интерпретация данных зависит от режима поворота *(см. контрольный регистр CRD_CNTR).* При чтении из этого регистра происходит обновление данных выходного FIFO, для всех выходных регистров OUT_X, OUT_Y, OUT_A | RO |

### 20.32.12 Регистр последовательного чтения OUT_SEQ

*(смещение: 0x0B)*

Таблица 574 – Описание бит регистра

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 31:0 | SEQ | Результат поворота, чтение данного регистра производит последовательное чтение регистров OUT_X, OUT_Y, OUT_A. Выбор текущего регистра для чтения зависит от адреса чтения *(см. регистр выбора адреса чтения OUT_ADDR)* | RO |

### 20.32.13 Регистр выбора адреса записи IN_ ADDR

*(смещение: 0x0C)*

Начальное значение: 0x00000000

Таблица 575 – Описание бит регистра

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 1:0 | ADDR | Адрес регистра данных в который будет произведена очередная запись *(см. регистр последовательной записи IN_SEQ).* После записи адрес автоматически переставляется на следующий регистр, после записи в последний доступный регистр, адрес переставляется на первый.<br>Адреса регистров:<br>– 00 – адрес IN_X<br>– 01 – адрес IN_Y<br>– 10 – адрес IN_A<br>– 11 – запись в этот регистр передает данные во входное FIFO, записанные данные игнорируются | RW |
| 31:2 | - | *Зарезервировано* | RO |

### 20.32.14 Регистр выбора адреса чтения OUT_ ADDR

*(смещение: 0x0D)*

Начальное значение: 0x00000000

Таблица 576 – Описание бит регистра

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 1:0 | ADDR | Адрес регистра данных из которого будет произведено очередное чтение (см. регистр последовательного чтения OUT_SEQ). После чтения адрес автоматически переставляется на следующий регистр, после чтения из последнего доступного регистра, адрес переставляется на первый.<br>Адреса регистров:<br>00 – адрес OUT_X<br>01 – адрес OUT_Y<br>10 – адрес OUT_A<br>11 – чтение из этого регистра обновляет данные выходного FIFO, возвращаемые данные 0 | RW |
| 31:2 | - | Зарезервировано | RO |

## 20.33 Описание регистров контроллера USB USB_CNTR

- Общие USB регистры – эти регистры обеспечивают контроль и статус работы ядра.
- Индексированные регистры Status/Control конечных точек – эти регистры обеспечивают контроль и хранят состояние выбранной в данный момент контрольной точки. Регистры, в этом разделе, зависят от того, находится ли ядро в периферийном режиме (DevCtl.D2 = 0) или в режиме хоста (DevCtl.D2 = 1) и от значения регистра Index.
- FIFO – этот диапазон адресов обеспечивает доступ к FIFO контрольных точек
- OTG, регистры Version и Vendor – эти регистры обеспечивают дополнительный доступ к состоянию и контролю устройства
- Контрольные регистры управления конечными точками – эти регистры предоставляют данные целевой функции и адресы узлов для каждой из точек
- Неиндексированные регистры Control/Status – регистры, доступные независимо от значения регистра Index. Регистры EP0 100h-10Fh; Регистры EP1 110h-11Fh; EP2 120h-12Fh; и так далее.
- Регистры управления DMA – эти регистры появляются только в том случае, если проект синтезирован для включения дополнительного контроллера DMA.

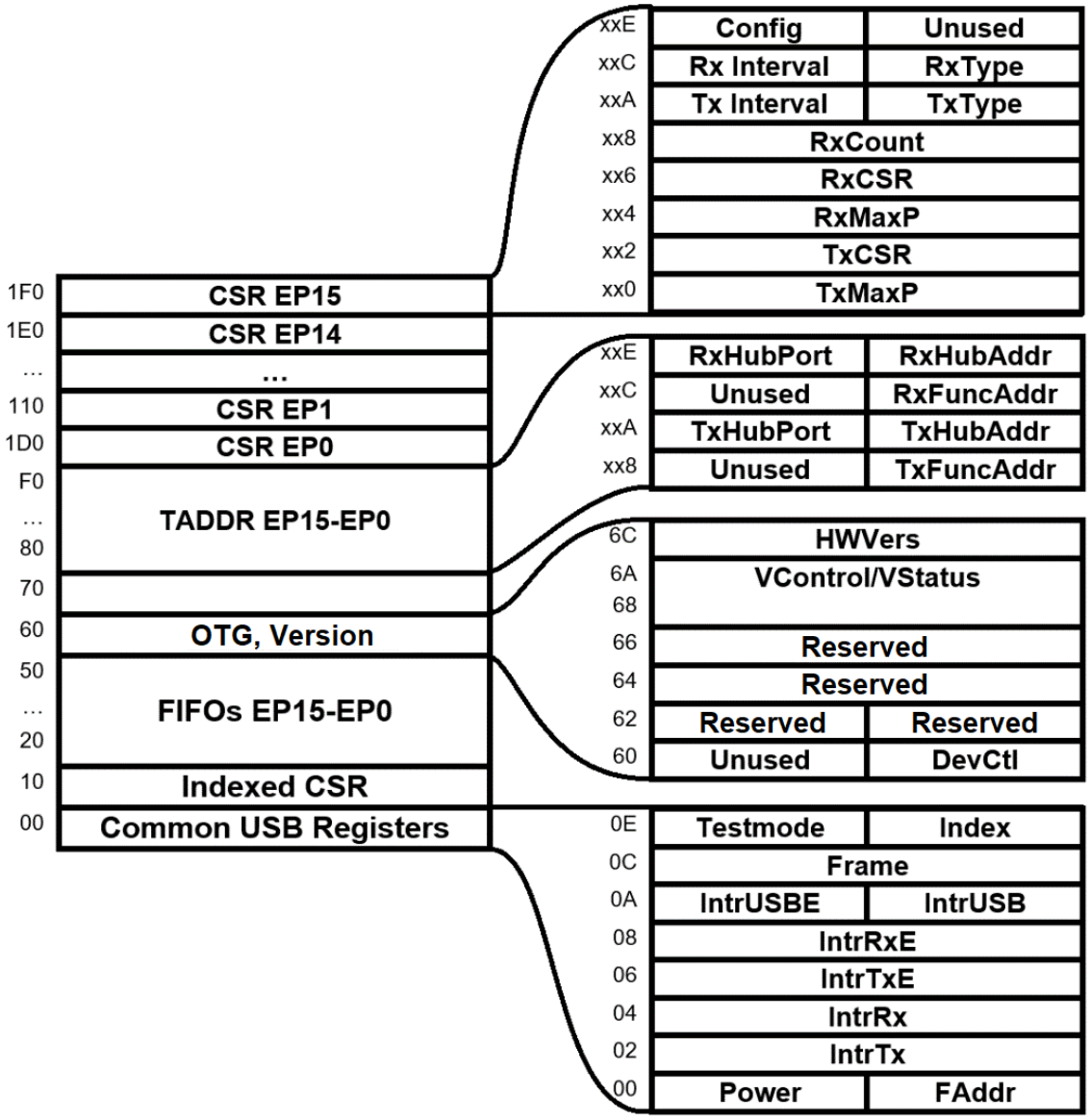



Рисунок 225 – Схема регистров USB контроллера

Таблица 577 – Регистры блока контроллера USB

| **Базовый адрес регистров USB контроллера 21010000** | | |
|---|---|---|
| **Общие регистры** | | |
| **Смещение** | **Название** | **Описание** |
| 0x0000_0000 | FAddr | Регистр адреса функции. |
| 0x0000_0001 | Power | Регистр управления питанием |
| 0x0000_0002 | IntrTx | Регистр прерывания для конечной точки 0 и точек Tx от 1 до 3 |
| 0x0000_0004 | IntrRx | Регистр прерываний для конечных точек Rx от 1 до 3 |
| 0x0000_0006 | IntrTxE | Регистр разрешения прерываний для IntrTx. |
| 0x0000_0008 | IntrRxE | Регистр разрешения прерываний для IntrRx. |
| 0x0000_000A | IntrUSB | Регистр прерываний для общих прерываний USB. |
| 0x0000_000B | IntrUSBE | Регистр разрешения прерываний для IntrUSB. |
| 0x0000_000C | Frame | Номер кадра |
| 0x0000_000E | Index | Индексный регистр для выбора регистров Status и Control конечных точек. |
| 0x0000_000F | Testmode | Включает тестовый режим USB 2.0 |
| **Индексные регистры периферийного режима** | | |
| 0x0000_0010 | TxMaxP | Максимальный размер пакета для периферийной конечной точки Tx. (Индексный регистр установлен только на конечные точки от 1 до 3) |
| 0x0000_0012 | CSR0 | Регистр состояния управления для конечной точки 0. (Индексный регистр установлен для выбора конечной точки 0) |
| | TxCSR | Регистр состояния управления для периферийной конечной точки Tx. (Индексный регистр установлен только на конечные точки от 1 до 3) |
| 0x0000_0014 | RxMaxP | Максимальный размер пакета для периферийной конечной точки Rx. (Индексный регистр установлен только на конечные точки от 1 до 3) |
| 0x0000_0016 | RxCSR | Регистр состояния управления для периферийной конечной точки Rx. (Индексный регистр установлен только на конечные точки от 1 до 3) |
| 0x0000_0018 | Count0 | Количество полученных байт в FIFO конечной точки 0. (Индексный регистр установлен только на конечную точку 0) |
| | RxCount | Количество байт данных в FIFO периферийной конечной точки Rx. (Индексный регистр установлен только на конечные точки от 1 до 3) |
| 0x0000_001A | - | Зарезервирован. Возвращаемое значение влияет на использование в хост-режиме (см. далее) |
| 0x0000_001C | - | Не используется, всегда хранит 0 |
| 0x0000_001F | ConfigData | Возвращает сведения о конфигурации ядра. (Регистр индекса установлен для выбора конечной точки 0.) |
| | FIFOSize | Возвращает сконфигурированный размер выбранных Rx FIFO и Tx FIFO (только для конечных точек 1 - 3) |
| **FIFO** | | |
| 0x0000_0020 | FIFOx | FIFO для конечных точек от 0 – 3 |
| 0x0000_0060 | DevCtl | Контрольный регистр устройств OTG |
| 0x0000_0061 | - | Зарезервировано |
| 0x0000_0062 | - | Зарезервировано |
| 0x0000_0063 | - | Зарезервировано |
| 0x0000_0064 | - | Зарезервировано |
| 0x0000_0066 | - | Зарезервировано |
| 0x0000_0068 | VControl/ VStatus | UTMI+ PHY регистры VENDOR |
| 0x0000_006C | HWVers | Регистр версии аппаратного обеспечения |
| 0x0000_006E | - | Не используется |
| **Целевой адрес регистра** | | |
| *x*0/8 | TxFuncAddr | Передача адреса функции конечной точки (только в хост-режиме) |
| *x*1/9 | - | Не используется. Всегда хранит 0 |

| Базовый адрес регистров USB контроллера 21010000 | | |
|---|---|---|
| **Общие регистры** | | |
| **Смещение** | **Название** | **Описание** |
| *x*2/A | TxHubAddr | Передача адреса конечной точки хаба (только в хост режиме) |
| *x*3/B | TxHubPort | Передача порта конечной точки хаба (только в хост режиме) |
| *x*4/C | RxFuncAddr | Получение адреса функции конечной точки (только в хост режиме) |
| *x*5/D | - | Не используется, всегда хранит 0 |
| *x*6/E | RxHubAddr | Получение адреса конечной точки хаба (только в хост режиме) |
| *x*7/F | RxHubPort | Получение порта конечной точки хаба (только в хост режиме) |

### 20.33.1 *Общие регистры*

#### 20.33.1.1 FADDR (только режим периферийного устройства)

FAddr – это 8-битный регистр, в который должен быть записан 7-битный адрес периферийной части передачи данных.

Когда контроллер используется в режиме периферии (DevCtl.D2 = 0), в этот регистр должен быть записан адрес, полученный с помощью команды SET_ADDRESS, который затем будет использоваться для декодирования адреса функции в последующих пакетах токенов.

Примечание – Этот регистр доступен только в операциях, выполняемых, когда контроллер находится в режиме периферийного устройства. В режиме хоста этот регистр игнорируется.

Таблица 578 – Регистр FAddr

| | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | 0 | Функциональный адрес | | | | | | |
| От ЦП | R | RW | RW | RW | RW | RW | RW | RW |
| От USB | - | R | R | R | R | R | R | R |

Таблица 579 – Описание бит регистра FAddr

| Бит | Имя | Функция |
|---|---|---|
| D7 | - | Не используется, всегда возвращает 0 |
| D6-D0 | **Func Addr** | Функциональный адрес |

#### 20.33.1.2 POWER

Power – это 8-битный регистр, который используется для управления передачей сигнала приостановкой и возобновлением, а также некоторыми дополнительными параметрами передачи данных контроллером.

Таблица 580 – Регистр Power

| | | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|---|
| | | ISO Update | **Зарезерви-ровано** | **HS Enab** | **HS Mode** | **Reset** | **Resume** | **Suspend Mode** | **Enable SuspendM** |
| Режим Периферийных устройств | От ЦП | RW | - | RW | R | R | RW | R | RW |
| | От USB | R | - | R | RW | RW | W | RW | R |
| Хост режим | От ЦП | - | - | RW | R | RW | RW | set | RW |
| | От USB | - | - | R | RW | RW | R | clear | R |

Таблица 581 – Описание бит регистра Power

| Бит | Имя | Функция |
|---|---|---|
| D7 | ISO update | При установке USB контроллер будет ожидать токена SOF с момента установки TxPktRdy перед отправкой пакета. Если токен IN принимается перед токеном SOF, тогда будет отправлен пакет данных нулевой длины.<br>*Примечание – Действует только в периферийном режиме. Кроме того, этот бит влияет только на конечные точки, выполняющие изохронные передачи* |
| D6 | **Зарезерви-ровано** | - |
| D5 | **HS Enab** | Если установлен через ЦП, контроллер попытается согласоваться для работы в режиме high-speed, когда устройство перезагрузится хабом. Если не установлено, устройство будет работать только в режиме full-speed |
| D4 | **HS Mode** | Когда установлено, этот бит «только для чтения» указывает на то, что высокоскоростной режим успешно согласован во время сброса USB. В периферийном режиме этот бит становится действительным после завершения сброса USB (как указано прерыванием Reset). В режиме хоста становится действительным, после сброса бита Reset. Остается действительным в течение всего сеанса |
| D3 | **Reset** | Этот бит устанавливается, когда на шине присутствует сигнал Reset.<br>*Примечание – Этот бит считывается/записывается в режиме хоста только для чтения в периферийном режиме* |
| D2 | **Resume** | Устанавливается ЦП для генерации сигнала Resume, когда функция находится в режиме Suspend. ЦП должен очистить этот бит через 10 мс (максимум 15 мс), чтобы завершить подачу сигнала Resume. В режиме хоста этот бит также автоматически устанавливается, когда обнаружен сигнал Resume от цели, если контроллер приостановлен |
| D1 | Suspend Mode | В режиме хоста этот бит устанавливается ЦП для перехода в режим Suspend. В Периферийном режиме этот бит устанавливается при входе в режим Suspend. Он сбрасывается, когда ЦП считывает регистр прерываний или устанавливает бит Resume (см. выше) |
| D0 | Enable SuspendM | Устанавливается ЦП для включения сигнала SUSPENDM |

Бит ISO Update влияет на все изохронные конечные точки IN в контроллере. Он обычно используется как способ обеспечения «чистого» запуска изохронного канала IN.

Бит HS Enab может использоваться для отключения режима high-speed. Обычно контроллер автоматически согласовывает работу high-speed при сбросе. Однако, если этот бит сброшен, контроллер останется в full-speed режиме, даже если он подключен к high-speed USB-порту.

Бит HS Mode может использоваться для определения того, находится ли контроллер в high- или full-speed режиме. Он будет установлен после того, как функция успешно согласуется для high-speed работы во время сброса USB.

Бит Reset может использоваться, чтобы определить, когда на USB-порте присутствует сигнал Reset. Бит будет установлен, когда будет обнаружена подача сигнала Reset и останется до тех пор, пока шина не вернется в состояние IDLE.

Бит Resume используется, чтобы заставить контроллер сгенерировать подачу сигнала Resume на USB для выполнения дистанционного пробуждения из режима Suspend. После того, как бит установлен, он должен простоять в течение приблизительно 10 мс (не менее 1 мс и не более 15 мс), а затем должен быть удален.

Бит Suspend Mode устанавливается либо для перехода в режим Suspend (в режиме хоста), либо при входе в режим Suspend (в периферийном режиме). Он будет сброшен при чтении регистра IntrUSB (в результате приема прерывания Suspend). Он также будет сброшен, если установить бит Resume, чтобы инициировать дистанционное пробуждение из режима Suspend.

Бит Enable SuspendM разрешает сигнал SUSPENDM, который указывает, находится ли контроллер в режиме Suspend. Если этот бит установлен, то сигнал SUSPENDM будет низким, если контроллер находится в режиме Suspend и высоким в любом другом случае.

#### 20.33.1.3 INTRTX

IntrTx – это 16-разрядный регистр чтения, который указывает, какие прерывания в настоящее время активны для конечной точки 0 и конечных точек Tx 1-3.

Примечание – Биты, относящиеся к конечным точкам, которые не были сконфигурированы, всегда будут возвращать 0. Обратите также внимание, что все активные прерывания сбрасываются при чтении этого регистра.

Таблица 582 – Регистр IntrTx

| | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - |
| От ЦП | r | r | r | R | r | R | r | r |
| От USB | set | set | set | Set | set | Set | set | set |

| | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | **EP3 Tx** | **E2 Tx** | **EP1 Tx** | **EP0 Tx** |
| От ЦП | r | r | r | R | r | R | r | r |
| От USB | set | set | set | Set | set | set | set | set |

Таблица 583 – Описание бит регистра IntrTx

| Бит | Имя | Функция |
|---|---|---|
| D15…D4 | - | Зарезервировано |
| D3 | **EP3 Tx** | Прерывание конечной точки Tx 3 |
| D2 | **EP2 Tx** | Прерывание конечной точки Tx 2 |
| D1 | **EP1 Tx** | Прерывание конечной точки Tx 1 |
| D0 | **EP0** | Прерывание конечной точки 0 |

#### 20.33.1.4 INTRRX

IntrRx – это 16-разрядный регистр только для чтения, который указывает, какие из прерываний для конечных точек Rx 1-3 активны в настоящее время. Примечание. Биты, относящиеся к конечным точкам, которые не были сконфигурированы, всегда будут возвращать 0. Обратите также внимание, что все активные прерывания сбрасываются при чтении этого регистра.

Таблица 584 – Регистр IntrRx

| | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - |
| От ЦП | r | r | r | r | r | r | r | r |
| От USB | set | set | set | set | set | set | set | set |

| | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | **EP3 Rx** | **E2 Rx** | **EP1 Rx** | **-** |
| От ЦП | r | r | r | r | r | r | r | r |
| От USB | set | set | set | set | set | set | set | r |

Таблица 585 – Описание бит регистра IntrRx

| Бит | Имя | Функция |
|---|---|---|
| D15…D4 | | Зарезервировано |
| D3 | **EP3 Rx** | Прерывание конечной точки Rx 3 |
| D2 | **EP2 Rx** | Прерывание конечной точки Rx 2 |
| D1 | **EP1 Rx** | Прерывание конечной точки Rx 1 |
| D0 | **-** | Не используется, всегда возвращает 0 |

#### 20.33.1.5 INTRTXE

IntrTxE – это 16-разрядный регистр, который предоставляет биты разрешения прерываний в IntrTx. При перезагрузке биты, соответствующие конечной точке 0 и используемым конечным точкам Tx, устанавливаются в 1, а остальные биты – в 0.

Примечание – Биты, относящиеся к конечным точкам, которые не были сконфигурированы, всегда будут возвращать 0.

Адрес:06h, значение reset 16'hFFFF замаскированы конечными точками Tx

Таблица 586 – Регистр IntrTxE

| | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - |
| От ЦП | rw | rw | rw | rw | rw | rw | rw | rw |
| От USB | r | r | r | r | r | r | r | r |

| | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | **EP3 Tx** | **E2 Tx** | **EP1 Tx** | **EP0** |
| От ЦП | rw | rw | rw | rw | rw | rw | rw | rw |
| От USB | r | r | r | r | r | r | r | r |

#### 20.33.1.6 INTRRXE

IntrRxE – это 16-разрядный регистр, который предоставляет биты разрешения прерываний в IntrRx. При перезапуске биты, соответствующие используемым конечным точкам Rx, устанавливаются в 1, а остальные биты – в 0.

Примечание – биты, относящиеся к конечным точкам, которые не были сконфигурированы, всегда будут возвращать 0.

Адрес:06h, значение reset 16'hFFFF замаскированы конечными точками Rx

Таблица 587 – Регистр IntrRxE

| | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | - |
| От ЦП | rw | rw | rw | rw | rw | rw | rw | rw |
| От USB | r | r | r | r | r | r | r | r |

| | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | **EP3 Rx** | **E2 Rx** | **EP1 Rx** | **-** |
| От ЦП | rw | rw | rw | rw | rw | rw | rw | r |
| От USB | r | r | r | r | r | r | r | r |

#### 20.33.1.7 INTRUSB

IntrUSB – это 8-разрядный регистр чтения, который показывает, какие USB-прерывания в настоящее время активны. Все активные прерывания будут сброшены при чтении этого регистра.

Таблица 588 – Регистр IntrUSB

| | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | **VBus Error** | **Sess Req** | **Discon** | **Conn** | **SOF** | **Reset/Babble** | **Resume** | **Suspend** |
| От ЦП | r | r | r | r | r | r | r | R |
| От USB | set | set | set | set | set | Set | set | Set |

Таблица 589 – Описание бит регистра IntrUSB

| Бит | Имя | Функция |
|---|---|---|
| D7 | **VBus Error** | Устанавливается, когда VBus падает ниже допустимого порога VBus во время сеанса. *Действует только тогда, когда контроллер является устройством типа «A»* |
| D6 | **Sess Req** | Устанавливается, когда обнаружен запрос начала сеанса. *Действует только в том случае, если контроллер является устройством типа «A»* |
| D5 | **Discon** | Устанавливается в хост режиме, когда обнаружено разъединение с устройством (то есть HOSTDISCON становится высоким). Устанавливается в режиме периферийного устройства при завершении сеанса |
| D4 | **Conn** | Устанавливается, когда обнаружено соединение с устройством (то есть сигнал HOSTDISCON становится низким). *Действителен только в хост режиме* |
| D3 | **SOF** | Устанавливается при начале нового кадра |
| D2 | **Reset** | Устанавливается в режиме периферийного устройства, когда на шине обнаружена подача сигнала сброса |
| | **Babble** | Устанавливается в хост режиме, когда обнаружены невостребованные данные |
| D1 | **Resume** | Устанавливается, когда обнаружена подача сигнала Resume в шине, если контроллер находится в режиме Suspend |
| D0 | **Suspend** | Устанавливается, если в шине обнаружена подача сигнала Suspend. *Действует только в режиме периферийного устройства* |

#### 20.33.1.8 INTRUSBE

IntrUSBE – это 8-битный регистр, который предоставляет биты разрешения прерываний в IntrUSB.

Таблица 590 – Регистр IntrUSBE

| | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | **VBus Error** | **Sess Req** | **Discon** | **Conn** | **SOF** | **Reset/Babble** | **Resume** | **Suspend** |
| От ЦП | rw | rw | rw | rw | rw | rw | rw | rw |
| От USB | r | r | r | r | r | r | r | r |

#### 20.33.1.9 FRAME

Frame – это 16-разрядный регистр чтения, который содержит последний принятый номер кадра.

Таблица 591 – Регистр Frame

| | D15 … D11 | D10 | … | D0 |
|---|---|---|---|---|
| | 0 0 0 0 0 | (MSB) | Номер кадра | (LSB) |
| От ЦП | r … r | r | … | r |
| От USB | w … w | w | … | w |

### 20.33.1.10 INDEX

Каждая конечная точка Tx и каждая конечная точка Rx имеют свой собственный набор регистров Control/Status. Кроме того, один набор регистров Control/Status Tx и один набор регистров Control/Status Rx отображаются в. Index представляет собой 4-битный регистр, который определяет, к какому регистру Control/Status можно получить доступ.

Таблица 592 – Регистр INDEX

| | D3 | D2 | D1 | D0 |
|---|---|---|---|---|
| | (MSB) | Выбранные конечные точки | | (LSB) |
| От ЦП | rw | rw | rw | rw |
| От USB | r | r | r | r |

Перед обращением к регистрам Control/Status конечной точки в номер конечной точки должен быть записан в регистр Index чтобы отобразились правильные регистры Control/Status.

### 20.33.1.11 TESTMODE

Testmode – это 8-разрядный регистр, который в основном используется для включения контроллера в один из четырех тестовых режимов для high-speed операций, описанных в спецификации USB 2.0, в ответ на команду SET FEATURE: TESTMODE. Он не используется при нормальной работе.

Таблица 593 – Регистр Testmode

| | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | **Force_Host** | **FIFO_Access** *(самоочищающийся)* | **Force_FS** | **Force_HS** | **Test_Packet** | **Test_K** | **Test_J** | **Test_SE0_NAK** |
| От ЦП | rw | Set | rw | rw | rw | rw | rw | rw |
| От USB | r | r | r | r | r | r | r | r |

Таблица 594 – Описание бит регистра Testmode

| Бит | Имя | Функция |
|---|---|---|
| D7 | **Force_Host** | ЦП устанавливает этот бит, чтобы дать указание ядру войти в режим Хоста, когда бит Session установлен, независимо от того, связан ли он с периферийным устройством или нет. Состояние входа CID, HostDisconnect и сигналы LineState игнорируются. Затем ядро останется в режиме хоста до тех пор, пока бит Session не будет сброшен. Даже если устройство отключено, и, если бит Force_Host установлен, контроллер снова войдет в режим Host в следующий раз, когда бит Session будет установлен. Пока контроллер находится в этом режиме статус сигнала HOSTDISCON из PHY может быть считан из бита 7 регистра DevCtl.<br>Скорость работы определяется по битам Force_HS и Force_FS следующим образом:<br>**Force_HS** / **Force_FS** / Скорость работы:<br>0 / 0 / Low-speed<br>0 / 1 / Full-speed<br>1 / 0 / High-speed<br>1 / 1 / Не определено |
| D6 | **FIFO_Access** | ЦП устанавливает этот бит для передачи пакета из конечной точки 0 Tx FIFO в конечную точку 0 Rx FIFO. Он очищается автоматически |

| | | |
|---|---|---|
| D5 | **Force_FS** | ЦП устанавливает этот бит либо в сочетании с 7 битом выше, либо для принудительного включения контроллера в full-speed режим, когда он получает сброс USB |
| D4 | **Force_HS** | ЦП устанавливает этот бит либо в сочетании с 7 битом выше, либо для принудительного включения контроллера в high-speed режим, когда он получает сброс USB |
| D3 | **Test_Packet** | (*High-speed режим*) ЦП устанавливает этот бит для входа в тестовый режим Test_Packet. В этом режиме контроллер повторно передает на шину тестовый пакет размером 53 байт, форма которого определена в *спецификации универсальной последовательной шины,* редакция 2.0, раздел 7.1.20. *Примечание* – тестовый пакет имеет фиксированный формат и должен быть загружен в FIFO конечной точку 0 до входа в тестовый режим |
| D2 | **Test_K** | (*High-speed режим*) ЦП устанавливает этот бит для входа в тестовый режим Test_K. В этом режиме контроллер подает логический уровень K на шину |
| D1 | **Test_J** | (*High-speed режим*) ЦП устанавливает этот бит для входа в тестовый режим Test_J. В этом режиме контроллер подает логический уровень J на шину |
| D0 | **Test_ SE0_NAK** | (*High-speed режим*) ЦП устанавливает этот бит для входа в тестовый режим Test_SE0_NAK. В этом режиме контроллер остается в *High-speed* режиме, но отвечает на любой действительный токен IN с помощью NAK |

### 20.33.1.12 DEVCTL

DevCtl – это 8-разрядный регистр, который используется для выбора режима работы контроллера (периферийное устройство или хост), а также для управления и контроля линии USB VBus.

Таблица 595 – Регистр DevCtl

| | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | **B-Device** | **FSDev** | **LSDe** | **VBus[1]** | **VBus[0]** | Host Mode | **Host Req** | **Session** |
| От ЦП | r | r | r | r | r | r | rw | rw |
| От USB | rw | rw | rw | rw | rw | r | r/clear | rw |

Таблица 596 – Описание бит регистра DevCtl

| Бит | Имя | Описание |
|---|---|---|
| D7 | **B-Device** | Этот бит только для чтения указывает, работает ли контроллер как «A» или «B» устройство. 0 – «A» устройство; 1 – «B». *Действителен только во время сеанса*. *Примечание – Если ядро находится в режиме Force_Host (то есть сеанс запущен с Testmode.D7 = 1), это бит будет указывать состояние входного сигнала HOSTDISCON из PHY* |
| D6 | **FSDev** | Этот бит только для чтения устанавливается при подключении full- или high-speed устройства к порту. (устройства high-speed отличают от full-speed при сбросе устройства по высокоскоростному импульсу.) *Действует только в режиме хоста* |
| D5 | **LSDev** | Этот бит только для чтения, устанавливается при обнаружении low-speed устройства, подключенного к порту. *Действует только в режиме хоста* |
| D4 – D3 | **VBus[1:0]** | Эти биты только для чтения, кодируют текущий уровень VBus следующим образом: |

| D4 | D3 | Значение |
|---|---|---|
| 0 | 0 | Ниже SessionEnd |

| | | | | | |
|---|---|---|---|---|---|
| | | 0 | 1 | Выше SessionEnd, ниже AValid | |
| | | 1 | 0 | Выше A Valid, ниже VBusValid | |
| | | 1 | 1 | Выше VBusValid | |
| D2 | **Host Mode** | Этот бит только для чтения устанавливается, когда контроллер действует как хост | | | |
| D1 | **Host Req** | Когда бит установлен, контроллер инициирует хост-взаимодействие, если он в режиме Suspend. Бит сбрасывается, когда хост-взаимодействие завершено | | | |
| D0 | **Session** | *При работе в качестве устройства «A»* этот бит устанавливается или сбрасывается ЦП для запуска или завершения сеанса.<br>*При работе в качестве устройства «B»* этот бит устанавливается/сбрасывается контроллером, когда сеанс начинается/заканчивается.<br>Он также устанавливается ЦП для инициирования протокола Session Request или очищается ЦП, в Suspend режиме для отключения программного обеспечения | | | |

### 20.33.2 *Индексные регистры*

Действие следующих регистров, если выбранная конечная точка не была настроена, не определено.

#### 20.33.2.1 CSR0

CSR0 – это 16-разрядный регистр, который содержит биты управления и состояния конечной точки 0. Интерпретация регистра зависит от того, действует ли контроллер как периферийное устройство или как хост.

***CSR0 в режиме периферийного устройства:***

| | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | - | - | **FlushFIFO** *(самоочищающийся***)** |
| От ЦП | r | R | r | r | r | r | r | set |
| От USB | r | R | r | r | r | r | r | r |

| | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | Serviced SetupEnd (*самоочищающийся)* | Serviced RxPktRdy (*самоочищающийся)* | **SendStall (***самоочищающийся)* | **SetupEnd** | **DataEnd (***самоочищающийся)* | **SentStall** | **TxPktRdy (***самоочищающийся)* | **RxPktRdy** |
| От ЦП | set | set | set | r | set | r/set | r/set | r |
| От USB | r | R | r | set | r | set | r | set |

Таблица 597 – Описание бит регистра CSR0 в режиме периферийного устройства

| Бит | Имя | Описание |
|---|---|---|
| D15-D8 | - | Не используется, при чтении возвращает 0 |
| D8 | **FlushFIFO** | ЦП записывает 1 в этот бит, чтобы удалить следующий пакет для передачи/чтения из FIFO конечной точки 0. Указатель FIFO очищается, а бит TxPktRdy/RxPktRdy (ниже) сбрасывается.<br>*Примечание – FlushFIFO не используется, если не установлен TxPktRdy/RxPktRdy* |
| D7 | **ServicedSetupEnd** | ЦП записывает 1 в этот бит, чтобы сбросить бит SetupEnd. Он сбрасывается автоматически |
| D6 | **ServicedRxPktRdy** | ЦП записывает 1 в этот бит, чтобы сбросить бит RxPktRdy. Он сбрасывается автоматически |

| Бит | Имя | Описание |
|---|---|---|
| D5 | **SendStall** | ЦП записывает 1 в этот бит для прекращения текущей передачи данных. STALL будет передан, и этот бит будет сброшен автоматически |
| D4 | **SetupEnd** | Этот бит будет установлен, если передача транзакции управления завершится до того, как бит DataEnd будет установлен.<br>Прерывание будет сгенерировано, и FIFO очистится в то же время. ЦП сбрасывает бит при помощи записи 1 в бит ServicedSetupEnd |
| D3 | **DataEnd** | ЦП устанавливает этот бит:<br>1. При установке TxPktRdy для последнего пакета данных.<br>2. При сбросе RxPktRdy после выгрузки последнего пакета данных.<br>3. При установке TxPktRdy для пакета данных с нулевой длиной.<br>Он сбрасывается автоматически |
| D2 | **SentStall** | Этот бит устанавливается при передаче STALL. ЦП должен сбросить этот бит |
| D1 | **TxPktRdy** | ЦП устанавливает этот бит после загрузки пакета данных в FIFO. Он автоматически сбрасывается после передачи пакета данных. Прерывание генерируется (если включено), когда бит сбрасывается |
| D0 | **RxPktRdy** | Этот бит устанавливается после принятия пакета данных. Прерывание сгенерируется, если этот бит установлен. ЦП сбрасывает этот бит, при помощи установки ServicedRxPktRdy |

***CSR0 в хост режиме:***

| | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
|---|---|---|---|---|---|---|---|---|
| | - | - | - | - | - | **Data Toggle Wr. Enable** *(самоочищающийся)* | **Data Toggle** | **FlushFIF0** *(самоочищающийся)* |
| От ЦП | r | r | r | r | r | Set | rw | set |
| От USB | r | r | r | r | r | R | rw | r |

| | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | NAK Timeout | **StatusPkt** | **ReqPkt** | **Error** | **SetupPkt** | **RxStall** | **TxPktRdy** | **RxPktRdy** |
| От ЦП | r/clear | rw | rw | r | rw | r/clear | r/set | r/clear |
| От USB | set | r | rw | Set | R | set | clear | rw |

Таблица 598 – Описание бит регистра CSR0 в хост режиме

| Бит | Имя | Описание |
|---|---|---|
| D15-D11 | | Не используется. При чтении возвращает 0 |
| D10 | **Data Toggle Write Enable** | ЦП записывает 1 в этот бит, чтобы разрешить запись текущего состояния Data Toggle конечной точки 0 (см. бит Data Toggle ниже). Этот бит автоматически сбросится, как только новое значение будет записано |
| D9 | **Data Toggle** | При чтении этот бит указывает текущее состояние Data Toggle конечной точки 0. Если D10 установлен, требуемое состояние изменения данных может быть записано в этот бит. Если D10 сброшен, любое значение, записанное в этот бит, игнорируется |
| D8 | **FlushFIFO** | ЦП записывает 1 в этот бит, чтобы очистить следующий пакет, который должен быть передан/прочитан из FIFO конечной точки 0. Указатель FIFO очищается, а бит TxPktRdy/RxPktRdy (ниже) сбрасывается. |

| Бит | Имя | Описание |
|---|---|---|
| | | *Примечание – FlushFIFO не действует, если не установлен TxPktRdy/RxPktRdy* |
| D7 | **NAK Timeout** | Этот бит будет установлен, когда конечная точка 0 будет остановлена после получения ответов NAK за время больше, чем установлено регистром NAKLimit0 |
| D6 | **StatusPkt** | ЦП устанавливает этот бит одновременно с установкой бита TxPktRdy или ReqPkt, чтобы выполнить передачу транзакции состояния. Установка этого бита гарантирует, что в Data Toggle установлена 1, а значит пакет DATA1 используется для передачи Status Stage |
| D5 | **ReqPkt** | ЦП устанавливает этот бит для запроса направления транзакции IN. Он сбрасывается, когда устанавливается бит RxPktRdy |
| D4 | **Error** | Этот бит будет установлен, после трех попыток выполнить передачу без ответа от периферии. ЦП должен сбросить этот бит. Прерывание сгенерируется, когда этот бит будет установлен |
| D3 | **SetupPkt** | ЦП устанавливает этот бит вместе с битом TxPktRdy, для отправки токена SETUP вместо токена OUT для передачи.<br>*Примечание – Установка этого бита также сбрасывает Data Toggle* |
| D2 | **RxStall** | Этот бит устанавливается при получении STALL. ЦП должен сбросить этот бит |
| D1 | **TxPktRdy** | ЦП устанавливает этот бит после загрузки пакета данных в FIFO. Он автоматически сбрасывается, после передачи пакета данных. Прерывание сгенерируется (если включено), после сброса бита |
| D0 | **RxPktRdy** | Этот бит устанавливается, после принятия пакета данных. Прерывание сгенерируется (если включено), после установки этого бита. ЦП должен сбросить этот бит, после прочтения пакета из FIFO |

### 20.33.2.2 COUNT0

Count0 – это 7-разрядный регистр только для чтения, который указывает количество принятых байт данных из FIFO конечной точки 0. Значение регистра изменяется вместе с содержимым FIFO и действительно только пока установлен RxPktRdy (CSR0.D0).

| | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| | (MSB) | Count Rx конечной точки 0 | | | | | (LSB) |
| От ЦП | r | r | R | r | r | r | r |
| От USB | w | w | W | w | w | W | w |

### 20.33.2.3 TYPE0 (только хост режим)

Type0 – это 8-битный регистр, из которого реализованы только биты 6 и 7. В эти биты должна быть записана рабочая скоростью целевого устройства.

| | D7 | D6 |
|---|---|---|
| | Speed | |
| От ЦП | rw | rw |
| От USB | r | r |

| Бит | Имя | Функция |
|---|---|---|
| D7-D6 | Speed | Скорость работы целевого устройства:<br>00: *Не используется*<br>01: high<br>10: full<br>11: low |

### 20.33.2.4 CONFIGDATA

ConfigData – это 8-битный регистр только для чтения, который возвращает информацию о выбранной конфигурации ядра.

| | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | **MPRxE** | **MPTxE** | **BigEndian** | **HBRxE** | **HBTxE** | DynFIFO Sizing | **Зарезерви-ровано** | UTMI DataWidth |
| От ЦП | r | r | R | R | r | r | r | r |
| От USB | r | r | R | R | r | r | r | r |

| Бит | Имя | Функция |
|---|---|---|
| D7 | **MPRxE** | Когда установлено значение «1», выбирается автоматическое объединение пакетов поточного типа. |
| D6 | **MPTxE** | Когда установлено значение «1», выбирается автоматическое разделение пакетов поточного типа. |
| D5 | **BigEndian** | Когда установлено значение «1», выбирается порядок Big Endian. |
| D4 | **HBRxE** | Когда установлено значение «1», выбирается поддержка конечной точки ISO RX с высокой пропускной способностью. |
| D3 | **HBTxE** | Когда установлено значение «1», выбирается поддержка конечной точки ISO TX с высокой пропускной способностью. |
| D2 | **-** | Зарезервировано |
| D1 | **Зарезерви-ровано** | - |
| D0 | **UTMI DataWidth** | Указывает выбранную ширину UTMI +. 0 → 8 бит; 1 → 16 бит. |

### 20.33.2.5 NAKLIMIT0 (только хост режим)

NAKLimit0 – это 5-битный регистр, который устанавливает количество кадров/микрокадров (high-speed режим), после которых конечная точка 0 превысит время получения NAK. (Эквивалентные настройки для других конечных точек могут быть выполнены через их регистры TxInterval и RxInterval.)

Количество выбранных кадров/микрокадров равно $2^{(m-1)}$ (где m – значение от 2 до 16, установленное в регистре). Если хост получает ответы NAK от цели для большего количества кадров, чем число, представленное лимитом, установленным в этом регистре, конечная точка будет остановлена. *Примечание*. *Значение m = 0 или 1 отключает функцию превышения времени получения ответа NAK.*

| | D4 | … | D0 |
|---|---|---|---|
| | (MSB) | **Endpoint 0 NAK Limit ( *m*)** | (LSB) |
| От ЦП | Rw | | rw |
| От USB | R | | r |

### 20.33.2.6 TXMAXP

Регистр TxMaxP определяет максимальный объем данных, который может быть передан через выбранную конечную точку Tx за одну операцию. Для каждой конечной точки Tx существует регистр TxMaxP (кроме конечной точки 0).

*Адрес:10h, значение reset 11/12/16'h0000*

| | D2/d15 | | D11 | D10 | … | D0 |
|---|---|---|---|---|---|---|
| | | ***m* – 1** | | (MSB) | **Maximum Payload/transaction** | (LSB) |
| От ЦП | rw | … | rw | rw | … | rw |
| От USB | r | … | r | r | … | r |

Биты 10:0 определяют (в байтах) максимальную полезную нагрузку, передаваемую в одной транзакции. Нагрузка может быть до 1024 байт, но ограничена спецификацией USB для размеров пакетов для поточных и изохронных типов передач и передачи по прерываниям в high- и full-speed операциях.

Если опция high-bandwidth для изохронных конечных точек или точек передачи по прерываниям или опция разделения пакетов поточных конечных точек была установлена, когда ядро конфигурировалось, то регистр включает в себя либо 2 или 5 дополнительных бит. Эти биты определяют множитель m, который равен значению большему на 1, чем записано в регистре.

В случае поточных конечных точек с включенной опцией разделения пакетов, множитель m может принимать значения не более 32 и определяет максимальное количество USB пакетов (то есть пакетов для передачи по USB) указанной полезной нагрузки, в которой один пакет данных, размещенный в FIFO, должен быть разделен до передачи. (Если опция разделения пакетов не включена, D15-D13 не используется и D12-D11 (если включен) игнорируется.) *Примечание*: *Пакет данных должен быть точно кратен полезной нагрузке, задаваемой битами 10:0, которая сама по себе может быть либо 8, 16, 32, 64, либо (в случае high-speed передач) 512 байт.*

Для изохронных конечных точек и точек прерывания, работающих в high-speed режиме и с включенной опцией высокой пропускной способности, множитель m может принимать значения только 2 или 3 (соответственно 11-й или 12-й бит установлен) и определяет максимальное количество транзакций, которые могут иметь место в одном микрокадре. Если бит 11 или бит 12 отличен от нуля, USB контроллер автоматически разделяет любой пакет данных, записанный в FIFO, на 2 или 3 USB пакета, каждый из которых содержит указанную полезную нагрузку (или меньше). Максимальная полезная нагрузка для каждой транзакции составляет 1024 байта, что позволяет передавать до 3072 байт в каждом микрокадре. (Для изохронной передачи и передачи по прерываниям в full-speed режиме или если high-speed передача данных не включена, биты 11 и 12 игнорируются.)

Значение, записанное в биты [10:0], (умноженное на m в случае изохронной передачи и передачи по прерываниям с высокой пропускной способностью), должно соответствовать значению, указанному в поле wMaxPacketSize стандартного дескриптора конечной точки для связанной конечной точки. Несоответствие может привести к непредсказуемым последствиям.

Общий объем данных, указанный значением, записанным в этот регистр (указанная полезная нагрузка × m), не должен превышать размер FIFO для конечной точки Tx и не должен превышать половину размера FIFO, если требуется двойная буферизация.

Если этот регистр изменяется после того, как пакеты были отправлены, FIFO конечной точки Tx должен быть полностью очищен (используя бит FlushFIFO в TxCSR) после записи нового значения в регистр.

### 20.33.2.7 TXCSR

TxCSR представляет собой 16-разрядный регистр, который предоставляет биты управления и состояния (Control/Status) передачи через выбранную конечную точку Tx. Для каждой сконфигурированной конечной точки Tx существует регистр TxCSR (кроме конечной точки 0).

Интерпретация регистра зависит от того, используется ли USB контроллер как периферийное устройство или как хост.

***В режиме периферийного устройства:***

| | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
|---|---|---|---|---|---|---|---|---|
| | **AutoSet** | **ISO** | **Mode** | **DMAReqEnab** | **FrcDataTog** | **DMAReqMode** | - | - |
| От ЦП | rw | rw | rw | rw | rw | rw | r | r |
| От USB | r | r | r | r | r | r | r | r |

| | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | IncompTx | ClrData Tog | SentStall | SendStall | FlushFIFO (самоочищающийся) | UnderRun | FIFO NotEmpty | TxPktRdy |
| От ЦП | r/clear | set | r/clear | rw | set | r/clear | r/clear | r/set |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| От USB | set | r/clear | Set | r | r | set | set | clear |

| Бит | Имя | Функция в режиме периферийного устройства |
|---|---|---|
| D15 | **AutoSet** | Если ЦП устанавливает этот бит, TxPktRdy будет автоматически установлен, когда данные максимального размера (значение в TxMaxP) загрузятся в FIFO точки Tx. Если размер пакета меньше максимального, то TxPktRdy необходимо будет установить вручную. *Примечание – Не следует устанавливать для high-speed изохронных конечных точек* |
| D14 | **ISO** | ЦП устанавливает этот бит для использования конечной точки Tx для изохронной передачи данных и сбрасывает его, чтобы разрешить конечной точке Tx поточную передачу или передачу по прерываниям. *Примечание – Этот бит действителен только в режиме периферийного устройства. В хост режиме он всегда возвращает 0* |
| D13 | **Mode** | ЦП устанавливает этот бит для включения использования конечной точки как Tx и очищает его, чтобы использовать конечную точку как Rx. *Примечание – Этот бит действителен, когда одна и та же конечная точка FIFO используется как для Tx, так и для Rx передачи данных*. |
| D12 | **DMAReqEnab** | ЦП устанавливает этот бит для разрешения запросов DMA для конечной точки Tx |
| D11 | **FrcDataTog** | ЦП устанавливает этот бит для принудительного изменения Data Toggle конечных точек, и очистке пакета данных в FIFO, независимо от того, получен ли ACK или нет |
| D10 | **DMAReqMode** | ЦП устанавливает этот бит для выбора режима запроса DMA 1 и очищает его, чтобы выбрать режим запроса DMA 0 |
| D9–D8 | – | Не используется, всегда возвращает 0 |
| D7 | **IncompTx** | Когда конечная точка используется для high-speed изохронной передачи данных или передачи данных по порываниям, этот бит устанавливается, чтобы указать, где большой пакет был разбит на 2 или 3 для передачи, но для отправки всех деталей было получено недостаточное количество токенов. *Примечание – Во всех других случаях пропускной способности, этот бит всегда будет возвращать 0* |
| D6 | **ClrDataTog** | ЦП записывает 1 в этот бит, чтобы сбросить Data Toggle конечных точек до нуля |
| D5 | **SentStall** | Этот бит устанавливается при передаче STALL. FIFO очищается, а бит TxPktRdy (см. ниже) сбрасывается. ЦП должен бросить этот бит |
| D4 | **SendStall** | ЦП записывает 1 в этот бит, чтобы передать STALL токену IN. ЦП должен сбросить этот бит для прекращения состояния STALL. *Примечание – Этот бит не используется, когда конечная точка используется для изохронной передачи данных* |
| D3 | **FlushFIFO** | ЦП записывает 1 в этот бит, чтобы очистить следующий пакет, который должен быть передан по FIFO конечной точки Tx. Указатель FIFO очищается, и бит TxPktRdy (см. ниже) сбрасывается. *Примечание – FlushFIFO не используется, если установлен TxPktRdy. Также обратите внимание, если FIFO имеет двойную буферизацию, возможно FlushFIFO будет необходимо установить дважды, чтобы полностью очистить FIFO* |
| D2 | **UnderRun** | USB устанавливает этот бит, если токен IN принимается, если бит TxPktRdy не установлен. ЦП должен сбросить этот бит |
| D1 | **FIFONotEmpty** | USB устанавливает этот бит, когда в FIFO точки Tx имеется по меньшей мере 1 пакет |
| D0 | **TxPktRdy** | ЦП устанавливает этот бит после загрузки пакета данных в FIFO. Он автоматически сбросится, когда пакет данных будет передан. Прерывание сгенерируется (если включено), когда бит сбросится |

***В хост режиме:***

| | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
|---|---|---|---|---|---|---|---|---|
| | **AutoSet** | **-** | **Mode** | **DMAReqEnab** | **FrcDataTog** | **DMAReqMode** | **Data Toggle Wr.Enable** | **Data Toggle** |
| От ЦП | rw | r | rw | Rw | rw | rw | set | Rw |
| От USB | r | r | r | R | r | r | r | Rw |

| | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | NAK Timeout | ClrData Tog | **RxStall** | **SetupPkt** | FlushFIFO (самоочищающийся) | Error | FIFO NotEmpty | TxPktRdy |
| От ЦП | r/clear | set | r/clear | Rw | set | r/clear | r/clear | r/set |
| От USB | set | r/clear | set | R | r | rw | set | clear |

| Бит | Имя | Функции в хост режима |
|---|---|---|
| D15 | **AutoSet** | Если ЦП установил этот бит, то TxPktRdy будет автоматически установлен, когда данные максимального размера (значение в TxMaxP) загрузятся в FIFO точки Tx. Если размер пакета меньше размера максимального, то TxPktRdy необходимо будет установить вручную. *Примечание. Не следует устанавливать для high-speed изохронных конечных точек* |
| D14 | | Не используется, всегда возвращает 0. |
| D13 | **Mode** | ЦП устанавливает этот бит для использования конечной точки как Tx и сбрасывает его, чтобы использовать как Rx. *Примечание*. Этот бит используется, когда одна и та же конечная точка FIFO используется для передачи данных как Tx и как Rx |
| D12 | **DMAReqEnab** | ЦП устанавливает этот бит для включения запроса DMA для конечной точки Tx |
| D11 | **FrcDataTog** | ЦП должен установить этот бит для принудительного изменения Data Toggle конечных точек, и очистки пакета данных в FIFO, независимо от того, получен ACK или нет |
| D10 | **DMAReqMode** | ЦП устанавливает этот бит для выбора режима запроса DMA 1 и очищает его, чтобы выбрать режим запроса DMA 0 |
| D9 | Data Toggle Write Enable | ЦП записывает 1 в этот бит, чтобы разрешить запись состояния Data Toggle конечной точки Tx (см. бит Data Toggle ниже). Этот бит автоматически сбрасывается после записи нового значения |
| D8 | **Data Toggle** | При чтении этот бит указывает текущее состояние Data Toggle конечной точки Tx. Если D10 установлен, то требуемое значение Data Toggle может быть записано. Если D10 сброшен, то любое записанное значение игнорируется |
| D7 | **NAK Timeout** | Этот бит будет установлен, когда конечная точка Tx будет остановлена из-за превышения времени получения ответов NAK (максимальное время установлено в регистре TxInterval). ЦП должен сбросить этот бит, чтобы позволить конечной точке возобновить прием. *Примечание. Действует только для поточных конечных точек* |
| D6 | **ClrDataTog** | ЦП записывает 1 в этот бит, чтобы сбросить Data Toggle точек до нуля |
| D5 | **RxStall** | Этот бит устанавливается при передаче STALL. FIFO и бит TxPktRdy (см. ниже) очищаются. ЦП должен сбросить этот бит |
| D4 | **SetupPkt** | ЦП устанавливает этот бит вместе с TxPktRdy, чтобы использовать SETUP токен вместо OUT токен для транзакции. *Примечание: установка этого бита также сбрасывает Data Toggle* |
| D3 | **FlushFIFO** | ЦП записывает 1 в этот бит, чтобы удалить следующий пакет, который должен быть передан по FIFO конечной точки Tx. Указатель FIFO очищается, и бит TxPktRdy (см. ниже) сбрасывается. *Примечание. FlushFIFO не имеет эффекта если установлен TxPktRdy. Также если FIFO двойной буферизации,* |

| Бит | Имя | Функции в хост режима |
|---|---|---|
| | | *FlushFIFO возможно придется установить дважды, чтобы полностью очистить FIFO* |
| D2 | **Error** | USB устанавливает этот после трех попыток отправить пакет без получения STALL. ЦП должен сбросить этот бит. Прерывание сгенерируется, когда бит будет установлен. Используется только, если конечная точка работает в поточном режиме или по прерываниям |
| D1 | **FIFONotEmpty** | USB устанавливает этот бит, когда в FIFO точки Tx имеется по меньшей мере один пакет |
| D0 | **TxPktRdy** | ЦП устанавливает этот бит после загрузки пакета данных в FIFO. Он автоматически сбрасывается, после передачи пакета данных. Прерывание сгенерируется (если включено), после очистки бита |

#### 20.33.2.8 RXMAXP

Регистр RxMaxP определяет максимальный объем данных, который может быть передан через выбранную конечную точку Rx за одну операцию. Регистр RxMaxP определен для каждой конечной точки Rx (кроме конечной точки 0).

| | D2/d15 | | D11 | D10 | … | D0 |
|---|---|---|---|---|---|---|
| | | ***m* – 1** | | (MSB) | **Maximum Payload/transaction** | (LSB) |
| От ЦП | rw | … | rw | rw | … | rw |
| От USB | r | … | r | r | … | r |

Биты 10: 0 определяют (в байтах) максимальную полезную нагрузку одной передачи данных. Набор значений может быть до 1024 байт, но подлежит ограничениям, установленным Спецификацией USB для размеров пакетов для поточных и изохронных передач и передач данных и по прерываниям в full- и high-speed операциях.

Если опция конечных точек с высокой пропускной способностью или объединения поточных пакетов была установлена при конфигурации ядра, то регистр включает в себя либо 2, либо 5 дополнительных битов, которые определяют множитель m, который равен записанному значению плюс 1.

Для поточных конечных точек с включенной опцией объединения пакетов множитель m может быть не более 32 и определяет количество USB-пакетов указанной полезной нагрузки, которые должны быть объединены в один пакет данных в FIFO. (Если опция разделения пакетов не включена, D15-D13 не используются и D12-D11 (если включены) игнорируются.)

Для изохронных конечных точек и точек по прерыванию, работающих в high-speed режиме и с включенной опцией высокой пропускной способности, множитель m может быть только 2 или 3 (соответствует установленному биту 11 или установленному биту 12), и он определяет максимальное количество передач данных, которые могут иметь место в одном микрокадре. Если бит 11 или бит 12 отличен от нуля, контроллер автоматически объединяет отдельные USB-пакеты, полученные в любом микрокадре, в один пакет в FIFO точки Rx. Максимальная полезная нагрузка для каждой передачи данных составляет 1024 байта, что позволяет получать до 3072 байт в каждом микрокадре. (Для изохронной передачи и передачи по прерываниям в full-speed режиме или если high-speed передача данных не включена, биты 11 и 12 игнорируются.)

#### 20.33.2.9 RXCSR

RxCSR представляет собой 16-разрядный регистр, который предоставляет биты управления и состояния (Control/Status) передачи через выбранную конечную точку Rx. Регистр RxCSR определен для каждой сконфигурированной конечной точки Rx (кроме конечной точку 0).

Интерпретация регистра зависит от того, действует ли USB контроллер как периферийное устройство или как хост.

***В режиме периферийного устройства:***

| | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
|---|---|---|---|---|---|---|---|---|
| | **AutoClear** | **ISO** | **DMAReqEnab** | **DisNyet** | **DMAReqMode** - | - | - | **IncompRx** |
| От ЦП | rw | rw | rw | rw | Rw | r | r | r |
| От USB | r | r | r | r | r | r | r | set |

| | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | **ClrDataTog** | **SentStall** | **SendStall** | **FlushFIFO (самоочищающийся)** | **DataError** | **OverRun** | **FIFOFull** | **RxPktRdy** |
| От ЦП | set | r/clear | rw | set | r | r/clear | r | r/clear |
| От USB | r/clear | set | r | r | set | set | set | set |

| Бит | Имя | Функция в режиме периферийного устройства |
|---|---|---|
| D15 | **AutoClear** | Если ЦП устанавливает этот бит, то RxPktRdy будет автоматически сброшен, когда пакет байт RxMaxP будет выгружен из FIFO точки Rx. Когда пакеты размера меньше максимального не выгружены, RxPktRdy необходимо будет сбросить вручную. *Примечание*: *не следует устанавливать для high-speed изохронных конечных точек.* |
| D14 | **ISO** | ЦП устанавливает этот бит для использования конечной точки Rx для изохронной передачи данных и сбрасывает его, чтобы использовать конечную точку Rx для поточной передачи или передачи по прерываниям. |
| D13 | **DMAReqEnab** | ЦП устанавливает этот бит для разрешения запроса DMA для конечной точки Rx. |
| D12 | **DisNyet** | ЦП устанавливает этот бит, чтобы отключить отправку NYET. Когда бит установлен, все успешно полученные Rx-пакеты являются ACK пакетами, включая тот, на котором происходит переполнение FIFO. *Примечание*: *этот бит используется только в high-speed режиме, в котором он должен быть установлен для всех конечных точек по прерываниям.* |
| D11 | **DMAReqMode** | ЦП устанавливает этот бит для выбора режима DMA 1 и сбрасывает этот бит, чтобы выбрать режим DMA 0. |
| D10-D9 | - | Не используется, всегда возвращает 0 |
| D8 | **IncompRx** | Этот бит устанавливается в high-speed изохронной передаче, если пакет в FIFO точки Rx является неполным, потому что части данных не были получены. Он сбрасывается, когда RxPktRdy сбрасывается. *Примечание*: всегда кроме high-speed режима, этот бит будет хранить 0. |
| D7 | **ClrDataTog** | ЦП записывает 1 в этот бит, чтобы сбросить Data Toggle конечных точек до 0. |
| D6 | **SentStall** | Этот бит устанавливается при получении STAL. ЦП должен сбросить этот бит. |
| D5 | **SendStall** | ЦП записывает 1 в этот бит для отправки STAL. ЦП сбрасывает этот бит для завершения STALL. *Примечание*. *Этот бит не используется, когда конечная точка используется для изохронной передачи.* |
| D4 | **FlushFIFO** | ЦП записывает 1 в этот бит, чтобы очистить следующий пакет, который должен прочитаться из FIFO конечной точки Rx. Указатель FIFO очищается, и бит RxPktRdy (см. ниже). Сбрасывается. *Примечание*: *FlushFIFO не используется если не установлен RxPktRdy. Также если FIFO двойной буферизации, FlushFIFO возможно придется установить дважды для полной очистки FIFO.* |

| | | |
|---|---|---|
| D3 | **DataError** | Этот бит устанавливается, если RxPktRdy установлен и пакет данных имеет ошибку CRC или Bit Stuff. Он сбрасывается, когда RxPktRdy сбрасывается. *Примечание*. *Этот бит используется только в том случае, если конечная точка работает в режиме ISO. В поточном режиме, он всегда возвращает 0.* |
| D2 | **OverRun** | Этот бит устанавливается, если пакет OUT не может быть загружен в FIFO точки RX. ЦП должен сбросить этот бит. *Примечание*. *Этот бит действует, когда конечная точка работает в режиме ISO. В поточном режиме он всегда возвращает 0.* |
| D1 | **FIFOFull** | Этот бит устанавливается, когда в FIFO точки RX больше нельзя загружать пакеты. |
| D0 | **RxPktRdy** | Этот бит устанавливается, когда пакет данных принят. ЦП должен сбросить этот бит, когда пакет будет выгружен из FIFO точки RX. Прерывание сгенерируется, когда бит будет установлен. |

***В хост- режиме:***

| | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
|---|---|---|---|---|---|---|---|---|
| | **AutoClear** | **AutoReq** | **DMAReqEnab** | **DisNyet** | **DMAReqMode** - | **Data Toggle Wr. Enable** (самоочищающийся) | **Data Toggle** - | **IncompRx** |
| От ЦП | rw | rw | rw | Rw | rw | **r** | r | r |
| От USB | r | r | r | R | r | r | r | set |

| | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | **ClrDataTog** | **RxStall** | **ReqPkt** | **Flush FIFO (самоочищающийся)** | **DataError/ NAK Timeout** | **OverRun** | **FIFOFull** (самоочищающийся) | **RxPktRdy** |
| От ЦП | set | r/clear | rw | set | r(/clear) | r/clear | r | r/clear |
| От USB | r/clear | set | rw | r | set | set | set | set |

| Бит | Имя | Функция в хост-режиме |
|---|---|---|
| D15 | **AutoClear** | Если ЦП устанавливает этот бит, RxPktRdy будет автоматически сброшен, если пакет байт RxMaxP будет выгружен из RX FIFO. Когда пакеты с размером меньше максимального не выгружены, RxPktRdy необходимо будет сбросить вручную. *Примечание*: *Не следует устанавливать для high-speed изохронных конечных точек.* |
| D14 | **AutoReq** | Если процессор устанавливает этот бит, то ReqPkt будет автоматически установлен, когда бит RxPktRdy будет сброшен. |
| D13 | **DMAReqEnab** | ЦП устанавливает этот бит для разрешения запроса DMA для конечной точки Rx. |
| D12 | **DisNyet** | ЦП устанавливает этот бит, чтобы отключить отправку NYET. Когда бит установлен, все успешно полученные Rx-пакеты являются ACK, включая тот пакет, на котором FIFO наполнилось. *Примечание*: *этот бит используется только в high-speed режиме, в котором он должен быть установлен для всех конечных точек по прерываниям*. |
| D11 | **DMAReqMode** | ЦП устанавливает этот бит для выбора режима DMA 1 и сбрасывает этот бит, чтобы выбрать режим DMA 0. |
| D10 | Data Toggle Write Enable | ЦП записывает 1 в этот бит, чтобы разрешить запись Data Toggle конечной точки Rx (см. бит Data Toggle ниже). Этот бит автоматически сбрасывается после записи нового значения. |
| D9 | Data Toggle | При чтении этот бит указывает текущее значение Data Toggle конечной точки Rx. Если D10 установлен, то в этот бит может быть |

| | | |
|---|---|---|
| | | записано требуемое значением Data Toggle. Если D10 сброшен, любое значение, записанное в этот бит, игнорируется. |
| D8 | **IncompRx** | Этот бит устанавливается в high-speed изохронной передаче, если пакет в FIFO точки Rx пришел неполным, потому что части данных не были получены. Он сбрасывается, когда RxPktRdy сбрасывается. *Примечание. Если протоколы USB соблюдаются, этот бит никогда не должен устанавливаться. Установленный бит указывает на то, что связанное периферийное устройство ведет себя неправильно. (в любом режиме, кроме режима с высокой полосой пропускания, этот бит всегда будет возвращать 0.)* |
| D7 | **ClrDataTog** | ЦП записывает 1 в этот бит, чтобы сбросить Data Toggle конечных точек до нуля. |
| D6 | **RxStall** | Когда принимается STALL, этот бит устанавливается и генерируется прерывание. ЦП должен сбросить этот бит. |
| D5 | **ReqPkt** | ЦП записывает 1 в этот бит для запроса IN транзакции. Он сбрасывается, когда устанавливается RxPktRdy. |
| D4 | **FlushFIFO** | ЦП записывает 1 в этот бит, чтобы очистить следующий пакет, который должен считаться из FIFO конечной точки Rx. Указатель FIFO и бит RxPktRdy (см. ниже) сбрасываются. *Примечание*: *FlushFIFO не используется если не установлен RxPktRdy. Также если FIFO имеет двойную буферизацию, возможно придется установить FlushFIFO дважды для полной очистки FIFO.* |
| D3 | **DataError/ NAK Timeout** | При работе в режиме ISO этот бит устанавливается, если RxPktRdy установлен, и, если пакет данных имеет ошибку CRC или Bit Stuff и сбрасывается, когда RxPktRdy сбрасывается. В поточном режиме этот бит будет установлен, когда конечная точка Rx останавливается в ожидании ответов NAK на время превышающее максимальное, установленное регистром RxInterval. ЦП должен сбросить этот бит, чтобы позволить конечной точке продолжить работу. |
| D2 | **Error** | USB устанавливает этот бит, когда было сделано 3 попытки получить пакет, и пакет данных не был получен. ЦП должен сбросить этот бит. Прерывание сгенерируется после установки бита. *Примечание*. *Этот бит используется только в том случае, когда конечная точка Tx работает в поточном режиме или режиме по прерываниям. В режиме ISO он всегда возвращает 0.* |
| D1 | **FIFOFull** | Этот бит устанавливается, когда в RX FIFO больше нельзя загрузить пакет. |
| D0 | **RxPktRdy** | Этот бит устанавливается, после принятия пакета данных. ЦП должен сбросить этот бит, когда пакет будет выгружен из FIFO точки RX. Прерывание сгенерируется, когда бит будет установлен. |

### 20.33.2.10 RXCOUNT

RxCount - это 13-разрядный регистр только для чтения, который содержит количество принятых байт данных в пакете из FIFO точки Rx. *Примечание*. *Возвращаемое значение изменяется по мере выгрузки FIFO и действительно только после установки RxPktRdy (RxCSR.D0).*

| | D12 | … | D0 |
|---|---|---|---|
| | ***(MSB)*** | **Endpoint Rx Count** | (LSB) |
| От ЦП | r | … | r |
| От USB | w | … | w |

### 20.33.2.11 TXTYPE (только хост режим)

TxType - это 8-разрядный регистр, в который должен быть записан номер конечной точки, с которой связана выбранная конечная точка, протокол передачи данных, который будет использоваться для выбранной конечной точки Tx, и ее рабочая скорость. Регистр TxType существует для каждой сконфигурированной конечной точки Tx (кроме конечной точки 0, которая имеет свой собственный Type регистр в 1Ah)

| | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | Speed | | Protocol | | **Target Endpoint Number** | | | |
| От ЦП | rw | rw | rw | rw | rw | rw | rw | rw |
| От USB | r | r | R | r | r | r | r | r |

| Бит | Имя | Функция |
|---|---|---|
| D7-D6 | Speed | Скорость работы целевого устройства:<br>00: Не используется<br>01: High<br>10: Full<br>11: Low |
| D5-D4 | Protocol | ЦП должен установить эти биты, чтобы выбрать требуемый протокол для конечной точки Tx:<br>00: Управляющий<br>01: Изохронный<br>10: Поточный<br>11: Прерывания |
| D3-D0 | **Target Endpoint Number** | ЦП должен установить в эти биты номер конечной точки, содержащийся в дескрипторе конечной точки Tx, вернувшейся в контроллер во время перечисления устройств. |

### 20.33.2.12 TXINTERVAL (только хост режим)

TxInterval - это 8-битный регистр, который определяет интервал опроса для текущей выбранной конечной точки Tx для передачи данных по прерываниям или изохронной передачи. Для поточных конечных точек этот регистр устанавливает количество кадров/микрокадров, после которых конечная точка должна приостановиться при получении потока ответов NAK. Регистр определен TxInterval для каждой сконфигурированной конечной точки Tx (кроме конечной точки 0).

| | D7 | … | D0 |
|---|---|---|---|
| | Tx Polling Interval/NAK Limit ( m) | | |
| От ЦП | rw | … | rw |
| От USB | w | … | r |

В каждом случае заданное значение определяет количество кадров /микрокадров (высокоскоростные передачи), как указано ниже

| Тип передачи данных | Скорость | Действительные значения (m) | Прерывание |
|---|---|---|---|
| Передача по прерыванию | Low-/full-speed | 1-225 | Интервал опроса - это m кадров. |
| | High-speed | 1-16 | Интервал опроса - $2^{(m-1)}$ микрокадра. |
| Изохронная передача | Full-/high-speed | 1-16 | Интервал опроса - $2^{(m-1)}$ кадра/микрокадра. |
| Поточная передача | Full-/high-speed | 2-16 | Предел NAK - это $2^{(m-1)}$ кадра/микрокадра *Примечание*. Значение 0 или 1 отключает функцию таймаута NAK. |

### 20.33.2.13 RXTYPE

RxType - это 8-разрядный регистр, в который должен быть записан номер конечной точки, на которую направлена выбранная конечная точка, протокол транзакции, который будет использоваться для текущей конечной точки Rx и ее рабочая скорость. Регистр RxType определен для каждой сконфигурированной конечной точки Rx (кроме конечной точки 0, которая имеет свой собственный регистр Type)

| | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | Speed | | Protocol | | **Target Endpoint Number** | | | |
| От ЦП | rw | rw | rw | Rw | rw | rw | Rw | rw |
| От USB | r | r | r | R | r | r | R | r |

| Бит | Имя | Функция |
|---|---|---|
| D7-D6 | Speed | Скорость работы целевого устройства:<br>00: Не используется<br>01: High<br>10: Full<br>11: Low |
| D5-D4 | Protocol | ЦП должен установить это, чтобы выбрать требуемый протокол для конечной точки Tx:<br>00: Управляющий<br>01: Изохронный<br>10: Поточный<br>11: По прерываниям |
| D3-D0 | **Target Endpoint Number** | ЦП должен установить в эти биты номер конечной точки, содержащийся в дескрипторе конечной точки Rx, возвращаемый в контроллер во время перечисления устройств. |

### 20.33.2.14 RXINTERVAL (только хост режим)

RxInterval - это 8-битный регистр, который определяет интервал опроса для только что выбранной конечной точки Rx в изохронной передаче или передаче по прерыванию. Для поточных конечных точек этот регистр устанавливает количество кадров/микрокадров, после которых конечная точка должна отключиться при получении потока ответов NAK. Для каждой сконфигурированной конечной точки Rx (кроме конечной точки 0) существует регистр RxInterval.

| | D7 | … | D4 |
|---|---|---|---|
| | **Rx Polling Interval/NAK Limit ( *m*)** | | |
| От ЦП | rw | … | rw |
| От USB | R | … | r |

В каждом случае заданное значение определяет количество кадров/микрокадров (высокоскоростные передачи), а именно:

| **Тип передачи данных** | **Скорость** | **Действительные значения (m)** | **Прерывание** |
|---|---|---|---|
| Передача по прерыванию | Low-/Full-speed | 1-225 | Интервал опроса - это m кадров. |
| | High-speed | 1-16 | Интервал опроса - $2^{(m-1)}$ микрокадра. |
| Изохронная передача | Full-/high-speed | 1-16 | Интервал опроса - $2^{(m-1)}$ кадра/микрокадра. |
| Поточная передача | Full-/high-speed | 2-16 | Предел NAK - это $2^{(m-1)}$ кадра/микрокадра *Примечание*. *Значение 0 или 1 отключает функцию таймаута NAK* |

### 20.33.2.15 FIFOSIZE

FIFOSize - это 8-разрядный регистр только для чтения, который возвращает размеры FIFO, связанные с выбранными дополнительными конечными точками Tx/Rx. Младшая половина байта кодирует размер FIFO выбранной конечной точки Tx; старшая половина байта кодирует размер FIFO выбранной конечной точки Rx. Значения n = 3 – 13 соответствуют размеру FIFO $2^n$ байтов (8 - 8192 байта). Если конечная точка не сконфигурирована, то будет отображаться значение 0. Если конечные точки Tx и Rx имеют один и тот же FIFO, размер Rx FIFO будет закодирован как 0xF.

Примечание – для этих целей регистр используется только в том случае, если регисистр Index установлен на одну из конечных точек 1-15. Также регистр имеет специальную интерпретацию, когда регистр Index установлен на конечную точку 0, и возвращаемый результат недействителен.

| | D7 | … | D4 | D3 | … | D0 |
|---|---|---|---|---|---|---|
| | **Rx FIFO Size** | | | **Tx FIFO Size** | | |
| От ЦП | r | … | r | r | … | r |
| От USB | r | … | r | r | … | r |

#### 20.33.2.16 FIFOx

Этот диапазон адресов предоставляет 16 адресов для доступа к ЦП для FIFO для каждой конечной точки. Запись по этим адресам загружает данные в TxFIFO для соответствующей конечной точки. Чтение по этим адресам выгружает данные из RxFIFO для соответствующей конечной точки.

*Примечание*: (I) Размер данных для передачи в и из FIFO может быть 8-битными, 16-битными или 32-битными, как потребуется, и любая комбинация доступа разрешена при условии, что доступ к данным является смежным. Тем не менее, все передачи, связанные с одним пакетом, должны иметь одинаковую ширину, чтобы данные были последовательно байтами, словами или двойными словами. Однако последняя передача может содержать меньше байт, чем предыдущие передачи, чтобы завершить передачу нечетного байта или нечетного слова.

(II) в зависимости от размера FIFO и ожидаемого максимального размера пакета, FIFO поддерживают либо однопакетную, либо двухпакетную буферизацию. Тем не менее, буферизация с большим количеством пакетов не поддерживается.

### 20.33.3 *Описание дополнительных контрольных/статусных регистров*

#### 20.33.3.1 TXFUNCADDR/ RXFUNCADDR (относится только к хост режиму)

TxFuncAddr и RxFuncAddr представляют собой 7-разрядные регистры чтения/записи, которые записывают адрес целевой функции, к которой необходимо получить доступ через соответствующую конечную точку.

| | D6 | … | D0 |
|---|---|---|---|
| | **Address of Target Function** | | |
| От ЦП | Rw | … | rw |
| От USB | R | … | r |

#### 20.33.3.2 TXHUBADDR /RXHUBADDR (только хост режим)

TxHubAddr и RxHubAddr - это 8-разрядные регистры чтения/записи, которые, как и TxHubPort и RxHubPort, должны быть установлены только в том случае, когда устройство full-/low-speed и подключено через высокоскоростной USB 2.0 хаб, который делает трансляцию транзакций между high-speed передачей и low-/full-speed передачей. Таким образом:

- младшие 7 бит должны хранить адрес этого USB 2.0 хаба;
- старший бит должен указывать на то, существует ли в хабе несколько конверторов транзакций (установлен на «0», если конвертор один и установлен на «1» если несколько).

| | D7 | D6 | … | D0 |
|---|---|---|---|---|
| | **Multiple Translators** | **Hub Address** | | |
| От ЦП | rw | rw | … | rw |
| От USB | r | r | … | r |

#### 20.33.3.3 TXHUBPORT/ RXHUBPORT (только хост режим)

В порты TxHubPort и RxHubPort нужно записывать, только если устройство high- или low-speed подключено через high-speed USB 2.0 хаб, который выполняет необходимую трансляцию транзакций. В таких обстоятельствах эти 7-битные регистры чтения/записи должны использоваться для записи номера порта хаба USB 2.0, через который осуществляется доступ к цели, ассоциируемой с конечной точкой.
*Address: x* 3/*x*Bh и *x*7/*x*Eh соответственно*; значение Reset:* 7'h00
(где *x* is 8 – F, в зависимости от конечной точки, к которой дается доступ)

| | D6 | … | D0 |
|---|---|---|---|
| | **Hub Port** | | |
| От ЦП | Rw | … | rw |
| От USB | R | … | r |

Эта информация вместе с данными адреса хаба, записанными выше, позволяет контроллеру поддерживать разделение транзакций.

## 20.34 Описание регистров контроллера SDIO SDIO_CNTR

Таблица 599 – Регистры контроллера SDIO

| **Базовый Адрес** | | **Название** | **Описание** |
|---|---|---|---|
| 0x400B_5000 | | SDIO_CNTR | |
| **Смещение** | | | |
| 0x0000 | 0 | SDIO_POWER | Регистр управления питанием |
| 0x0004 | 1 | SDIO_CLKCR | Регистр управление тактированием |
| 0x0008 | 2 | SDIO_ARG | Регистр аргумента SDIO |
| 0x000c | 3 | SDIO_CMD | Регистр команды |
| 0x0010 | 4 | SDIO_RESPCMD | Регистр ответов на команду |
| 0x0014 | 5 | SDIO_RESP0 | Регистр ответа 0 |
| 0x0018 | 6 | SDIO_RESP1 | Регистр ответа 1 |
| 0x001c | 7 | SDIO_RESP2 | Регистр ответа 2 |
| 0x0020 | 8 | SDIO_RESP3 | Регистр ответа 3 |
| 0x0024 | 9 | SDIO_DTIMER | Регистр таймера данных SDIO |
| 0x0028 | 10 | SDIO_DLEN | Регистр длины данных SDIO |
| 0x002c | 11 | SDIO_DCNTR | Регистр управления данными SDIO |
| 0x0030 | 12 | SDIO_DCOUNT | Регистр счетчика данных SDIO |
| 0x0034 | 13 | SDIO_STA | Регистр статуса SDIO |
| 0x0038 | 14 | SDIO_ICR | Регистр сброса прерывания SDIO |
| 0x003c | 15 | SDIO_MASK | Регистр маски SDIO |
| 0x0048 | 16 | SDIO_FIFOCNT | Регистр счетчика FIFO |
| 0x0080 | 17 | SDIO_FIFO | Регистр данных FIFO |

### 20.34.1 SDIO_POWER

Таблица 600 – Описание бит регистра *SDIO_POWER*

| Base ADDR= | | | **0x400B_5000** | | | Offset= | | **0x0000_0000** | | | Reset= | | 0x0000_0000 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | SDIO_POWER | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Зарезервировано | | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | | | | | | | | | | | PWRCNTR [1:0] | |
| | | | | | | | | | | | | | | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…2 | Зарезервировано | | Резерв |
| 1…0 | PWRCNTR [1:0] | 0 | Данные биты определяют текущий статус тактирования карты:<br>00: Выключено: тактирование карты остановлено;<br>01: Зарезервировано;<br>10: Зарезервировано включение;<br>11: Включено: тактирование карты активно |

### 20.34.2 *SDIO_CLKCR*

Регистр SDIO_CLKCR управляет выходным тактовым сигналом SDIO_CK.

Таблица 601 – Описание бит регистра *SDIO_CLKCR*

| Base ADDR= | **0x400B_5000** | Offset= | **0x0000_0004** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | SDIO_CLKCR | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | HWFC_EN | NEGEDGE | WIDBUS [1] | WIDBUS [0] | BYPASS | PWRSAV | CLKEN | CLKDIV [7:0] | | | | | | | |
| | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…15 | Зарезервировано | - | Резерв |
| 14 | HWFC_EN | 0 | Включение аппаратного управления потоком передачи данных (HW Flow Control)<br>0: выключено<br>1: включено |
| 13 | NEGEDGE | 0 | Дефазирование сигнала SDIO_CK<br>0: Команда и данные изменены во время заднего фронта сигнала SDIOCLK, после чего следует передний фронт сигнала SDIO_CK. (Передний фронт сигнала SDIO_CK возникает при переднем фронте SDIOCLK).<br>1: Команда и данные изменены во время заднего фронта сигнала SDIO_CK.<br><br>Если бит BYPASS активен, команда и данные изменяются при заднем фронте сигнала SDIOCLK независимо от значения бита NEGEDGE |

| | | | |
|---|---|---|---|
| 12…11 | WIDBUS [1:0] | 0 | Включение режима работы широкой шины<br>00: Режим по умолчанию: используется SDIO_D0<br>01: Режим шины 4-бит: используется SDIO_D [3:0]<br>10: Режим шины 8-бит: используется SDIO_D [7:0] |
| 10 | BYPASS | 0 | Режим bypass для делителя частоты<br>0: Режим bypass выключен: SDIOCLK делится в соответствии со значением CLKDIV перед передачей сигнала SDIO_CK.<br>1: Режим bypass включен: SDIOCLK напрямую передает сигнал SDIO_CK |
| 9 | PWRSAV | 0 | Конфигурация экономии энергии<br>В целях экономии энергии можно отключить выход тактирования SDIO_CK, если шина находится в состоянии ожидания, путем установки бита PWRSAV:<br>0: SDIO_CK всегда включен<br>1: SDIO_CK включен только когда шина активна |
| 8 | CLKEN | 0 | Включение тактирования<br>0: SDIO_CK выключен<br>1: SDIO_CK включен |
| 7…0 | CLKDIV [7:0] | 0 | Коэффициент делителя частоты<br>Данное поле определяет коэффициент делителя частоты между входом тактирования (SDIOCLK) и выходом тактирования (SDIO_CK): Частота SDIO_CK = SDIOCLK / [CLKDIV + 2] |

### 20.34.3 *SDIO_ARG*

Регистр SDIO_ARG содержит 32-битный аргумент команды, который отправляется на карту в качестве части командного сообщения.

Таблица 602 – Описание бит регистра *SDIO_ARG*

| Base ADDR= | **0x400B_5000** | Offset= | **0x0000_0008** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | SDIO_ARG | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CMDARG [31:16] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CMDARG [15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | CMDARG | 0 | Аргумент команды<br>Аргумент команды отправляется на карту как часть командного сообщения. Если команда содержит аргумент, он должен быть загружен в данный регистр перед записью команды в регистр команды |

### 20.34.4 *SDIO_CMD*

Регистр SDIO_CMD содержит индекс команды и тип команды. Индекс команды отправляется на карту в составе командного сообщения. Биты типа команды управляют конечным автоматом команд (command path state machine, CPSM).

Таблица 603 – Описание бит регистра *SDIO_CMD*

| Base ADDR= | **0x400B_5000** | Offset= | **0x0000_000C** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | SDIO_CMD | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | SDIO Suspend | CPSMEN | WAITPEND | WAITINT | WAITRESP [1:0] | | CMDINDEX [5:0] | | | | | |
| | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…12 | Зарезервировано | | Резерв |
| 11 | SDIOSuspend | 0 | Команда приостановки SD I/O<br>Если данный бит установлен, то посылается команда приостановки (используется только для карт SDIO) |
| 10 | CPSMEN | 0 | Включение конечного автомата команд (CPSM)<br>Если данный бит установлен, конечный автомат команд включен |
| 9 | WAITPEND | 0 | Конечный автомат команд ожидает окончание передачи данных (внутренний сигнал CmdPend).<br>Если данный бит установлен, конечный автомат команд ожидает окончание передачи данных перед тем, как он начнет отправку команды. Данная функция доступна только в потоковом режиме передачи данных SDIO_DCNTR [2] = 1 |
| 8 | WAITINT | 0 | Конечный автомат команд ожидает запрос прерывания<br>Если данный бит установлен, конечный автомат команд (CPSM) сбрасывает время ожидания команды и ждет запрос на прерывание |
| 7…6 | WAITRESP | 0 | Биты ожидания ответа<br>Биты ожидания ответа используются для настройки того, должен ли конечный автомат команд ожидать ответа, и если да, то какого типа ответа.<br>00: нет ответа, ожидание флага CMDSENT<br>01: короткий ответ, ожидание флага CMDREND или CCRCFAIL<br>10: нет ответа, ожидание флага CMDSENT<br>11: длинный ответ, ожидание флага CMDREND или CCRCFAIL |
| 5…0 | CMDINDEX | 0 | Индекс команды<br>Индекс команды отправляется на карту в качестве части командного сообщения |

### 20.34.5 *SDIO_RESPCMD*

Регистр SDIO_RESPCMD содержит поле индекса команды последнего полученного ответа на команду. Если при передаче ответного сообщения, оно не содержит поле индекса (длинный ответ или ответ OCR), поле RESPCMD неизвестно, хотя оно должно содержать значение 111111b (значение зарезервированного поля из ответного сообщения).

Таблица 604 – Описание бит регистра *SDIO_RESPCMD*

| Base ADDR= | **0x400B_5000** | Offset= | **0x0000_0010** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | SDIO_PESPCMD | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | | | | | | | RESPCMD [5:0] | | | | | |
| | | | | | | | | | | r | r | r | r | r | r |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…6 | Зарезервировано | | Резерв |
| 5…0 | RESPCMD [5:0] | 0 | Индекс команды ответа<br>Только для чтения.<br>Содержит индекс команды последнего полученного ответа на команду |

### 20.34.6 *SDIO_RESPx*

Регистры SDIO_RESP1/2/3/4 содержат статус карты, который является частью полученного ответного сообщения.

Таблица 605 – Описание бит регистров *SDIO_RESPx*

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Base ADDR= | | | **0x400B_5000** | | | Offset= | | **0x0000_0014** | | | Reset= | | 0x0000_0000 | | |
| | | | | | | Offset= | | **0x0000_0018** | | | Reset= | | 0x0000_0000 | | |
| | | | | | | Offset= | | **0x0000_001C** | | | Reset= | | 0x0000_0000 | | |
| | | | | | | Offset= | | **0x0000_0020** | | | Reset= | | 0x0000_0000 | | |
| REG Name: | | | SDIO_PESP0 | | | | | | | | | | | | |
| | | | SDIO_PESP1 | | | | | | | | | | | | |
| | | | SDIO_PESP2 | | | | | | | | | | | | |
| | | | SDIO_PESP3 | | | | | | | | | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CARDSTATUSx [31:16] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CARDSTATUSx [15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | CARDSTATUS [31:0] | 0 | Статус карты |

Статус карты имеет размер 32 или 127 бит в зависимости от типа ответа.

Старший значащий бит (MSB) статуса карты идет первым. Младший значащий бит (LSB) регистра SDIO_RESP3 всегда равен 0b.

Таблица 606 – Тип ответа и регистры *SDIO_RESPx*

| Регистр | Короткий ответ | Длинный ответ |
|---|---|---|
| SDIO RESP0 | Статус карты [31:0] | Статус карты [127:96] |
| SDIO RESP1 | Не используется | Статус карты [95:64] |
| SDIO RESP2 | Не используется | Статус карты [63:32] |
| SDIO RESP3 | Не используется | Статус карты [31:1]0b |

### 20.34.7 *SDIO_DTIMER*

Регистр SDIO_DTIMER содержит значение времени ожидания данных в виде числа тактовых сигналов карты.

Счетчик загружает значение из регистра SDIO_DTIMER и начинает уменьшаться, когда конечный автомат с каналом данных (data path state machine, DPSM) переходит в состояние Wait_R или Busy (занят). Когда таймер достигает значения 0, пока конечный автомат с каналом данных находится в одном из указанных состояний, устанавливается флаг статуса времени ожидания.

Таблица 607 – Описание бит регистра *SDIO_DTIMER*

| Base ADDR= | **0x400B_5000** | Offset= | **0x0000_0024** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | SDIO_DTIMER | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATATIME [31:16] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATATIME [15:0] | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | DATATIME [1:0] | 0 | Время ожидания данных<br>Время ожидания данных выражается в виде тактовых сигналов карты |

### 20.34.8 *SDIO_DLEN*

Регистр SDIO_DLEN содержит число байт данных для передачи. Значение регистра загружается в счетчик данных при начале передачи данных.

Таблица 608 – Описание бит регистров *SDIO_DLEN*

| Base ADDR= | **0x400B_5000** | Offset= | **0x0000_0028** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | SDIO_DLEN | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | | | | DATALENGTH [24:16] | | | | | | | | |
| | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATALENGTH [15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…25 | Зарезервировано | | Резерв. |
| 24…0 | DATALENGTH [24:0] | 0 | Число байт данных для передачи |

Для поблочной передачи данных, значение регистра длины данных должно быть кратно размеру блока (см. описание регистра SDIO_DCNTR). Перед записью в управляющий регистр данных время ожидания должно быть записано в регистр таймера и регистр длины данных:

- если выбран режим потоковой передачи данных или многобайтовой передачи данных SDIO, значение в регистре длины данных должно быть между 1 и 512.
- если выбрана поблочная передача данных, значение в регистре длины данных должно лежать в диапазоне между 1 x Размер блока данных и 512 x Размер блока данных.

### 20.34.9 *SDIO_DCNTR*

Регистр SDIO_DCNTR управляет конечным автоматом с каналом данных (data path state machine, DPSM).

Таблица 609 – Описание бит регистра *SDIO_DCNTR*

| Base ADDR= | | | | **0x400B_5000** | | | | Offset= | | | | **0x0000_002C** | | | | Reset= | | | | 0x0000_0000 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | | SDIO_DCNTR | | | | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | | | | | | | | |
| Зарезервировано | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | SDIOEN | RWMOD | RWSTOP | RWSTART | DBLOCKSIZE [7:4] | | | | DMAEN | DTMODE | DTDIR | DTEN |
| | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…12 | Зарезервировано | | Резерв. |
| 11 | SDIOEN | 0 | Включение функций интерфейса SDIO<br>Если данный бит установлен, конечный автомат с каналом данных (DPSM) взаимодействует с картой SDIO |
| 10 | RWMOD | 0 | Режим ожидания чтения (Read Wait)<br>0: Режим ожидания чтения останавливает SDIO_D2<br>1: Режим ожидания чтения использует SDIO_CK |
| 9 | RWSTOP | 0 | Остановка режима ожидания чтения<br>0: Режим ожидания чтения запущен, если установлен бит RWSTART<br>1: Остановка режима ожидания чтения, если бит RWSTART установлен |
| 8 | RWSTART | 0 | Запуск режима ожидания чтения<br>Если данный бит установлен, режим ожидания чтения запущен |
| 7…4 | DBLOCKSIZE [7:4] | 0 | Длина блока данных определяется при выборе режима передачи данных:<br>0000: 1 байт<br>0001: 2 байта<br>0010: 4 байта<br>0011: 8 байт<br>…<br>1110: 16384 байта<br>1111: зарезервировано |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 3 | DMAEN | 0 | Бит разрешения DMA<br>0: DMA запрещен.<br>1: DMA разрешен |
| 2 | DTMODE | 0 | Бит выбора режима передачи:<br>0: Передача блока данных<br>1: Потоковая или многобайтовая передача данных SDIO |
| 1 | DTDIR | 0 | Бит выбора направления передачи данных<br>0: от контроллера на карту<br>1: с карты на контроллер |
| 0 | DTEN | 0 | Бит разрешения передачи данных<br>Передача данных начинается, если в бит DTEN записано значение 1. В зависимости от значения бита DTDIR, конечный автомат с каналом данных переходит в состояние Wait_S, Wait_R или Readwait, если бит RW Start установлен сразу в начале передачи. Нет необходимости сбрасывать бит разрешения по окончании передачи данных, но необходимо обновить регистр SDIO_DCNTR для разрешения передачи новых данных |

Значение бита DTMODE изменяется в соответствии со значением бита SDIOEN. Если SDIOEN=0 и DTMODE=1, включается режим передачи MultiMediaCard, а если SDIOEN=1 и DTMODE=1, периферийные устройства разрешают многобайтовую передачу данных по SDIO.

### 20.34.10 *SDIO_DCOUNT*

Регистр SDIO_DCOUNT загружает значение из регистра длины данных (см. описание регистра SDIO_DLEN), когда конечный автомат с каналом данных переходит из состояния ожидания (Idle) в состояние Wait_R или Wait_S. По ходу передачи данных, счетчик уменьшает значение до тех пор, пока не достигнет 0. После чего конечный автомат с каналом данных переходит в состояние ожидания, и устанавливается флаг статуса окончания данных, DATAEND.

Таблица 610 – Описание бит регистров *SDIO_DCOUNT*

| Base ADDR= | **0x400B_5000** | Offset= | **0x0000_0030** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | SDIO_DCOUNT | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | | | | DATACOUNT [24:16] | | | | | | | | |
| | | | | | | | r | r | r | r | r | r | r | r | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATACOUNT [15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…25 | Зарезервировано | | Резерв |
| 24…0 | DATACOUNT [24:0] | 0 | Значение счетчика данных<br>При чтении данного поля, возвращается число оставшихся байт данных для передачи. Запись не имеет значения |

### 20.34.11 *SDIO_STA*

Регистр SDIO_STA доступен только для чтения. Он содержит два типа флагов:

- СТАТИЧЕСКИЕ флаги (биты [23:22], [10:0]): данные биты остаются установленными до тех пор, пока они не будут сброшены записью в регистр сброса прерывания SDIO (см. SDIO_ICR);
- динамические флаги (биты [21:11]): данные биты изменяют состояние в зависимости от состояния лежащей в основе логики (например, флаги заполнения и опустошения FIFO буфера устанавливаются и снимаются по мере записи данных в FIFO буфер).

Таблица 611 – Описание бит регистра *SDIO_STA*

| Base ADDR= | **0x400B_5000** | Offset= | **0x0000_0034** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | SDIO_STA | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | | | | | | SDIOIT | RXDAVL | TXDAVL | RXFIFOE | TXFIFOE | RXFIFOF | TXFIFOF |
| | | | | | | | | | r | r | r | r | r | r | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RXFIFOHF | TXFIFOHE | RXACT | TXACT | CMDACT | DBCKEND | Зарезервировано | DATAEND | CMDSENT | CMDREND | RXOVERR | TXUNDERR | DTIMEOUT | CTIMEOUT | DCRCFAIL | CCRCFAIL |
| r | r | r | r | r | r | | r | r | r | r | r | r | r | r | r |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…23 | Зарезервировано | | Резерв. |
| 22 | SDIOIT | | Получен запрос на прерывание SDIO |
| 21 | RXDAVL | | Есть данные в FIFO буфере приемника |
| 20 | TXDAVL | | Есть данные в FIFO буфере передатчика |
| 19 | RXFIFOE | | Буфер FIFO приемника пуст |
| 18 | TXFIFOE | | Буфер FIFO передатчика пуст<br>Если аппаратное управление потоком данных включено, сигналы TXFIFOE активируются, когда FIFO буфер содержит 2 слова. |
| 17 | RXFIFOF | | Буфер FIFO приемника полон<br>Если аппаратное управление потоком данных включено, сигналы RXFIFOF активируются за 2 слова до того, как буфер FIFO заполнен. |
| 16 | TXFIFOF | | Буфер FIFO передатчика полон |
| 15 | RXFIFOHF | | Буфер FIFO приемника наполовину полон: буфер FIFO содержит как минимум 8 слов |
| 14 | TXFIFOHE | | Буфер FIFO передатчика наполовину пуст: как минимум 8 слов могут быть записаны в буфер FIFO |
| 13 | RXACT | | Прием данных в процессе |
| 12 | TXACT | | Передача данных в процессе |
| 11 | CMDACT | | Передача команды в процессе |
| 10 | DBCKEND | | Блок данных передан/получен (проверка CRC выполнена успешно) |
| 9 | Зарезервировано | | Резерв. |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 8 | DATAEND | | Конец данных (счетчик данных, SDIDCOUNT, равен нулю) |
| 7 | CMDSENT | | Команда отправлена (ответ не требуется) |
| 6 | CMDREND | | Ответ на команду получен (проверка CRC выполнена успешно) |
| 5 | RXOVERR | | Ошибка из-за переполнения FIFO буфера приемника<br>*Примечание: Если используется DMA для чтения SDIO FIFO (бит DMAEN установлен в регистре SDIO_DCNTR), пользовательское программное обеспечение должно отключить поток DMA, и затем записать '0' (для отключения запросов DMA).* |
| 4 | TXUNDERR | | Ошибка из-за не заполнения FIFO буфера передатчика<br>*Примечание: Если DMA используется для заполнения SDIO FIFO (бит DMAEN установлен в регистре SDIO_DCNTR), пользовательское программное обеспечение должно отключить поток DMA, и затем установить бит DMAEN в '0' (для отключения запросов DMA).* |
| 3 | DTIMEOUT | | Время ожидания данных |
| 2 | CTIMEOUT | | Время ожидания ответа на команду<br>Время ожидания ответа на команду имеет фиксированное значение – 64 такта SDIO_CK. |
| 1 | DCRCFAIL | | Блок данных отправлен/получен (проверка CRC не выполнена) |
| 0 | CCRCFAIL | | Ответ на команду получен (проверка CRC не выполнена) |

### 20.34.12 *SDIO_ICR*

Регистр SDIO_ICR доступен только по записи. Запись 1b сбрасывает соответствующий бит в регистре статуса SDIO_STA.

Таблица 612 – Описание бит регистров *SDIO_ICR*

| Base ADDR= | **0x400B_5000** | Offset= | **0x0000_0038** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | SDIO_ICR | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | | | | | | SDIOITC | Зарезервировано | | | | | |
| | | | | | | | | | rw | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | | DBCKENDC | Зарезервировано | DATAENDC | CMDSENTC | CMDRENDC | RXOVERRC | TXUNDERRC | DTIMEOUTC | CTIMEOUTC | DCRCFAILC | CCRCFAILC |
| | | | | | rw | | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…23 | Зарезервировано | | Резерв |
| 22 | SDIOITC | 0 | Сброс флага SDIOIT<br>Устанавливается программно для сброса флага SDIOIT.<br>0: SDIOIT не сброшен<br>1: SDIOIT сброшен |
| 21…11 | Зарезервировано | | Резерв |
| 10 | DBCKENDC | 0 | Сброс флага DBCKEND<br>Устанавливается программно для сброса флага DBCKEND.<br>0: DBCKEND не сброшен<br>1: DBCKEND сброшен |
| 9 | Зарезервировано | | Резерв |
| 8 | DATAENDC | 0 | Сброс флага DATAEND<br>Устанавливается программно для сброса флага DATAEND.<br>0: DATAEND не сброшен<br>1: DATAEND сброшен |
| 7 | CMDSENTC | 0 | Сброс флага CMDSENT<br>Устанавливается программно для сброса флага CMDSENT.<br>0: CMDSENT не сброшен<br>1: CMDSENT сброшен |
| 6 | CMDRENDC | 0 | Сброс флага CMDREND<br>Устанавливается программно для сброса флага CMDREND.<br>0: CMDREND не сброшен<br>1: CMDREND сброшен |
| 5 | RXOVERRC | 0 | Сброс флага RXOVERR<br>Устанавливается программно для сброса флага RXOVERR.<br>0: RXOVERR не сброшен<br>1: RXOVERR сброшен |
| 4 | TXUNDERRC | 0 | Сброс флага TXUNDERR<br>Устанавливается программно для сброса флага TXUNDERR.<br>0: TXUNDERR не сброшен<br>1: TXUNDERR сброшен |
| 3 | DTIMEOUTC | 0 | Сброс флага DTIMEOUT<br>Устанавливается программно для сброса флага DTIMEOUT.<br>0: DTIMEOUT не сброшен<br>1: DTIMEOUT сброшен |
| 2 | CTIMEOUTC | 0 | Сброс флага CTIMEOUT<br>Устанавливается программно для сброса флага CTIMEOUT.<br>0: CTIMEOUT не сброшен<br>1: CTIMEOUT сброшен |
| 1 | DCRCFAILC | 0 | Сброс флага DCRCFAIL<br>Устанавливается программно для сброса флага DCRCFAIL.<br>0: DCRCFAIL не сброшен<br>1: DCRCFAIL сброшен |
| 0 | CCRCFAILC | 0 | Сброс флага CCRCFAIL<br>Устанавливается программно для сброса флага CCRCFAIL.<br>0: CCRCFAIL не сброшен<br>1: CCRCFAIL сброшен |

Примечание – Для организации надежного сброса статусных флагов прерываний необходимо учитывать требования, изложенные в пункте 22.1.2 «Обработчики исключений».

### 20.34.13 *SDIO_MASK*

Регистр маски прерывания определяет какие флаги статуса генерируют запрос прерывания путем установки соответствующего бита в значение 1b.

Таблица 613 – Описание бит регистров *SDIO_MASK*

| Base ADDR= | **0x400B_5000** | Offset= | **0x0000_003C** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | SDIO_MASK | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | | | | | | SDIOITIE | RXDAVLE | TXDAVLE | RXFIFOEE | TXFIFOEE | RXFIFOFE | TXFIFOFE |
| | | | | | | | | | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RXFIFOHFIE | TXFIFOHEE | RXACTE | TXACTE | CMDACTE | DBCKENDE | Зарезервировано | DATAENDE | CMDSENTE | CMDRENDE | RXOVERRE | TXUNDERRE | DTIMEOUTE | CTIMEOUTE | DCRCFAILE | CCRCFAILE |
| rw | rw | rw | rw | rw | rw | | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…23 | Зарезервировано | | Резерв |
| 22 | SDIOITIE | 0 | Разрешение прерывания по запросу прерывания в режиме SDIO<br>Устанавливается и сбрасывается программно для разрешения/запрета прерывания при получении запроса на прерывание режима SDIO.<br>0: прерывание запрещено<br>1: прерывание разрешено |
| 21 | RXDAVLIE | 0 | Разрешение прерывания по наличию данных в буфере FIFO приемника<br>Устанавливается и сбрасывается программно для разрешения/запрета прерывания по наличию данных в буфере FIFO приемника.<br>0: прерывание запрещено<br>1: прерывание разрешено |
| 20 | TXDAVLIE | 0 | Разрешение прерывания по наличию данных в буфере FIFO передатчика<br>Устанавливается и сбрасывается программно для разрешения/запрета прерывания по наличию данных в буфере FIFO передатчика.<br>0: прерывание запрещено<br>1: прерывание разрешено |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 19 | RXFIFOEIE | 0 | Разрешение прерывания по отсутствию данных в буфере FIFO приемника<br>Устанавливается и сбрасывается программно для разрешения/запрета прерывания, вызванного отсутствием данных в буфере FIFO приемника.<br>0: прерывание запрещено<br>1: прерывание разрешено |
| 18 | TXFIFOEIE | 0 | Разрешение прерывания по отсутствию данных в буфере FIFO передатчика<br>Устанавливается и сбрасывается программно для разрешения/запрета прерывания, вызванного отсутствием данных в буфере FIFO передатчика.<br>0: прерывание запрещено<br>1: прерывание разрешено |
| 17 | RXFIFOFIE | 0 | Разрешение прерывания при полном буфере FIFO приемника<br>Устанавливается и сбрасывается программно для разрешения/запрета прерывания при полном буфере FIFO приемника.<br>0: прерывание запрещено<br>1: прерывание разрешено |
| 16 | TXFIFOFIE | 0 | Разрешение прерывания при полном буфере FIFO передатчика<br>Устанавливается и сбрасывается программно для разрешения/запрета прерывания при полном буфере FIFO передатчика.<br>0: прерывание запрещено<br>1: прерывание разрешено |
| 15 | RXFIFOHFIE | 0 | Разрешение прерывания при наполовину полном буфере FIFO приемника<br>Устанавливается и сбрасывается программно для разрешения/запрета прерывания при наполовину полном буфере FIFO приемника.<br>0: прерывание запрещено<br>1: прерывание разрешено |
| 14 | TXFIFOHEIE | 0 | Разрешение прерывания при наполовину пустом буфере FIFO передатчика<br>Устанавливается и сбрасывается программно для разрешения/запрета прерывания при наполовину пустом буфере FIFO передатчика.<br>0: прерывание запрещено<br>1: прерывание разрешено |
| 13 | RXACTIE | 0 | Разрешение прерывания при получении данных<br>Устанавливается и сбрасывается программно для разрешения/запрета прерывания при получении данных (получение данных в процессе).<br>0: прерывание запрещено<br>1: прерывание разрешено |
| 12 | TXACTIE | 0 | Разрешение прерывания при передаче данных<br>Устанавливается и сбрасывается программно для разрешения/запрета прерывания при передаче данных (передача данных в процессе).<br>0: прерывание запрещено<br>1: прерывание разрешено |
| 11 | CMDACTIE | 0 | Разрешение прерывания при передаче команды<br>Устанавливается и сбрасывается программно для разрешения/запрета прерывания при передаче команды (передача команды в процессе).<br>0: прерывание запрещено<br>1: прерывание разрешено |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 10 | DBCKENDIE | 0 | Разрешение прерывания при завершении передачи блока данных<br>Устанавливается и сбрасывается программно для разрешения/запрета прерывания по завершению передачи блока данных.<br>0: прерывание запрещено<br>1: прерывание разрешено |
| 9 | Зарезервировано | | Резерв. |
| 8 | DATAENDIE | 0 | Разрешение прерывания по окончании данных<br>Устанавливается и сбрасывается программно для разрешения/запрета прерывания по окончании данных.<br>0: прерывание запрещено<br>1: прерывание разрешено |
| 7 | CMDSENTIE | 0 | Разрешение прерывания по отправке команды<br>Устанавливается и сбрасывается программно для разрешения/запрета прерывания по отправке команды.<br>0: прерывание запрещено<br>1: прерывание разрешено |
| 6 | CMDRENDIE | 0 | Разрешение прерывания при получении ответа на команду<br>Устанавливается и сбрасывается программно для разрешения/запрета прерывания при получении ответа на команду.<br>0: прерывание запрещено<br>1: прерывание разрешено |
| 5 | RXOVERRIE | 0 | Разрешение прерывания при переполнении буфера FIFO приемника<br>Устанавливается и сбрасывается программно для разрешения/запрета прерывания, вызванного переполнением буфера FIFO приемника.<br>0: прерывание запрещено<br>1: прерывание разрешено |
| 4 | TXUNDERRIE | 0 | Разрешение прерывания при недостаточности данных в буфере FIFO передатчика<br>Устанавливается и сбрасывается программно для разрешения/запрета прерывания, вызванного недостаточным количеством данных в буфере FIFO передатчика.<br>0: прерывание запрещено<br>1: прерывание разрешено |
| 3 | DTIMEOUTIE | 0 | Разрешение прерывания по превышению времени ожидания данных<br>Устанавливается и сбрасывается программно для разрешения/запрета прерывания, вызванного временем ожидания данных.<br>0: прерывание запрещено<br>1: прерывание разрешено |
| 2 | CTIMEOUTIE | 0 | Разрешение прерывания по превышению времени ожидания команды<br>Устанавливается и сбрасывается программно для разрешения/запрета прерывания, вызванного временем ожидания команды.<br>0: прерывание запрещено<br>1: прерывание разрешено |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 1 | DCRCFAILIE | 0 | Разрешение прерывания по ошибке проверки CRC данных.<br>Устанавливается и сбрасывается программно для разрешения/запрета прерывания, вызванного ошибкой проверки CRC данных.<br>0: прерывание запрещено<br>1: прерывание разрешено |
| 0 | CCRCFAILIE | 0 | Разрешение прерывания по ошибке проверки CRC команды<br>Устанавливается и сбрасывается программно для разрешения/запрета прерывания, вызванного ошибкой проверки CRC команды.<br>0: прерывание запрещено<br>1: прерывание разрешено |

### 20.34.14 *SDIO_FIFOCNT*

Регистр SDIO_FIFOCNT содержит оставшееся число слов для записи или чтения в/из буфера FIFO. Счетчик FIFO буфера загружает значение из регистра длины данных (см. описание регистра SDIO_DLEN), если бит DTEN установлен в управляющем регистре данных (регистр SDIO_DCNTR) и конечный автомат с каналом данных (DPSM) находится в состоянии ожидания. Если длина данных не выровнена по слову (кратно 4), оставшиеся байты с 1 по 3 рассматриваются как слово.

Таблица 614 – Описание бит регистров *SDIO_FIFOCNT*

| Base ADDR= | **0x400B_5000** | Offset= | **0x0000_0048** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | SDIO_FIFOCNT | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | | | | | FIFOCOUNT [23:16] | | | | | | | |
| | | | | | | | | r | r | r | r | r | r | r | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FIFOCOUNT [15:0] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…24 | Зарезервировано | | Резерв |
| 23…0 | FIFOCOUNT [23:0] | 0 | Оставшееся число слов для записи или чтения в/из буфера FIFO |

### 20.34.15 *SDIO_FIFO*

Буферы FIFO приемника и передатчика доступны по чтению и записи как 32-битные регистры. Буферы FIFO содержат 32 записи по 32 последовательным адресам. Это позволяет центральному процессору использовать множество операндов загрузки и сохранения для чтения и записи FIFO буферов.

Таблица 615 – Описание бит регистров *SDIO_FIFO*

| Base ADDR= | **0x400B_5000** | Offset= | **0x0000_0080** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | SDIO_FIFO | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| FIFODATA [31:16] | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FIFODATA [15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | FIFODATA [31:0] | | Данные FIFO буферов приемника и передатчика<br>Данные FIFO буфера занимают 32 записи по 32-битных слова, начиная с адреса: SDIO base + 0x080 до SDIO base + 0xFC |

## 20.35 Описание регистров контроллера CRC CRC_CNTR

Таблица 616 – Описание регистров контроллера CRC

| Базовый Адрес | Название | Состояние после сброса | Описание |
|---|---|---|---|
| 0x400B_6000 | CRC | | Контроллер CRC |
| **Смещение** | | | |
| 0x00 | DR | 0xFFFF_FFFF | Регистр входных/выходных данных |
| 0x04 | IDR | 0x0000_0000 | Регистр общего назначения |
| 0x08 | CR | 0x0000_0000 | Регистр управления |
| 0x10 | INIT | 0xFFFF_FFFF | Регистр начального значения CRC |
| 0x14 | POL | 0x04C1_1DB7 | Регистр полинома |

### 20.35.1 *DR*

Таблица 617 – Описание бит регистра DR

| Base ADDR= | 0x400B_6000 | Offset= | 0x0000_0000 | Reset= | 0xFFFF_FFFF |
|---|---|---|---|---|---|
| REG Name: | DR | | | | |
| 31...0 | | | | | |
| DR[31:0] | | | | | |
| R/W | | | | | |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31…0 | DR | RW | 0xFFFFFFFF | Биты регистра данных.<br>Регистр используется для записи новых данных в калькулятор CRC.<br>При чтении он содержит предыдущий результат вычисления CRC.<br>Если размер данных меньше 32 бит, то в операциях записи / чтения используются младшие значащие биты. |

### 20.35.2 *IDR*

Таблица 618 – Описание бит регистра IDR

| Base ADDR= | 0x400B_6000 | Offset= | 0x0000_0004 | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | IDR | | | | |

| 31...8 | 7…0 |
|---|---|
| - | IDR[7:0] |
| - | R/W |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31…8 | - | - | 0x000000 | Зарезервировано. |
| 7…0 | IDR | RW | 0x00 | 8 бит данных общего назначения<br>Эти биты могут использоваться для временного хранения одного байта.<br>Регистр не сбрасывается битом RESET регистра CR. |

### 20.35.3 *CR*

Таблица 619 – Описание бит регистра CR

| Base ADDR= | 0x400B_6000 | Offset= | 0x0000_0008 | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | CR | | | | |

| 31...8 | 7 | 6 | 5 | 4 | 3 | 2...1 | 0 |
|---|---|---|---|---|---|---|---|
| - | REV_OUT | REV_IN[1:0] | | POLYSIZE[1:0] | | - | RESET |
| - | R/W | R/W | | R/W | | - | R/SET |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31…8 | - | - | 0x000000 | Зарезервировано. |
| 7 | REV_OUT | R/W | 0 | Реверс выходных данных:<br>0 – нормальный порядок бит;<br>1 – побитовый реверс. |
| 6…5 | REV_IN | R/W | 2'b00 | Реверс входных данных:<br>00 – нормальный порядок бит;<br>01 – реверс по 8 бит;<br>10 – реверс по 16 бит;<br>11 – реверс по 32 бита. |
| 4…3 | POLYSIZE | R/W | 2'b00 | Размер полинома:<br>00 – 32-битный полином;<br>01 – 16-битный полином;<br>10 – 8-битный полином;<br>11 – 7-битный полином. |
| 2…1 | - | - | 0 | Зарезервировано. |
| 0 | RESET | R/SET | 0 | Бит сброса вычислений блока CRC.<br>Устанавливает в регистр данных DR знечение регистра INIT.<br>Возможна только установка в 1.<br>Самоочищающийся бит. |

### 20.35.4 *INIT*

Таблица 620 – Описание бит регистра INIT

| Base ADDR= | 0x400B_6000 | Offset= | 0x0000_0010 | Reset= | 0xFFFF_FFFF |
|---|---|---|---|---|---|
| REG Name: | INIT | | | | |

| 31...0 |
|---|
| INIT[31:0] |
| R/W |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31…0 | INIT | RW | 0xFFFFFFFF | Первоначальное значение CRC. |

### 20.35.5 *POL*

Таблица 621 – Описание бит регистра POL

| Base ADDR= | 0x400B_6000 | Offset= | 0x0000_0014 | Reset= | 0x04C1_1DB7 |
|---|---|---|---|---|---|
| REG Name: | POL | | | | |
| 31...0 | | | | | |
| POL[31:0] | | | | | |
| R/W | | | | | |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31…0 | POL | RW | 0x04C1_1DB7 | Программируемый полином для расчёта CRC. Если размер полинома меньше 32 бит, то для программирования должны использоваться младшие значащие биты. |

## 20.36 Описание регистров контроллеров интерфейсов обмена GATE_CNTR

Таблица 622 – Регистры контроллеров интерфейсов обмена

| **Базовый Адрес** | | **Название** | **Описание** |
|---|---|---|---|
| 0x2103_0000 | | GATE_0 | Контроллер шлюза, канал 0 |
| 0x2103_0080 | | GATE_1 | Контроллер шлюза, канал 1 |
| **Смещение** | | | |
| 0x0000 | 0 | FIFO_OP_TO_SF | Регистр записи во входное FIFO (от открытой стороны к защищенной), читаемое на защищенной стороне |
| 0x0004 | 1 | FIFO_SF_TO_OP | Регистр чтения из выходного FIFO (от защищенной стороны к открытому), записываемого на защищенной стороне |
| 0x0008 | 2 | FIFO_LEVELS | Регистр задания контрольных уровней заполненности FIFO |
| 0x000C | 3 | FIFO_INT_MASK | Регистр задания маски разрешения источников запроса прерывания |
| 0x0010 | 4 | FIFO_INT_SOURCE | Регистр индикации и очистки источников запроса прерывания |
| 0x0014 | 5 | OP_REG0 | Регистры записи данных, читаемых на защищенной стороне |
| | | … | |
| 0x0030 | 12 | OP_REG7 | |
| 0x0034 | 13 | OP_REG8 | Регистры чтения данных, записанных на защищенной стороне |
| | | … | |
| 0x0050 | 20 | OP_REG15 | |
| 0x0054 | 21 | REGS_BUSY | Регистр статуса занятости входных регистров |
| 0x0058 | 22 | REGS_INT_MASK | Регистр задания маски разрешения источников запроса прерывания REGxx |
| 0x005C | 23 | REGS_INT_SOURCE | Регистр индикации и очистки источников запроса прерывания REGxx |

### 20.36.1 *FIFO_OP_TO_SF*

Таблица 623 – Описание бит регистра FIFO_OP_TO_SF

| Base ADDR= | 0x2103_0000<br>0x2103_0080 | Offset= | 0x0000_0000 | Reset= | 0x----_---- |
|---|---|---|---|---|---|
| REG Name: | FIFO_OP_TO_SF | | | | |

| 31…0 |
|---|
| FIFO_OP_TO_SF[31:0] |
| WO |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31…0 | FIFO_OP_TO_SF | WO | 0x----_---- | Поле записи во входное FIFO, читаемое на защищенной стороне. Размер FIFO – 16 32-битных слов. |

### 20.36.2 *FIFO_SF_TO_OP*

Таблица 624 – Описание бит регистра FIFO_SF_TO_OP

| Base ADDR= | 0x2103_0000<br>0x2103_0080 | Offset= | 0x0000_0004 | Reset= | 0x----_---- |
|---|---|---|---|---|---|
| REG Name: | FIFO_SF_TO_OP | | | | |

| 31…0 |
|---|
| FIFO_SF_TO_OP[31:0] |
| RO |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31…0 | FIFO_SF_TO_OP | RO | 0x----_---- | Поле чтения из выходного FIFO, записываемого на защищенной стороне. Размер FIFO – 16 32-битных слов.<br>При чтении пустого FIFO возвращает случайные данные. |

### 20.36.3 *FIFO_LEVELS*

Таблица 625 – Описание бит регистра FIFO_LEVELS

| Base ADDR= | 0x2103_0000<br>0x2103_0080 | Offset= | 0x0000_0008 | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | FIFO_LEVELS | | | | |

| 31…16 |
|---|
| Зарезервировано |
| |

| 15…10 | 9…5 | 4…0 |
|---|---|---|
| Зарезервировано | FIFO_OP_TO_SF_LEVEL[4:0] | FIFO_SF_TO_OP_LEVEL[4:0] |
| | R/W | R/W |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31…10 | - | - | 0 | Зарезервировано |
| 9…5 | FIFO_OP_TO_SF_LEVEL[4:0] | R/W | 0 | Поле задания контрольного уровня заполненности по записи входного FIFO, читаемого на открытой стороне. |
| 4…0 | FIFO_SF_TO_OP_LEVEL[4:0] | R/W | 0 | Поле задания контрольного уровня заполненности по чтению выходного FIFO, записываемого на открытой стороне. |

### 20.36.4 *FIFO_INT_MASK*

Таблица 626 – Описание бит регистра FIFO_INT_MASK

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Base ADDR= | 0x2103_0000 0x2103_0080 | | Offset= | | 0x0000_000C | | | Reset= | | | 0x0000_0000 | | | | |
| REG Name: | FIFO_INT_MASK | | | | | | | | | | | | | | |
| 31…16 | | | | | | | | | | | | | | | |
| Зарезервировано | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15…8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Зарезервировано | FIFO_SF_TO_OP_RD_FROM_EMPTY_MASK | FIFO_SF_TO_OP_FULL_MASK | FIFO_SF_TO_OP_GREATER_MASK | FIFO_SF_TO_OP_NOT_EMPTY_MASK | FIFO_OP_TO_SF_WR_TO_FULL_MASK | FIFO_OP_TO_SF_NOT_FULL_MASK | FIFO_OP_TO_SF_LOWER_MASK | FIFO_OP_TO_SF_EMPTY_MASK |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31…8 | - | - | 0 | Зарезервировано |
| 7 | FIFO_SF_TO_OP_RD_FROM_EMPTY_MASK | R/W | 0 | Разрешение прерывания от источника запроса: было осуществлено чтение из пустого выходного FIFO |
| 6 | FIFO_SF_TO_OP_FULL_MASK | R/W | 0 | Разрешение прерывания от источника запроса: выходное FIFO полно |
| 5 | FIFO_SF_TO_OP_GREATER_MASK | R/W | 0 | Разрешение прерывания от источника запроса: выходное FIFO заполнено выше уровня, указанного в поле FIFO_SF_TO_OP_LEVEL |
| 4 | FIFO_SF_TO_OP_NOT_EMPTY_MASK | R/W | 0 | Разрешение прерывания от источника запроса: выходное FIFO не пусто |
| 3 | FIFO_OP_TO_SF_WR_TO_FULL_MASK | R/W | 0 | Разрешение прерывания от источника запроса: была осуществлена запись в заполненное входное FIFO |
| 2 | FIFO_OP_TO_SF_NOT_FULL_MASK | R/W | 0 | Разрешение прерывания, если входное FIFO не заполнено |
| 1 | FIFO_OP_TO_SF_LOWER_MASK | R/W | 0 | Разрешение прерывания от источника запроса: входное FIFO заполнено ниже уровня, указанного в поле FIFO_OP_TO_SF_LEVEL |
| 0 | FIFO_OP_TO_SF_EMPTY_MASK | R/W | 0 | Разрешение прерывания от источника запроса: входное FIFO пусто |

### 20.36.5 *FIFO_INT_SOURCE*

Таблица 627 – Описание бит регистра FIFO_INT_SOURCE

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Base ADDR= | 0x2103_0000 0x2103_0080 | | Offset= | | 0x0000_0010 | | | Reset= | | | 0x0000_0005 | | | | |
| REG Name: | FIFO_INT_ SOURCE | | | | | | | | | | | | | | |
| 31…16 | | | | | | | | | | | | | | | |
| Зарезервировано | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15…8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Зарезервировано | FIFO_SF_TO_OP_RD_FROM_EMPTY | FIFO_SF_TO_OP_FULL | FIFO_SF_TO_OP_GREATER | FIFO_SF_TO_OP_NOT_EMPTY | FIFO_OP_TO_SF_WR_TO_FULL | FIFO_OP_TO_SF_NOT_FULL | FIFO_OP_TO_SF_LOWER | FIFO_OP_TO_SF_EMPTY |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31…8 | - | - | 0 | Зарезервировано |
| 7 | FIFO_SF_TO_OP_RD_FROM_EMPTY | R/W | 0 | Равно 1, если было осуществлено чтение из пустого выходного FIFO. Очищается в 0 по записи 1 в данный разряд. |
| 6 | FIFO_SF_TO_OP_FULL | R/W | 0 | Равно 1, если выходное FIFO полно. Очищается в 0 автоматически. |
| 5 | FIFO_SF_TO_OP_GREATER | R/W | 0 | Равно 1, если выходное FIFO заполнено выше уровня, указанного в поле FIFO_SF_TO_OP_LEVEL. Очищается в 0 автоматически. |
| 4 | FIFO_SF_TO_OP_NOT_EMPTY | R/W | 0 | Равно 1, если выходное FIFO не пусто. Очищается в 0 автоматически. |
| 3 | FIFO_OP_TO_SF_WR_TO_FULL | R/W | 0 | Равно 1, если была осуществлена запись в заполненное входное FIFO. Очищается в 0 по записи 1 в данный разряд. |
| 2 | FIFO_OP_TO_SF_NOT_FULL | R/W | 1 | Равно 1, если входное FIFO не заполнено. Очищается в 0 автоматически. |
| 1 | FIFO_OP_TO_SF_LOWER | R/W | 0 | Равно 1, если входное FIFO заполнено ниже уровня, указанного в поле FIFO_OP_TO_SF_LEVEL. Очищается в 0 автоматически. |
| 0 | FIFO_OP_TO_SF_EMPTY | R/W | 1 | Равно 1, если входное FIFO пусто. Очищается в 0 автоматически. |

### 20.36.6 *OP_REG0…OP_REG7*

Таблица 628 – Описание бит регистров OP_REG0…OP_REG7

| Base ADDR= | 0x2103_0000<br>0x2103_0080 | Offset= | 0x0000_0014<br>0x0000_0018<br>…<br>0x0000_0030 | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | OP_REG0…<br>OP_REG7 | | | | |
| 31…0 | | | | | |
| OP_REGx | | | | | |
| R/W | | | | | |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31…0 | OP_REGx | R/W | 0 | Поле записи в регистр, читаемый на защищенной стороне из SF_REGxx. |

### 20.36.7 *OP_REG8…OP_REG15*

Таблица 629 – Описание бит регистров OP_REG8…OP_REG15

| Base ADDR= | 0x2103_0000<br>0x2103_0080 | Offset= | 0x0000_0034<br>0x0000_0038<br>…<br>0x0000_0050 | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | OP_REG8…<br>OP_REG15 | | | | |

| 31…0 |
|---|
| OP_REGxx |
| RO |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31…0 | OP_REGxx | RO | 0 | Поле чтения из регистра SF_REGx, записанного на защищенной стороне. |

### 20.36.8 *REGS_BUSY*

Таблица 630 – Описание бит регистра REGS_BUSY

| Base ADDR= | 0x2103_0000<br>0x2103_0080 | Offset= | 0x0000_0054 | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | REGS_BUSY | | | | |

| 31…16 |
|---|
| Зарезервировано |
| |

| 15…8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Зарезервировано | OP_REG7_BUSY | OP_REG6_BUSY | OP_REG5_BUSY | OP_REG4_BUSY | OP_REG3_BUSY | OP_REG2_BUSY | OP_REG1_BUSY | OP_REG0_BUSY |
| | RO | RO | RO | RO | RO | RO | RO | RO |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31…8 | - | - | 0 | Зарезервировано |
| 7 | OP_REG7_BUSY | RO | 0 | Равно 1, если OP_REG8 занят. |
| 6 | OP_REG6_BUSY | RO | 0 | Равно 1, если OP_REG7 занят. |
| 5 | OP_REG5_BUSY | RO | 0 | Равно 1, если OP_REG6 занят. |
| 4 | OP_REG4_BUSY | RO | 0 | Равно 1, если OP_REG5 занят. |
| 3 | OP_REG3_BUSY | RO | 0 | Равно 1, если OP_REG4 занят. |
| 2 | OP_REG2_BUSY | RO | 0 | Равно 1, если OP_REG3 занят. |
| 1 | OP_REG1_BUSY | RO | 0 | Равно 1, если OP_REG2 занят. |
| 0 | OP_REG0_BUSY | RO | 0 | Равно 1, если OP_REG1 занят. |

### 20.36.9 *REGS_INT_MASK*

Таблица 631 – Описание бит регистра REGS_INT_MASK

| Base ADDR= | 0x2103_0000 0x2103_0080 | Offset= | 0x0000_0058 | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | REGS_INT_MASK | | | | |

| 31…16 |
|---|
| Зарезервировано |
| |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP_REG15_NEW_DATA_MASK | OP_REG14_NEW_DATA_MASK | OP_REG13_NEW_DATA_MASK | OP_REG12_NEW_DATA_MASK | OP_REG12_NEW_DATA_MASK | OP_REG12_NEW_DATA_MASK | OP_REG12_NEW_DATA_MASK | OP_REG12_NEW_DATA_MASK | OP_REG2_RD_BUSY_MASK | OP_REG2_RD_BUSY_MASK | OP_REG2_RD_BUSY_MASK | OP_REG2_RD_BUSY_MASK | OP_REG2_RD_BUSY_MASK | OP_REG2_RD_BUSY_MASK | OP_REG1_RD_BUSY_MASK | OP_REG0_RD_BUSY_MASK |
| 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 | R/W |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31 … 16 | - | - | 0 | Зарезервировано |
| 15 | OP_REG15_NEW_DATA_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG15 имеет новые данные. |
| 14 | OP_REG14_NEW_DATA_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG14 имеет новые данные. |
| 13 | OP_REG13_NEW_DATA_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG13 имеет новые данные. |
| 12 | OP_REG12_NEW_DATA_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG12 имеет новые данные. |
| 11 | OP_REG11_NEW_DATA_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG11 имеет новые данные. |
| 10 | OP_REG10_NEW_DATA_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG10 имеет новые данные. |
| 9 | OP_REG9_NEW_DATA_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG9 имеет новые данные. |
| 8 | OP_REG8_NEW_DATA_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG8 имеет новые данные. |
| 7 | OP_REG7_RD_BUSY_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG7: попытка записи в занятом состоянии. |
| 6 | OP_REG6_RD_BUSY_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG6: попытка записи в занятом состоянии. |
| 5 | OP_REG5_RD_BUSY_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG5 имеет новые данные. |
| 4 | OP_REG4_RD_BUSY_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG4: попытка записи в занятом состоянии. |
| 3 | OP_REG3_RD_BUSY_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG3: попытка записи в занятом состоянии. |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 2 | OP_REG2_RD_BUSY_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG2: попытка записи в занятом состоянии. |
| 1 | OP_REG1_RD_BUSY_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG1: попытка записи в занятом состоянии. |
| 0 | OP_REG0_RD_BUSY_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG0: попытка записи в занятом состоянии. |

### 20.36.10 *REGS_INT_SOURCE*

Таблица 632 – Описание бит регистра REGS_INT_SOURCE

| Base ADDR= | 0x2103_0000 0x2103_0080 | Offset= | 0x0000_005C | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | REGS_INT_SOURCE | | | | |

| 31…16 |
|---|
| Зарезервировано |
| |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP_REG15_NEW_DATA | OP_REG14_NEW_DATA | OP_REG13_NEW_DATA | OP_REG12_NEW_DATA | OP_REG12_NEW_DATA | OP_REG12_NEW_DATA | OP_REG12_NEW_DATA | OP_REG12_NEW_DATA | OP_REG2_RD_BUSY | OP_REG2_RD_BUSY | OP_REG2_RD_BUSY | OP_REG2_RD_BUSY | OP_REG2_RD_BUSY | OP_REG2_RD_BUSY | OP_REG1_RD_BUSY | OP_REG0_RD_BUSY |
| 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 | R/W |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31 … 16 | - | - | 0 | Зарезервировано |
| 15 | OP_REG15_NEW_DATA | R/W | 0 | OP_REG15 имеет новые данные. Очищается при чтении OP_REG15. |
| 14 | OP_REG14_NEW_DATA | R/W | 0 | OP_REG14 имеет новые данные. Очищается при чтении OP_REG14. |
| 13 | OP_REG13_NEW_DATA | R/W | 0 | OP_REG13 имеет новые данные. Очищается при чтении OP_REG13. |
| 12 | OP_REG12_NEW_DATA | R/W | 0 | OP_REG12 имеет новые данные. Очищается при чтении OP_REG12. |
| 11 | OP_REG11_NEW_DATA | R/W | 0 | OP_REG11 имеет новые данные. Очищается при чтении OP_REG11. |
| 10 | OP_REG10_NEW_DATA | R/W | 0 | OP_REG10 имеет новые данные. Очищается при чтении OP_REG10. |
| 9 | OP_REG9_NEW_DATA | R/W | 0 | OP_REG9 имеет новые данные. Очищается при чтении OP_REG9. |
| 8 | OP_REG8_NEW_DATA | R/W | 0 | OP_REG8 имеет новые данные. Очищается при чтении OP_REG8. |
| 7 | OP_REG7_RD_BUSY | R/W | 0 | OP_REG7: попытка записи в занятом состоянии. Очищается при записи 1. |
| 6 | OP_REG6_RD_BUSY | R/W | 0 | OP_REG6: попытка записи в занятом состоянии. Очищается при записи 1. |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 5 | OP_REG5_RD_BUSY | R/W | 0 | OP_REG5: попытка записи в занятом состоянии. Очищается при записи 1. |
| 4 | OP_REG4_RD_BUSY | R/W | 0 | OP_REG4: попытка записи в занятом состоянии. Очищается при записи 1. |
| 3 | OP_REG3_RD_BUSY | R/W | 0 | OP_REG3: попытка записи в занятом состоянии. Очищается при записи 1. |
| 2 | OP_REG2_RD_BUSY | R/W | 0 | OP_REG2: попытка записи в занятом состоянии. Очищается при записи 1. |
| 1 | OP_REG1_RD_BUSY | R/W | 0 | OP_REG1: попытка записи в занятом состоянии. Очищается при записи 1. |
| 0 | OP_REG0_RD_BUSY | R/W | 0 | OP_REG0: попытка записи в занятом состоянии. Очищается при записи 1. |

Примечание – Для организации надежного сброса статусных флагов прерываний необходимо учитывать требования, изложенные в пункте 22.1.2 «Обработчики исключений».

# 21 Программная модель защищенной подсистемы выполнения криптографических преобразований

## 21.1 Адресное пространство защищенной подсистемы

Адресное пространство защищенной подсистемы представлено на рисунке 226. Открытая подсистема микроконтроллера имеет доступ к устройствам защищенной подсистемы только посредством шлюза.

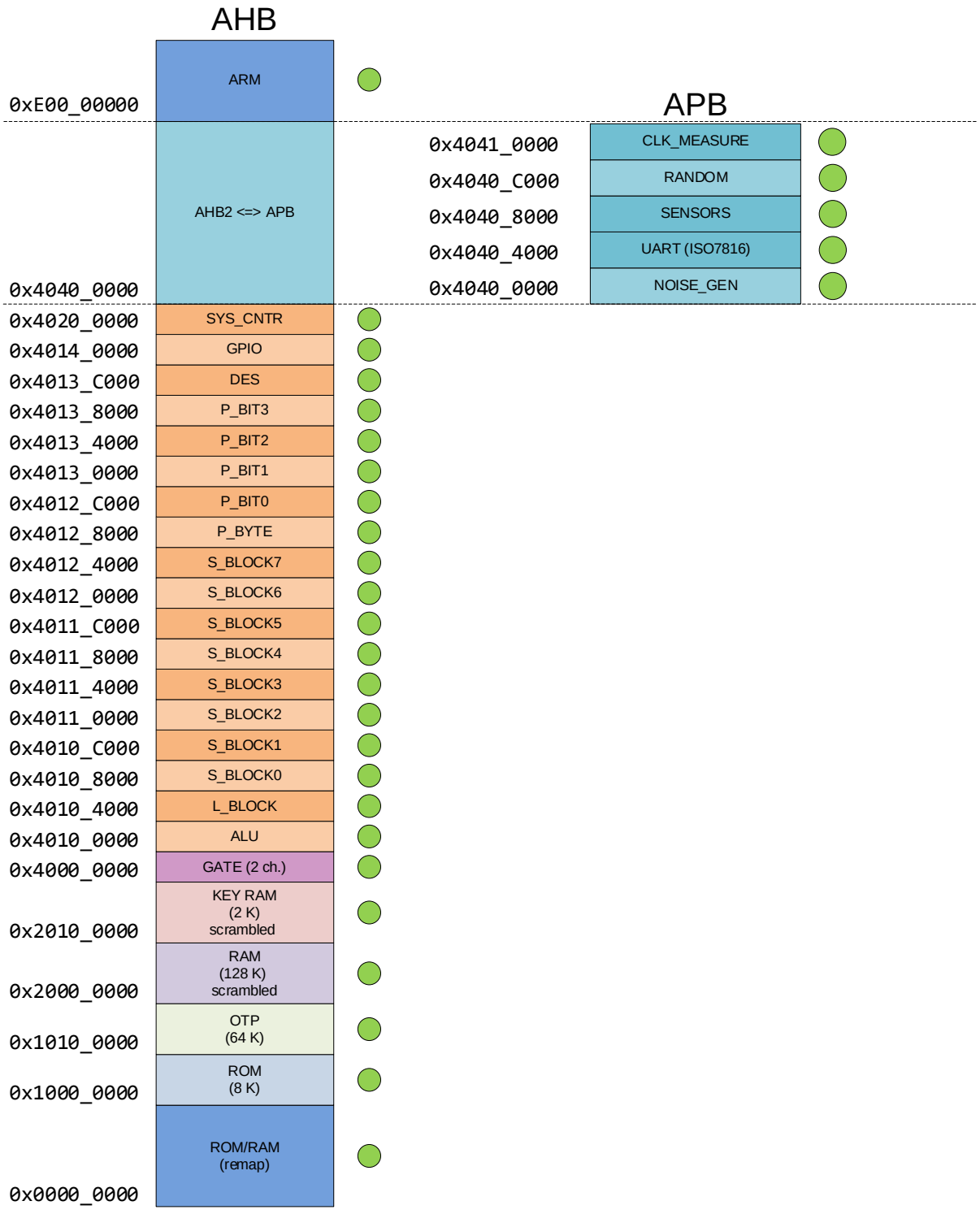



Рисунок 226 – Адресное пространство защищенной подсистемы

## 21.2 Описание регистров контроллера монитора частоты (CLK_MEASURE_CNTR)

Таблица 633 – Регистры контроллера монитора частоты

| **Базовый Адрес** | | **Название** | **Описание** |
|---|---|---|---|
| 0x4041_0000 | | CLK_MEASURE | Блок монитора частоты |
| **Смещение** | | | |
| 0x0000 | 0 | CLK_CNTR_STAT | Регистр настроек и состояния |
| 0x0004 | 1 | ALARM_SHIFT_RST | Регистр задания пороговых значений для сигнала сброса |
| 0x0008 | 2 | ALARM_SHIFT_INT | Регистр задания пороговых значений для сигнала прерывания |
| 0x000c | 3 | ALARM_PREG_0 | Регистр настройки параметров сброса счетчиков |
| 0x0010 | 4 | ALARM_PREG_1 | Регистр настройки параметров сброса счетчиков |
| 0x0014 | 5 | CLK_STAT | Регистр максимального значения SHIFT_REG каждого из каналов |

### 21.2.1 *CLK_CNTR_STAT*

Таблица 634 – Описание бит регистра CLK_CNTR_STAT

| Base ADDR= | **0x4041_0000** | Offset= | **0x0000_0000** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | CLK_CNTR_STAT | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | | | | | | | KEY_RESET | INTERRUPT | Зарезервировано | | | EN |
| | | | | | | | | | | r | r | | | | rw |

| **Бит** | **Имя** | **Значение** | **Описание** |
|---|---|---|---|
| 31…16 | Зарезервировано | 0 | Резерв |
| 5 | KEY_RESET | 0 | Флаг возникновения события сброса ключей |
| 4 | INTERRUPT | 0 | Флаг возникновения прерывания |
| 3…1 | Зарезервировано | 0 | Резерв |
| 0 | EN | 0 | Бит разрешения работы блока контроля частоты<br>0 – блок выключен<br>1 – блок включен |

### 21.2.2 *ALARM_SHIFT_RST*

Таблица 635 – Описание бит регистра *ALARM_SHIFT_RST*

| Base ADDR= | **0x4041_0000** | Offset= | **0x0000_0004** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|

| REG Name: | ALARM_SHIFT_RST | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| BASE_REG2[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BASE_REG0[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…16 | BASE_REG2[15:0] | 0 | Значение MAX_SHIFT0, при котором возникает событие сброса ключей |
| 15…0 | BASE_REG0[15:0] | 0 | Значение MAX_SHIFT1, при котором возникает событие сброса ключей |

### 21.2.3 *ALARM_SHIFT_INT*

Таблица 636 – Описание бит регистра *ALARM_SHIFT_INT*

| Base ADDR= | **0x4041_0000** | | | Offset= | | | **0x0000_0008** | | | Reset= | | | 0x0000_0000 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | ALARM_SHIFT_INT | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| BASE_REG3[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BASE_REG1[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…16 | BASE_REG3[15:0] | 0 | Значение MAX_SHIFT0, при котором возникает прерывание |
| 15…0 | BASE_REG1[15:0] | 0 | Значение MAX_SHIFT1, при котором возникает прерывание |

### 21.2.4 *ALARM_PREG_0*

Таблица 637 – Описание бит регистра *ALARM_PREG_0*

| Base ADDR= | **0x4041_0000** | | | Offset= | | | **0x0000_000C** | | | Reset= | | | 0x0000_0000 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | ALARM_PREG_0 | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PREG0[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PREG1[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…16 | PREG0[15:0] | 0 | Поле основания счета для счетчика 0.<br>Всегда задавать больше нуля |
| 15…0 | PREG1[15:0] | 0 | Поле основания счета для счетчика 1.<br>Всегда задавать больше нуля |

### 21.2.5 *ALARM_PREG_1*

Таблица 638 – Описание бит регистра *ALARM_PREG_1*

| Base ADDR= | | | **0x400B_5000** | | | Offset= | | **0x0000_0010** | | | Reset= | | 0x0000_0000 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | ALARM_PREG_1 | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PREG2[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PREG3[15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…16 | PREG2[15:0] | 0 | Поле основания счета для счетчика 2.<br>Всегда задавать больше нуля |
| 15…0 | PREG3[15:0] | 0 | Поле основания счета для счетчика 3.<br>Всегда задавать больше нуля |

## 21.3 Описание регистров контроллера генератора случайных чисел (RANDOM_CNTR)

Таблица 639 – Регистры контроллера генератора случайных чисел

| Базовый Адрес | | Название | Описание |
|---|---|---|---|
| 0x4040_C000 | | RANDOM | Блок генератора случайных чисел |
| **Смещение** | | | |
| 0x0000 | 0 | STAT_CNTR | Регистр статуса и управления |
| 0x0004 | 1 | INT_CNTR | Регистр управления прерыванием |
| 0x0008 | 2 | CLK_DIV | Регистр делителя генератора |
| 0x000c | 3 | PAUSE | Регистр паузы включения |
| 0x0010 | 4 | OUTPUT | Регистр случайного значения |
| 0x0014 | 5 | PAUSE_CNT | Регистр счетчика паузы |
| 0x0018 | 6 | TEMP | Регистр сбора случайного числа |

### 21.3.1 *Регистр статуса и управления STAT_CNTR_REG*

Относительный адрес: 0x00, начальное значение 0x000000F0.

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 0 | FORCE_EN | Бит принудительного включения генераторов, при установленном данном бите в 1, состояние на входе enable не имеет значение | RW |
| 1 | EN_STATE | Генераторы включены, данный бит устанавливается в 1 если есть хоть одна группа включенных генераторов (значение MASK не нулевое, есть сигнал enable или бит FORCE_EN) и вышла пауза после включения (*см. PAUSE_REG)* | RO |
| 2 | BUSY | Генератор занят, идет сбор случайного числа | RO |
| 3 | UNREAD_DATA | Флаг наличия несчитанных данных. После сбора очередного случайного числа в данном бите появляется 1, бит сбрасывается при чтении случайного числа (*см. OUTPUT_REG)* | RO |
| 5:4 | MASK | Маска выбора активного генератора: 4 бит – генератор 0, 5 бит – генератор 1. Для включения генераторов необходимо также наличие сигнала enable или бита FORCE_EN. | RW |
| 7:6 | - | Зарезервировано | RO |
| 9:8 | WORK_MODE | Режим сбора случайного числа<br>00 - сбор случайных чисел остановлен<br>01 - однократный запуск сбора случайного числа (после начала сбора поле сбрасывается в 00 автоматически)<br>10 - сбор нового случайного числа начинается если нет непрочитанного случайного числа (UNREAD_DATA равен 0)<br>11 - постоянный сбор случайных числе, сбор нового случайного числа начинается сразу после окончания сбора очередного числа | RW |
| 11:10 | - | Зарезервировано | RO |
| 17:12 | BIT_CN | Счетчик оставшихся бит до сбора нового случайного числа | RO |
| 31:18 | - | Зарезервировано | RO |

### 21.3.2 *Регистр управления прерыванием INT_CNTR_REG*

Относительный адрес: 0x04.

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 0 | INT_EN | Разрешение прерывания. При наличии 1 в этом бите и 1 в поле UNREAD_DATA, на выходе interrupt появляется 1 | RW |
| 1 | UNREAD_DATA | Флаг наличия несчитанных данных. После сбора очередного случайного числа в данном бите появляется 1, бит сбрасывается при чтении случайного числа (*см. OUTPUT_REG)* | RO |
| 31:2 | - | *Зарезервировано* | RO |

### 21.3.3 *Регистр делителя клока генератора CLK_DIV_REG*

Относительный адрес: 0x08, начальное значение 7 (500 кГц @ 8МГц).

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 15:0 | DIV | Для формирования клока защелки данных генератора используется входной клок модуля, деленный на 2 • (DIV + 1). По описанию модуля генераторов частота защёлки данных на генераторах должна быть порядка 500 кГц | RW |
| 31:16 | - | Зарезервировано | RO |

### 21.3.4 *Регистр паузы включения PAUSE_REG*

Относительный адрес: 0x0C.

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 31:0 | PAUSE | После включения генераторов через внешний сигнал или бит управляющего регистра, при включении любого блока через маску, при добавлении новых включенных блоков к работающим, сбор случайного числа может быть начат не ранее чем выйдет данная пауза. Если включение происходит во время сбора случайного числа, сбор приостанавливается до тех пор, пока не выйдет данная пауза | RW |

### 21.3.5 *Регистр случайного значения OUTPUT_REG*

Относительный адрес: 0x10.

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 31:0 | DATA | Последние собранное случайного число. Во время сбора очередного случайного числа, значение данного регистра неизменно. Обновление данных происходит по окончанию сбора случайного числа | RO |

### 21.3.6 *Регистр счетчика паузы PAUSE_CNT_REG*

Относительный адрес: 0x14.

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 31:0 | CUR_PAUSE | Текущее значение счетчика паузы включения генераторов (значение для справки). 0 значение означает что пауза выдержана | RO |

### 21.3.7 *Регистр сбора случайного числа TEMP_REG*

Относительный адрес: 0x18.

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 31:0 | TEMP_DATA | Текущее значение сдвигового регистра сбора случайного числа (значение для справок) | RO |

## 21.4 Описание регистров контроллера обработки датчиков безопасности (SENSORS_CNTR)

Таблица 640 – Регистры контроллера обработки датчиков безопасности

| Базовый Адрес | Название | Описание |
|---|---|---|
| 0x4040_8000 | SENSORS_CNTR | |
| **Смещение** | | |
| 0x00 | SENS_STATE | Регистр состояния датчиков защиты |
| 0x04 | SENS_RT_STATE | Регистр мониторинга состояния входов сенсоров |
| 0x0C | SENS_ INT_MASK | Регистр макирования запроса прерывания |
| 0x10 | SENS_KEY_RESET_MASK | Регистр маскирования запроса сброса памяти ключей |
| 0x18 | SENS_ENABLE | Регистр разрешения работы датчиков |
| 0x1C | MESH_KEY7 | Регистр ключа защитной сетки 7 (биты 255:224) |
| 0x20 | MESH_KEY6 | Регистр ключа защитной сетки 6 (биты 223:192) |
| 0x24 | MESH_KEY5 | Регистр ключа защитной сетки 5 (биты 191:160) |
| 0x28 | MESH_KEY4 | Регистр ключа защитной сетки 4 (биты 159:128) |
| 0x2C | MESH_KEY3 | Регистр ключа защитной сетки 3 (биты 127:96) |
| 0x30 | MESH_KEY2 | Регистр ключа защитной сетки 2 (биты 95:64) |
| 0x34 | MESH_KEY1 | Регистр ключа защитной сетки 1 (биты 63:32) |
| 0x38 | MESH_KEY0 | Регистр ключа защитной сетки 0 (биты 31:0) |
| 0x3C | MESH_INIT1 | Регистр инициализации данных защитной сетки 1 (биты 63:32) |
| 0x40 | MESH_INIT0 | Регистр инициализации данных защитной сетки 0 (биты 31:0) |
| 0x44 | MESH_CNTR | Регистр управления защитной сетки |

### 21.4.1 SENS_STATE

Доступ: r/w

Значение после сброса: 0x0

Регистр статуса SENS_STATE предназначен для отображения срабатывания датчиков.

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 31:5 | - | Зарезервировано | RO |
| 4 | MESH_ALARM | Сигнал атаки на защитную сетку – mesh active alarm | RW |
| 3 | EMFI3_ALARM | Сигнал датчика ЭМИ 3 – mesh emfi alarm3 | RW |
| 2 | EMFI2_ALARM | Сигнал датчика ЭМИ 2 – mesh emfi alarm2 | RW |
| 1 | EMFI1_ALARM | Сигнал датчика ЭМИ 1 – mesh emfi alarm1 | RW |
| 0 | - | Зарезервировано | RO |

### 21.4.2 SENS_RT_STATE

Доступ: r/o

Значение после сброса: 0x0

Регистр SENS_RT_STATE предназначен для непрерывного мониторинга состояния входов сенсоров.

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 31:5 | - | Зарезервировано | RO |
| 4 | MESH_RT_ALARM | Сигнал датчика повреждения активной защитной сетки | RO |
| 3 | EMFI3_RT_ALARM | Сигнал датчика электромагнитной атаки 3 | RO |

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 2 | EMFI2_RT_ALARM | Сигнал датчика электромагнитной атаки 2 | RO |
| 1 | EMFI1_RT_ALARM | Сигнал датчика электромагнитной атаки 1 | RO |
| 0 | - | Зарезервировано | RO |

### 21.4.3 SENS_INT_MASK

Доступ: r/w

Значение после сброса: 0x0

Регистр SENS_INT_MASK предназначен для выбора датчиков, вызывающих формирования сигнала запроса прерывания. Разряды распределены также, как в регистре SENS_STATE. Сигнал прерывания формируется при наличии единичных значений в регистре статуса (SENS_STATE) в битах, помеченных единицами в регистре маски (SENS_INT_MASK). Возможна только однократная запись в регистр.

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 31:5 | - | Зарезервировано | RO |
| 4 | MESH_INT_MASK | Маска прерывания по сигналу датчика повреждения активной защитной сетки<br>0 – прерывание запрещено<br>1 – прерывание разрешено | RW |
| 3 | EMFI3_INT_MASK | Маска прерывания по сигналу датчика электромагнитной атаки 3<br>0 – прерывание запрещено<br>1 – прерывание разрешено | RW |
| 2 | EMFI2_INT_MASK | Маска прерывания по сигналу датчика электромагнитной атаки 2<br>0 – прерывание запрещено<br>1 – прерывание разрешено | RW |
| 1 | EMFI1_INT_MASK | Маска прерывания по сигналу датчика электромагнитной атаки 1<br>0 – прерывание запрещено<br>1 – прерывание разрешено | RW |
| 0 | - | Зарезервировано | RO |

### 21.4.4 SENS_KEY_RESET_MASK

Доступ: r/w

Значение после сброса: 0x0

Регистр SENS_KEY_RESET_MASK предназначен для выбора датчиков, вызывающих формирования сигнала запроса сброса памяти ключей. Разряды распределены также, как в регистре SENS_STATE. Сигнал прерывания формируется при наличии единичных значений в регистре статуса (SENS_STATE) в битах, помеченных единицами в регистре маски (SENS_KEY_RESET_MASK). Возможна только однократная запись в регистр.

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 31:5 | - | Зарезервировано | RO |
| 4 | MESH_KEY_RESET_MASK | Маска сброса памяти ключей по сигналу датчика повреждения активной защитной сетки<br>0 – сброс запрещен<br>1 – сброс разрешен | RW |
| 3 | EMFI3_KEY_RESET_MASK | Маска сброса памяти ключей по сигналу датчика электромагнитной атаки 3<br>0 – сброс запрещен<br>1 – сброс разрешен | RW |

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 2 | EMFI2_KEY_RESET_MASK | Маска сброса памяти ключей по сигналу датчика электромагнитной атаки 2<br>0 – сброс запрещен<br>1 – сброс разрешен | RW |
| 1 | EMFI1_KEY_RESET_MASK | Маска сброса памяти ключей по сигналу датчика электромагнитной атаки 1<br>0 – сброс запрещен<br>1 – сброс разрешен | RW |
| 0 | - | Зарезервировано | RO |

### 21.4.5 SENS_ENABLE

Доступ: r/w
Значение после сброса: 0x0

Регистр SENS_ENABLE предназначен для включения-выключения датчиков безопасности, соответствие бит датчикам такое же как в регистре SENS_STATE. Единица в бите соответствующего датчика – датчик включен, ноль - датчик выключен. Возможна только однократная запись в регистр.

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 31:5 | - | Зарезервировано | RO |
| 4 | MESH_EN | Разрешение работы датчика активной защитной сетки<br>0 – разрешен<br>1 – запрещен | RW |
| 3 | EMFI3_EN | Разрешение работы датчика электромагнитной атаки 3<br>0 – разрешен<br>1 – запрещен | RW |
| 2 | EMFI2_EN | Разрешение работы датчика электромагнитной атаки 2<br>0 – разрешен<br>1 – запрещен | RW |
| 1 | EMFI1_EN | Разрешение работы датчика электромагнитной атаки 1<br>0 – разрешен<br>1 – запрещен | RW |
| 0 | - | Зарезервировано | RO |

### 21.4.6 MESH_CNTR

Доступ: r/w
Значение после сброса: 0x0
Возможна только однократная запись в регистр.

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 31:3 | - | Зарезервировано | RO |
| 2:0 | MESH_CLK_DIV | Регистр делителя опорной частоты HSI для работы активной сетки:<br>000 – переключение каждые 3 такта<br>001 – переключение каждые 4 такта<br>010 – переключение каждые 8 тактов<br>011 – переключение каждые 16 такта<br>100 – переключение каждые 32 такта<br>101 – переключение каждые 64 такта<br>110 – переключение каждые 128 такта<br>111 – переключение каждые 256 тактов | RW |

## 21.5 Описание регистров контроллера USART ISO7816 (USART_CNTR)

Таблица 641 – Регистры контроллера USART ISO7816

| Базовый Адрес | | Название | Описание |
|---|---|---|---|
| 0x4040_4000 | | USART_CNTR | Блок контроллера USART (ISO7816) |
| **Смещение** | | | |
| 0x0000 | 0 | USART_SR | Регистр статуса |
| 0x0004 | 1 | USART_DR | Регистр данных |
| 0x0008 | 2 | USART_BRR | Регистр настройки скорости передачи данных |
| 0x000c | 3 | USART_CR1 | Регистр управления 1 |
| 0x0010 | 4 | USART_CR2 | Регистр управления 2 |
| 0x0014 | 5 | USART_CR3 | Регистр управления 3 |
| 0x0018 | 6 | USART_GTPR | Регистр защищенного интервала предделителя |

### 21.5.1 *USART_SR*

Таблица 642 – Описание бит регистра USART_SR

| Base ADDR= | **0x4040_4000** | Offset= | **0x0000_0000** | Reset= | 0x0000_00C0 |
|---|---|---|---|---|---|
| REG Name: | USART_SR | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | | | CTS | Зарезервировано | TXE | TC | RXNE | IDLE | ORE | NF | FE | PE |
| | | | | | | rw | | r | rw | rw | r | r | r | r | r |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…10 | Зарезервировано | 0 | Резерв |
| 9 | CTS | 0 | Флаг CTS<br>Данный бит устанавливается аппаратно, когда на входе CTS меняется уровень, если установлен бит CTSE. Бит сбрасывается программно записью 0. Прерывание генерируется, если CTSIE=1 в регистре USART_CR3.<br>0: не было изменения статуса линии CTS<br>1: на линии CTS обнаружено изменение уровня |
| 8 | Зарезервировано | 0 | Резерв |
| 7 | TXE | 1 | Регистр данных передатчика пуст<br>Данный бит устанавливается аппаратно, когда содержимое регистра TDR было перемещено в регистр смещения. Прерывание генерируется, если TXEIE=1 в регистре USART_CR1. Бит сбрасывается записью регистра USART_DR.<br>0: данные не были переданы в регистр сдвига<br>1: данные переданы в регистр сдвига |
| 6 | TC | 1 | Передача завершена |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| | | | Данный бит устанавливается аппаратно, когда передача фрейма, содержащего данные, завершена, и бит TXE установлен. Прерывание генерируется, если TCIE=1 в регистре USART_CR1. Бит сбрасывается программной последовательностью (чтение регистра USART_SR, затем запись регистра USART_DR). Бит TC может быть также сброшен записью 0. Данный сброс рекомендуется выполнять только при мультибуферном обмене.<br>0: Передача не завершена<br>1: Передача завершена |
| 5 | RXNE | 0 | Регистр данных приема не пуст<br>Данный бит устанавливается аппаратно, когда содержимое регистра сдвига RDR было передано в регистр USART_DR. Прерывание генерируется, если RXNEIE=1 в регистре USART_CR1. Бит сбрасывается чтением регистра USART_DR. Флаг RXNE может быть также сброшен записью 0. Данная последовательность сброса рекомендована для мультибуферного обмена.<br>0: данные не получены<br>1: полученные данные готовы для чтения |
| 4 | IDLE | 0 | Обнаружение линии IDLE (ожидание)<br>Данный бит устанавливается аппаратно, когда обнаружена линия Idle. Прерывание генерируется, если IDLEIE=1 в регистре USART_CR1. Бит сбрасывается программной последовательностью (чтение регистра USART_SR, затем чтения регистра USART_DR).<br>0: не обнаружена линия Idle<br>1: линия Idle обнаружена<br><br>Примечание – Бит IDLE не будет снова установлен, пока не установится бит RXNE (то есть пока не будет обнаружено состояние ожидания на линии). |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 3 | ORE | 0 | Ошибка переполнения<br>Данный бит устанавливается аппаратно, когда принятое в настоящий момент слово в регистре сдвига готово к передаче в регистр RDR, если RXNE=1. Прерывание генерируется, если RXNEIE=1 в регистре USART_CR1. Бит сбрасывается программной последовательностью (чтение регистра<br>USART_SR, затем чтение регистра USART_DR).<br>0: нет ошибки переполнения<br>1: обнаружена ошибка переполнения<br><br>Примечание – Когда установлен этот бит, содержимое регистра RDR не будет потеряно, но регистр сдвига будет перезаписан. Прерывание генерируется по флагу ORE в случае мультибуферного обмена, если установлен бит EIE. |
| 2 | NF | 0 | Флаг обнаружения шума<br>Данный бит устанавливается аппаратно, когда в принятом фрейме обнаружен шум. Бит сбрасывается программной последовательностью (чтение регистра USART_SR, затем чтение регистра USART_DR).<br>0: шум не обнаружен<br>1: шум обнаружен<br><br>Примечания<br>1 Данный бит не генерирует прерывание, так как он появляется одновременно с битом RXNE, который сам по себе генерирует прерывание по флагу NF в случае мультибуферного обмена, если установлен бит EIE.<br>2 Если шумов на линии нет, флаг NF может быть отключен установкой бита ONEBIT в 1 для увеличения допуска на отклонения приемопередатчика (см. Допуск ухода тактовой частоты приемника USART). |
| 1 | FE | 0 | Ошибка фрейма<br>Данный бит устанавливается аппаратно, когда произошла рассинхронизация, чрезмерный шум или был получен символ break. Бит сбрасывается программной последовательностью (чтение регистра USART_SR, затем чтение регистра USART_DR).<br>0: ошибки фрейма не обнаружено<br>1: обнаружена ошибка фрейма или символ break<br><br>Примечание – Данный бит не генерирует прерывание, так как он появляется одновременно с битом RXNE который сам по себе генерирует прерывание. Если слово, передаваемое в настоящий момент, генерирует и ошибку фрейма, и ошибку переполнения, оно будет передано, и установится только бит ORE.<br>Прерывание генерируется по флагу FE в случае мультибуферного обмена, если установлен бит EIE. |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 0 | PE | 0 | Ошибка четности<br>Данный бит устанавливается аппаратно, когда в режиме приема обнаружена ошибка четности. Бит сбрасывается программной последовательностью (чтение регистра статуса, затем чтение или запись регистра данных USART_DR). Программа должна дождаться, пока установится флаг RXNE, перед тем как сбрасывать бит PE.<br>Прерывание генерируется, если PEIE = 1 в регистре USART_CR1. Бит PE устанавливается одновременно со всеми остальными флагами RX (RXNE, ORE, NF, FE), поэтому данный бит может сгенерировать запрос прерывания только по окончании приёма.<br>0: нет ошибки четности<br>1: обнаружена ошибка четности |

Примечание – Для организации надежного сброса статусных флагов прерываний необходимо учитывать требования, изложенные в пункте 22.1.2 «Обработчики исключений».

### 21.5.2 *USART_DR*

Таблица 643 – Описание бит регистра USART_DR

| Base ADDR= | | | **0x4040_4000** | | | Offset= | | | **0x0000_0004** | | | Reset= | | | 0xXXXX_XXXX | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | USART_DR | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | | |
| Зарезервировано | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | | | | DR [8:0] | | | | | | | | |
| | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…9 | Зарезервировано | 0 | Резерв |
| 8…0 | DR [8:0] | 0 | Значение данных<br>Содержит принятый или передаваемый символ данных в зависимости от операции с регистром – чтение или запись. Регистр данных выполняет двойную функцию (чтение и запись), так как состоит из двух регистров, один для передачи данных (TDR), другой для приема данных (RDR).<br>Регистр TDR обеспечивает параллельный интерфейс между внутренней шиной и выходным регистром сдвига. Регистр RDR обеспечивает параллельный интерфейс между входным регистром сдвига и внутренней шиной.<br>Когда разрешена передача с генерацией бита четности (бит PCE установлен в 1 в регистре USART_CR1), значение, записываемое в MSB (бит 7 или бит 8 в зависимости от длины данных) не играет значения, поскольку он заменяется битов четности. Когда на приеме разрешен контроль четности, значение, считываемое из MSB, является принятым битом четности. |

### 21.5.3 *USART_BRR*

Таблица 644 – Описание бит регистра USART_BRR

| Base ADDR= | **0x4040_4000** | Offset= | **0x0000_0008** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | USART_BRR | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DIV_Mantissa [11:0] | | | | | | | | | | | | DIV_Fraction [3:0] | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…16 | Зарезервировано | 0 | Резерв |
| 15…4 | DIV_Mantissa [11:0] | 0 | мантисса USARTDIV.<br>Эти 12 бит определяют мантиссу делителя USART (USARTDIV). |
| 3…0 | DIV_Fraction [3:0] | 0 | дробная часть USARTDIV.<br>Эти 4 бита определяют дробную часть делителя USART (USARTDIV). Если OVER8=1, бит DIV_Fraction3 не учитывается и должен быть сброшен. |

### 21.5.4 *USART_CR1*

Таблица 645 – Описание бит регистра USART_CR1

| Base ADDR= | **0x4040_4000** | Offset= | **0x0000_000C** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | USART_CR1 | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OVER8 | Зарезервировано | UE | M | WAKE | PCE | PS | PEIE | TXEIE | TCIE | RXNEIE | IDLEIE | TE | RE | RWU | SBK |
| rw | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…16 | Зарезервировано | 0 | Резерв |
| 15 | OVER8 | 0 | Режим передискретизации<br>0: передискретизация на 16<br>1: передискретизация на 8 |

| | | | |
|---|---|---|---|
| | | | Примечание – Передискретизация на 8 недоступна в режиме Smartcard: если SCEN=1, тогда OVER8 принудительно сбрасывается аппаратурой в 0. |
| 14 | Зарезервировано | 0 | Резерв |
| 13 | UE | 0 | Разрешение USART<br>Сброс данного бита в 0 моментально отключает TX, RX и генератор CK. При этом текущие передачи по TX и RX обрываются (если они идут). Данный бит устанавливается и сбрасывается программно.<br>0: предделитель и выходы USART запрещены<br>1: USART разрешен |
| 12 | M | 0 | Длина слова<br>Данный бит определяет длину слова. Данный бит устанавливается и сбрасывается программно.<br>0: 1 стартовый бит, 8 бит данных, n стоп-бит<br>1: 1 стартовый бит, 9 бит данных, n стоп-бит<br><br>Примечание – Бит M не должен быть модифицирован во время передачи данных (и приема и передачи) |
| 11 | WAKE | 0 | Метод пробуждения<br>Данный бит определяет метод пробуждения USART. Данный бит устанавливается и сбрасывается программно.<br>0: Линия Idle (состояние ожидания на линии)<br>1: Метка адреса |
| 10 | PCE | 0 | Разрешение контроля четности<br>Данный бит выбирает аппаратную генерацию и обнаружение бита четности. Если разрешен контроль бита четности, вычисленное значение четности вставляется на место MSB (9 бит, если M=1; 8 бит, если M=0) и на приеме этот бит проверяется на четность. Данный бит устанавливается и сбрасывается программно. Как только бит установлен, PCE активируется после текущего байта (при приеме и при передаче).<br>0: Контроль четности запрещен<br>1: Контроль четности разрешен |
| 9 | PS | 0 | Выбор четности/нечетности<br>Данный бит выбирает, как вычисляется и проверяется бит четности – либо на четность, либо на нечетность (бит PCE установлен). Данный бит устанавливается и сбрасывается программно. Вариант контроля четности/нечетности выбирается после текущего байта.<br>0: Контроль на четность<br>1: Контроль на нечетность |
| 8 | PEIE | 0 | Разрешение прерывание по событию PE<br>Данный бит устанавливается и сбрасывается программно.<br>0: прерывание запрещено<br>1: прерывание генерируется при PE=1 в регистре USART_SR |
| 7 | TXEIE | 0 | Разрешение прерывание по событию TXE<br>Данный бит устанавливается и сбрасывается программно.<br>0: прерывание запрещено<br>1: прерывание генерируется при TXE=1 в регистре USART_SR |

| | | | |
|---|---|---|---|
| 6 | TCIE | 0 | Разрешение прерывания по завершению передачи<br>Данный бит устанавливается и сбрасывается программно.<br>0: прерывание запрещено<br>1: прерывание генерируется при TC=1 в регистре USART_SR |
| 5 | RXNEIE | 0 | Разрешение прерывания по событию RXNE<br>Данный бит устанавливается и сбрасывается программно.<br>0: прерывание запрещено<br>1: прерывание генерируется при ORE=1 или RXNE=1 в регистре USART_SR |
| 4 | IDLEIE | 0 | Разрешение прерывания по событию IDLE<br>Данный бит устанавливается и сбрасывается программно.<br>0: прерывание запрещено<br>1: прерывание генерируется при IDLE=1 в регистре USART_SR |
| 3 | TE | 0 | Разрешение передатчика<br>Данные бит разрешает передатчик. Данный бит устанавливается и сбрасывается программно.<br>0: Передатчик запрещен<br>1: Передатчик разрешен<br><br>Примечание – Во время передачи, нулевой импульс в бите TE (“0”, за которым следует “1”) передает преамбулу (сигнала ожидания на линии) после текущего слова, кроме режима smartcard. Когда установлен бит TE, существует 1-битная задержка перед началом передачи. |
| 2 | RE | 0 | Разрешение приемника<br>Данный бит разрешает приемник. Данный бит устанавливается и сбрасывается программно.<br>0: Приемник запрещен<br>1: Приемник разрешен, и он начинает искать стартовый бит |
| 1 | RWU | 0 | Пробуждение приемника<br>Данный бит определяет, находится ли приемопередатчик в режиме молчания или нет. Данный бит устанавливается и сбрасывается программно, а также может быть сброшен аппаратно при распознавании последовательности пробуждения.<br>0: приемник в активном режиме<br>1: приемник в режиме молчания<br><br>Примечание – Перед выбором режима молчания (Mute) (установка бита RWU) приемопередатчик USART должен сначала получить байт данных, иначе он не сможет работать в режиме молчания с пробуждением по обнаружению сигнала ожидания на линии.<br>При конфигурации пробуждения по метке адреса (WAKE=1) бит RWU не может быть модифицирован программно, пока установлен бит RXNE. |
| 0 | SBK | 0 | Отправка сигнала break<br>Данный бит используется для отправки символов break. Данный бит устанавливается и сбрасывается программно. Он должен быть установлен программой и будет сброшен |

| | | | |
|---|---|---|---|
| | | | аппаратно во время стопового бита или символа break.<br>0: Символ break не передается<br>1: Символ Break передается |

### 21.5.5 *USART_CR2*

Таблица 646 – Описание бит регистра USART_CR2

| Base ADDR= | | | **0x4040_4000** | | | Offset= | | **0x0000_0010** | | | Reset= | | 0x0000_0000 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | USART_CR2 | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Зарезервировано | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | STOP [1:0] | | CLKEN | CPOL | CPHA | LBCL | Зарезервировано | | | | ADD [3:0] | | | |
| | | rw | rw | rw | rw | rw | rw | | | | | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…14 | Зарезервировано | 0 | Резерв |
| 13..12 | STOP [1:0] | 0 | стоповые биты<br>Данные биты используются для программирования стоп бит.<br>00: зарезервировано<br>01: 0.5 стоп-бит<br>10: 2 стоп-бита<br>11: 1.5 стоп-бита |
| 11 | CLKEN | 0 | разрешение тактирования<br>Данный бит позволяет разрешить работу вывода CK.<br>0: Вывод CK запрещен<br>1: Вывод CK разрешен |
| 10 | CPOL | 0 | Полярность тактов<br>Данный бит позволяет выбрать полярность выхода тактов на выводе CK в синхронном режиме.<br>Совместно с битом CPHA данный бит определяет взаимосвязь тактов и данных<br>0: постоянное значение 0 на выводе CK вне окна передачи.<br>1: постоянное значение 1 на выводе CK вне окна передачи. |
| 9 | CPHA | 0 | Фаза тактов<br>Данный бит позволяет выбрать фазу выхода тактов на выводе CK в синхронном режиме.<br>Совместно с битом CPOL данный бит определяет взаимосвязь тактов и данных<br>0: первое изменение сигнала синхронизации является первым перепадом захвата данных<br>1: второе изменение сигнала синхронизации является первым перепадом захвата данных |
| 8 | LBCL | 0 | Тактовый импульс последнего бита |

| | | | |
|---|---|---|---|
| | | | Бит позволяет выбрать, будет ли тактовый импульс последнего переданного бита (MSB) выдаваться на вывод CK в синхронном режиме.<br>0: тактовый импульс последнего бита данных не выводится на вывод CK<br>1: тактовый импульс последнего бита данных выводится на вывод CK<br><br>Примечание – 1: последний бит это 8 или 9 бит данных, в зависимости от выбранного формата, задаваемого битом M в регистре USART_CR1. |
| 7…4 | Зарезервировано | 0 | Резерв |
| 3…0 | ADD [3:0] | 0 | адреса узла USART<br>Данное битовое поле указывает адрес режима USART.<br>Данные биты используются в многопроцессорном обмене во время режима молчания, для пробуждения приемопередатчика при обнаружении метки адреса.<br><br>Примечание – Эти 3 бита (CPOL, CPHA, LBCL) не должны быть записаны, пока разрешен передатчик. |

### 21.5.6 *USART_CR3*

Таблица 647 – Описание бит регистра USART_CR3

| Base ADDR= | **0x4040_4000** | Offset= | **0x0000_0014** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | USART_CR3 | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | ONEBIT | CSTIE | CTSE | RTSE | DMAT | DMAR | SCEN | NACK | HDSEL | Зарезервировано | | EIE |
| | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | | | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…12 | Зарезервировано | 0 | Резерв |
| 11 | ONEBIT | 0 | разрешение режима одной выборки на бит<br>Бит позволяет выбрать метод оцифровки сигнала. Когда выбран метод одной выборки, флаг обнаружения шума (NF) запрещен.<br>0: метод трех выборок<br>1: метод одной выборки<br>Примечание – функция ONEBIT применима только к битам данных. Это не относится к стартовому биту. |
| 10 | CSTIE | 0 | Разрешение прерывания CTS<br>0: прерывание запрещено<br>1: прерывание генерируется каждый раз при CTS=1 в регистре USART_SR |

| | | | |
|---|---|---|---|
| 9 | CTSE | 0 | Разрешение CTS<br>0: запрещено аппаратное управление потоком CTS<br>1: режим CTS разрешен, данные передаются только, когда вход CTS имеет логический 0. Если вход CTS установлен на лог. 1 во время передачи данных, тогда передача завершается до остановки. Если данные записаны в регистр данных, когда CTS в логическую «1», то передача будет отложена до тех пор, пока на входе CTS не появится логический «0». |
| 8 | RTSE | 0 | Разрешение RTS<br>0: запрещено аппаратное управление потоком RTS<br>1: прерывание RTS разрешено, данные запрашиваются, когда есть место в буфере приемника. Ожидается, что передача данных прекратится после того, как будет передан последний текущий символ данных. Выход RTS имеет логический 0, когда приемопередатчик может принимать данные. |
| 7 | DMAT | 0 | Разрешение передатчика DMA<br>Данный бит устанавливается и сбрасывается программно.<br>1: Режим DMA разрешен для передачи.<br>0: Режим DMA запрещен для передачи. |
| 6 | DMAR | 0 | Разрешение приемника DMA<br>Данный бит устанавливается и сбрасывается программно.<br>1: Режим DMA разрешен для приема<br>0: Режим DMA запрещен для приема |
| 5 | SCEN | 0 | Разрешение режима Smartcard<br>Данный бит используется для разрешения режима Smartcard.<br>0: Режим Smartcard запрещен<br>1: Режим Smartcard разрешен |
| 4 | NACK | 0 | Разрешение NACK в режиме Smartcard<br>0: ответ NACK в случае ошибки четности запрещен<br>1: ответ NACK в случае ошибки четности разрешен |
| 3 | HDSEL | 0 | Выбор полудуплексного режима<br>Бит позволяет выбрать однопроводной режим или полудуплексный режим<br>0: полудуплексный режим не выбран<br>1: полудуплексный режим выбран |
| 2..1 | Зарезервировано | 0 | Резерв |
| 0 | EIE | 0 | Разрешение прерывания по ошибке<br>Бит используется для разрешения генерации прерывания по ошибке фрейма, ошибке переполнения или при флаге шума (FE=1 или ORE=1 или NF=1 в регистре USART_SR) в случае мультибуферного обмена (DMAR=1 в регистре USART_CR3).<br>0: прерывание запрещено<br>1: прерывание генерируется каждый раз, когда DMAR=1 в регистре USART_CR3 и FE=1 или ORE=1 или NF=1 в регистре USART_SR. |

### 21.5.7 *USART_GTPR*

Таблица 648 – Описание бит регистра USART_GTPR

| Base ADDR= | | | **0x4040_4000** | | | Offset= | | **0x0000_0018** | | | Reset= | | 0x0000_0000 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | USART_GTPR | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Зарезервировано | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GT [7:0] | | | | | | | | Зарезервировано | | | PSC [4:0] | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | | | | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…16 | Зарезервировано | 0 | Резерв |
| 15…8 | GT [7:0] | 0 | Значение защитного интервала<br>Это битовое поле показывает значение защитного интервала времени в единицах тактов скорости. Биты используются в режиме Smartcard. Флаг завершения передачи устанавливается после этого значения защитного интервала. |
| 7…5 | Зарезервировано | 0 | Резерв |
| 4…0 | PSC [4:0] | 0 | Значение предделителя в режиме smartcard<br>Используется для программирования предделителя для деления системной тактовой частоты, чтобы обеспечить тактирование смарт-карты.<br>Значение, заданное в регистре (5 значащих бит), умножается на 2, чтобы получить коэффициент деления исходной тактовой частоты:<br>00000: зарезервировано – не программировать это значение;<br>00001: делит источник тактов на 2;<br>00010: делит источник тактов на 4;<br>00011: делит источник тактов на 6. |

## 21.6 Описание регистров контроллера генератора шума (NOISE_GEN_CNTR)

Таблица 649 – Регистры контроллера генератора шума

| **Базовый Адрес** | | **Название** | **Описание** |
|---|---|---|---|
| 0x4040_0000 | | NOISE_GEN | |
| **Смещение** | | | |
| 0x0000 | 0 | CONTROL | Регистр задания режима работы подблоков ПСП |
| 0x0004 | 1 | SEED_0 | Регистр начального значения для подблока ПСП №0 |
| 0x0008 | 2 | SEED_1 | Регистр начального значения для подблока ПСП №1 |
| 0x000C | 3 | SEED_2 | Регистр начального значения для подблока ПСП №2 |
| 0x0010 | 4 | SEED_3 | Регистр начального значения для подблока ПСП №3 |
| 0x0014 | 5 | LFSR_0 | Регистр текущего состояния LFSR подблока ПСП №0 |

| **Базовый Адрес** | | **Название** | **Описание** |
|---|---|---|---|
| 0x4040_0000 | | NOISE_GEN | |
| **Смещение** | | | |
| 0x0018 | 6 | LFSR_1 | Регистр текущего состояния LFSR подблока ПСП №1 |
| 0x001C | 7 | LFSR_2 | Регистр текущего состояния LFSR подблока ПСП №2 |
| 0x0020 | 8 | LFSR_3 | Регистр текущего состояния LFSR подблока ПСП №3 |
| 0x0024 | 9 | RAND_0 | Регистр текущего состояния выхода подблока ПСП №0 |
| 0x0028 | 10 | RAND_1 | Регистр текущего состояния выхода подблока ПСП №1 |
| 0x002C | 11 | RAND_2 | Регистр текущего состояния выхода подблока ПСП №2 |
| 0x0030 | 12 | RAND_3 | Регистр текущего состояния выхода подблока ПСП №3 |
| 0x0034 | 13 | POLY_0 | Регистр задания полинома LFSR для подблока ПСП №0 |
| 0x0038 | 14 | POLY_1 | Регистр задания полинома LFSR для подблока ПСП №1 |
| 0x003C | 15 | POLY_2 | Регистр задания полинома LFSR для подблока ПСП №2 |
| 0x0040 | 16 | POLY_3 | Регистр задания полинома LFSR для подблока ПСП №3 |
| 0x0044 | 17 | PERIODS | Регистр задания периодов обновления LFSR для подблоков ПСП №0-№3 |
| 0x0048 | 18 | MASK_0 | Регистр маски выхода подблока ПСП №0 |
| 0x400C | 19 | MASK_1 | Регистр маски выхода подблока ПСП №1 |
| 0x0050 | 20 | MASK_2 | Регистр маски выхода подблока ПСП №2 |
| 0x0054 | | MASK_3 | Регистр маски выхода подблока ПСП №3 |

### 21.6.1 *Регистр CONTROL*

Относительный адрес: 0x0.
Доступ: r/w.
Значение после сброса: 0x0.

| Разряды | Назначения разрядов |
|---|---|
| 31:16 | Поле зарезервировано (читается как 0) |
| 15 | Прямой выход подблока ПСП №3: 0 – выход генератора определяется битами 7 и 3, 1 – на выход всегда выдается внутренние значение регистра |
| 14 | Прямой выход подблока ПСП №2: 0 – выход генератора определяется битами 6 и 2, 1 – на выход всегда выдается внутренние значение регистра |
| 13 | Прямой выход подблока ПСП №1: 0 – выход генератора определяется битами 5 и 1, 1 – на выход всегда выдается внутренние значение регистра |
| 12 | Прямой выход подблока ПСП №0: 0 – выход генератора определяется битами 4 и 0, 1 – на выход всегда выдается внутренние значение регистра |
| 11 | Поле включения режима каскада Голлмана подблока ПСП №3: 0 — выключено, 1 — включено |
| 10 | Поле включения режима каскада Голлмана подблока ПСП №2: 0 — выключено, 1 — включено |
| 9 | Поле включения режима каскада Голлмана подблока ПСП №1: 0 — выключено, 1 — включено |
| 8 | Поле зарезервировано (читается как 0). Режима каскада Голлмана подблока ПСП №0 всегда отключён |
| 7 | Разрешение вывода подблока ПСП №3: 0 — запрещено (выводы устанавливаются в 0, кроме режима прямого вывода бит 15), 1 — разрешено |
| 6 | Разрешение вывода подблока ПСП №2: 0 — запрещено (выводы устанавливаются в 0, кроме режима прямого вывода бит 14), 1 — разрешено |
| 5 | Разрешение вывода подблока ПСП №1: 0 — запрещено (выводы устанавливаются в 0, кроме режима прямого вывода бит 13), 1 — разрешено |
| 4 | Разрешение вывода подблока ПСП №0: 0 — запрещено (выводы устанавливаются в 0, кроме режима прямого вывода бит 12), 1 — разрешено |
| 3 | Режим работы подблока ПСП №3: 0 — инициализация, 1 — генерация псевдослучайных чисел |
| 2 | Режим работы подблока ПСП №2: 0 — инициализация, 1 — генерация псевдослучайных чисел |
| 1 | Режим работы подблока ПСП №1: 0 — инициализация, 1 — генерация псевдослучайных чисел |
| 0 | Режим работы подблока ПСП №0: 0 — инициализация, 1 — генерация псевдослучайных чисел |

В режиме инициализации подблок ПСП в каждом такте записывает внутрь текущее значение соответствующего регистра SEED_x и POLY_x и в этом режиме псевдослучайные числа не генерируются на выходе подблока ПСП. В режиме генерации псевдослучайных чисел в каждом следующем такте могут генерироваться новые выходные значения.

### 21.6.2 *Регистр SEED_0*

Относительный адрес: 0x4.
Доступ: r/w.
Значение после сброса: 0x1.

| Разряды | Назначения разрядов |
|---|---|
| 31:0 | Поле начального значения регистра LFSR для подблока ПСП №0 |

Поле начального значения регистра LFSR должно быть больше нуля для нормальной работы генератора ПСП. В случае, если значение поля начального значения регистра LFSR равно нулю, на выходе подблока ПСП будет всегда нулевой выходной вектор.

### 21.6.3 *Регистр SEED_1*

Относительный адрес: 0x8.
Доступ: r/w.
Значение после сброса: 0x2.

| Разряды | Назначения разрядов |
|---|---|
| 31:0 | Поле начального значения регистра LFSR для подблока ПСП №1 |

Поле начального значения регистра LFSR должно быть больше нуля для нормальной работы генератора ПСП. В случае, если значение поля начального значения регистра LFSR равно нулю, на выходе подблока ПСП будет всегда нулевой выходной вектор.

### 21.6.4 *Регистр SEED_2*

Относительный адрес: 0xC.
Доступ: r/w.
Значение после сброса: 0x3.

| Разряды | Назначения разрядов |
|---|---|
| 31:0 | Поле начального значения регистра LFSR для подблока ПСП №2 |

Поле начального значения регистра LFSR должно быть больше нуля для нормальной работы генератора ПСП. В случае, если значение поля начального значения регистра LFSR равно нулю, на выходе подблока ПСП будет всегда нулевой выходной вектор.

### 21.6.5 *Регистр SEED_3*

Относительный адрес: 0x10.
Доступ: r/w.
Значение после сброса: 0x4.

| Разряды | Назначения разрядов |
|---|---|
| 31:0 | Поле начального значения регистра LFSR для подблока ПСП №3 |

Поле начального значения регистра LFSR должно быть больше нуля для нормальной работы генератора ПСП. В случае, если значение поля начального значения регистра LFSR равно нулю, на выходе подблока ПСП будет всегда нулевой выходной вектор.

### 21.6.6 *Регистр LFSR_0*

Относительный адрес: 0x14.
Доступ: r/o.
Значение после сброса: 0x1.

| Разряды | Назначения разрядов |
|---|---|
| 31:0 | Поле текущего состояния LFSR подблока ПСП №0 |

### 21.6.7 *Регистр LFSR_1*

Относительный адрес: 0x18.
Доступ: r/o.
Значение после сброса: 0x1.

| Разряды | Назначения разрядов |
|---|---|
| 31:0 | Поле текущего состояния LFSR подблока ПСП №1 |

### 21.6.8 *Регистр LFSR_2*

Относительный адрес: 0x1C.
Доступ: r/o.
Значение после сброса: 0x1.

| Разряды | Назначения разрядов |
|---|---|
| 31:0 | Поле текущего состояния LFSR подблока ПСП №2 |

### 21.6.9 *Регистр LFSR_3*

Относительный адрес: 0x20.
Доступ: r/o.
Значение после сброса: 0x1.

| Разряды | Назначения разрядов |
|---|---|
| 31:0 | Поле текущего состояния LFSR подблока ПСП №3 |

### 21.6.10 *Регистр RAND_0*

Относительный адрес: 0x24.
Доступ: r/o.
Значение после сброса: 0x0.

| Разряды | Назначения разрядов |
|---|---|
| 31:0 | Поле текущего состояния выхода подблока ПСП №0 |

### 21.6.11 *Регистр RAND_1*

Относительный адрес: 0x28.
Доступ: r/o.
Значение после сброса: 0x0.

| Разряды | Назначения разрядов |
|---|---|
| 31:0 | Поле текущего состояния выхода подблока ПСП №1 |

### 21.6.12 *Регистр RAND_2*

Относительный адрес: 0x2C.
Доступ: r/o.
Значение после сброса: 0x0.

| Разряды | Назначения разрядов |
|---|---|
| 31:0 | Поле текущего состояния выхода подблока ПСП №2 |

### 21.6.13 *Регистр RAND_3*

Относительный адрес: 0x30.
Доступ: r/o.
Значение после сброса: 0x0.

| Разряды | Назначения разрядов |
|---|---|
| 31:0 | Поле текущего состояния выхода подблока ПСП №. |

### 21.6.14 *Регистр POLY_0*

Относительный адрес: 0x34.
Доступ: r/w.
Значение после сброса: 0x80000057.

| Разряды | Назначения разрядов |
|---|---|
| 31:0 | Поле задания полинома LFSR для подблока ПСП №0 |

Поле задания полинома LFSR поразрядно задаёт коэффициенты полинома: разряды [31:0] соответствуют коэффициентам от 32 до 1. Коэффициент 0 всегда считается равным 1.

### 21.6.15 *Регистр POLY_1*

Относительный адрес: 0x38.
Доступ: r/w.
Значение после сброса: 0x80000057.

| Разряды | Назначения разрядов |
|---|---|
| 31:0 | Поле задания полинома LFSR для подблока ПСП №1 |

Поле задания полинома LFSR поразрядно задаёт коэффициенты полинома: разряды [31:0] соответствуют коэффициентам от 32 до 1. Коэффициент 0 всегда считается равным 1.

### 21.6.16 *Регистр POLY_2*

Относительный адрес: 0x3C.
Доступ: r/w.
Значение после сброса: 0x80000057.

| Разряды | Назначения разрядов |
|---|---|
| 31:0 | Поле задания полинома LFSR для подблока ПСП №2 |

Поле задания полинома LFSR поразрядно задаёт коэффициенты полинома: разряды [31:0] соответствуют коэффициентам от 32 до 1. Коэффициент 0 всегда считается равным 1.

### 21.6.17 *Регистр POLY_3*

Относительный адрес: 0x40.
Доступ: r/w.
Значение после сброса: 0x80000057.

| Разряды | Назначения разрядов |
|---|---|
| 31:0 | Поле задания полинома LFSR для подблока ПСП №3 |

Поле задания полинома LFSR поразрядно задаёт коэффициенты полинома: разряды [31:0] соответствуют коэффициентам от 32 до 1. Коэффициент 0 всегда считается равным 1.

### 21.6.18 *Регистр PERIODS*

Относительный адрес: 0x44.
Доступ: r/w.
Значение после сброса: 0x0.

| Разряды | Назначения разрядов |
|---|---|
| 31:24 | Поле задания периода обновления подблока ПСП №3: период обновления (в тактах) на единицу больше значения поля |
| 23:16 | Поле задания периода обновления подблока ПСП №2: период обновления (в тактах) на единицу больше значения поля |
| 15:8 | Поле задания периода обновления подблока ПСП №1: период обновления (в тактах) на единицу больше значения поля |
| 7:0 | Поле задания периода обновления подблока ПСП №0: период обновления (в тактах) на единицу больше значения поля |

### 21.6.19 *Регистр MASK_0*

Относительный адрес: 0x48.
Доступ: r/w.
Значение после сброса: 0xFFFF_FFFF.

| **Разряды** | **Назначения разрядов** |
|---|---|
| 31:0 | Маска выхода подблока ПСП №0: для каждого бита 0 – выход всегда 0, 1 – выход задается подблоком |

### 21.6.20 *Регистр MASK_1*

Относительный адрес: 0x4C.
Доступ: r/w.
Значение после сброса: 0xFFFF_FFFF.

| **Разряды** | **Назначения разрядов** |
|---|---|
| 31:0 | Маска выхода подблока ПСП №1: для каждого бита 0 – выход всегда 0, 1 – выход задается подблоком |

### 21.6.21 *Регистр MASK_2*

Относительный адрес: 0x50.
Доступ: r/w.
Значение после сброса: 0xFFFF_FFFF.

| **Разряды** | **Назначения разрядов** |
|---|---|
| 31:0 | Маска выхода подблока ПСП №2: для каждого бита 0 – выход всегда 0, 1 – выход задается подблоком |

### 21.6.22 *Регистр MASK_3*

Относительный адрес: 0x54.
Доступ: r/w.
Значение после сброса: 0xFFFF_FFFF.

| **Разряды** | **Назначения разрядов** |
|---|---|
| 31:0 | Маска выхода подблока ПСП №3: для каждого бита 0 – выход всегда 0, 1 – выход задается подблоком |

## 21.7 Описание регистров системного контроллера (SYS_CNTR)

Таблица 650 – Регистры системного контроллера

| **Базовый Адрес** | | **Название** | **Описание** |
|---|---|---|---|
| 0x4020_0000 | | SYS_CNTR | |
| **Смещение** | | | |
| 0x0000 | 0 | STATUS | |
| 0x0004 | 1 | CNTR | |
| 0x0008 | 2 | PEN | |
| 0x000C | 3 | SEN | |
| 0x0010 | 4 | DSEN | |
| 0x0014 | 5 | MMAP | |
| 0x0018 | 6 | KSEED | |
| 0x001C | 7 | RSEED | |
| 0x0020 | 8 | M_SCNTR | |

### 21.7.1 *Описание бит регистра STATUS*

| Base ADDR= | **0x4020_0000** | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | STATUS | | | Offset= | **0x0000_0000** | | | | | | Reset= | 0x0000_0000 | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. | Res. |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Res. | Res. | Res. | Res. | Res. | DEBUG POWEN | Res. | MEM CNTR LOCK | DEBUG EN | SENSORS ALARM | Res. | KEY_RESET_REQ | OTP_CORR | RAM_CORR | KEY_CORR | MEMS_COR R |
| r | r | r | r | r | r | r | rw | r | rw | r | rw | rw | rw | rw | r |

| **Бит** | **Имя** | **Значение** | **Описание** |
|---|---|---|---|
| [31:11] | - | 0 | ***Зарезервировано*** |
| 10 | DEBUG POWEN | | Флаг готовности работы отладчика:<br>1 – готов<br>0 – не готов |
| 9 | - | 0 | ***Зарезервировано*** |
| 8 | MEM CNTR LOCK | 0 | Бит разрешения записи в регистр MEM_CNTR:<br>1 – запрещено;<br>0 – разрешено. |
| 7 | DEBUG EN | 0 | Флаг разрешения работы отладчика:<br>1 – разрешено<br>0 – запрещено |
| 6 | SENSORS ALARM | 0 | Флаг ошибки целостности от датчиков безопасности:<br>1 – было событие;<br>0 – не было события.<br>Флаг сбрасывается записью 1. |

| | | | |
|---|---|---|---|
| 5 | - | 0 | ***Зарезервировано*** |
| 4 | KEY_RESET_REQ | 0 | Бит программного сброса KEY RAM.<br>1 – сброс<br>0 – нет сброса. |
| 3 | OTP_CORR | 0 | Флаг ошибки целостности OTP:<br>1 – было событие;<br>0 – не было события.<br>Флаг сбрасывается записью 1. |
| 2 | RAM_CORR | 0 | Флаг ошибки целостности SRAM:<br>1 – было событие;<br>0 – не было события.<br>Флаг сбрасывается записью 1. |
| 1 | KEY_CORR | 0 | Флаг ошибки целостности KEY RAM:<br>1 – было событие;<br>0 – не было события.<br>Флаг сбрасывается записью 1. |
| 0 | MEMS_CORR | 0 | Общий флаг ошибки целостности памятей.<br>Данный флаг по ИЛИ объединяет флаги OTP_CORR \| RAM_CORR \| KEY_CORR. |

## 21.8 Описание регистров контроллера портов ввода-вывода (GPIO_CNTR)

Таблица 651 – Регистры контроллера портов ввода-вывода

| **Базовый Адрес** | | **Название** | **Описание** |
|---|---|---|---|
| 0x4014_0000 | | GPIO_CNTR | |
| **Смещение** | | | |
| 0x0000 | 0 | DATA | |
| 0x0004 | 1 | SET_CLR | |
| 0x0008 | 2 | OD_MODE | |
| 0x000C | 3 | SLEW_RATE | |
| 0x0010 | 4 | PORT_MODE | |
| 0x0014 | 5 | PU_PD | |
| 0x0018 | 6 | AF_L | |
| 0x001C | 7 | AF_H | |
| 0x0020 | 8 | INT_L | |
| 0x0024 | 9 | INT_H | |
| 0x0028 | 10 | INT_F | |
| 0x002C | 11 | INT_M | |
| 0x0030 | 12 | FILT_CURR | |
| 0x0034 | 13 | HCUR | |
| 0x0038 | 14 | LOCK | |

## 21.9 Описание регистров контроллера шифрования (DES_CNTR)

Таблица 652 – Регистры контроллера шифрования

| Базовый Адрес | | Название | Описание |
|---|---|---|---|
| 0x4013_C000 | | DES_CNTR | |
| **Смещение** | | | |
| 0x0000 | 0 | TRM | Регистр преобразования. |
| 0x0004 | 1 | KEY_L | Регистр младшей части ключа. |
| 0x0008 | 2 | KEY_H | Регистр старшей части ключа. |
| 0x000C | 3 | E_MUX_SET | Регистр установки S-преобразования. |
| 0x0010 | 4 | E_MUX_VAL | Регистр значения S-преобразования. |
| 0x0014 | 5 | P_MUX_SET | Регистр установки P-преобразования. |
| 0x0018 | 6 | P_MUX_VAL | Регистр значения P-преобразования. |
| 0x001C | 7 | STABLE | Регистр S-таблицы. |
| 0x0020 | 8 | M_SCNTR | Регистр памяти. |

## 21.10 Описание регистров контроллера битовой замены (P_BIT_CNTR)

Таблица 653 – Регистры контроллера битовой замены

| Базовый Адрес | | Название | Описание |
|---|---|---|---|
| 0x4012_C000 | | P_BIT0 | |
| 0x4013_0000 | | P_BIT1 | |
| 0x4013_4000 | | P_BIT2 | |
| 0x4013_8000 | | P_BIT3 | |
| **Смещение** | | | |
| 0x0000 | 0 | TRM | Регистр преобразования |
| 0x0004 | 1 | KEY_L | Регистр младшей части ключа |
| 0x0008 | 2 | KEY_H | Регистр старшей части ключа |
| 0x000C | 3 | E_MUX_SET | Регистр установки S-преобразования |
| 0x0010 | 4 | E_MUX_VAL | Регистр значения S-преобразования |
| 0x0014 | 5 | P_MUX_SET | Регистр установки P-преобразования |
| 0x0018 | 6 | P_MUX_VAL | Регистр значения P-преобразования |
| 0x001C | 7 | S_TABLE | Регистр S-таблицы |
| 0x0020 | 8 | M_SCNTR | Регистр памяти |

### 21.10.1 Регистр задания n-го слова данных для перестановки TRANSFORM_n

Адрес: 0x00 + 4 * n ... 0x0C.

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 31:0 | DATA | Запись в данный регистр передает n-ое слово для перестановки.<br>Чтение из данного регистра возвращает n-ое слово результата перестановки | RW |

### 21.10.2 Регистр задания перестановки MUX_SET

Адрес: 0x10.

При записи в данный регистр *(младшие 16 бит требуется записывать одновременно)* задается перестановка. Выходному биту с номером OUT_SEL ставится в соответствие входной бит с номером IN_SEL.

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 6:0 | IN_SEL | Запись задает перестановку IN_SEL → OUT_SEL.<br>Данное поле только для записи, при чтении возвращается 0. | WO |
| 7 | - | ***Зарезервировано*** | RO |
| 14:8 | OUT_SEL | Запись задает перестановку IN_SEL → OUT_SEL.<br>Данное поле только для записи, при чтении возвращается 0. | WO |
| 31:15 | - | ***Зарезервировано*** | RO |

### 21.10.3 Регистр чтения перестановки MUX_VAL

Адрес: 0x14.

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 6:0 | VAL | Данное поле только для чтения, запись игнорируется.<br>При чтении из данного поля возвращается значение перестановки для выбранного через SEL выходного бита | RO |
| 7 | - | ***Зарезервировано*** | RO |
| 14:8 | SEL | Запись в данное поле задает выходной бит, для которого буде возвращено значение перестановки при чтении регистра.<br>Данное поле только для записи, при чтении возвращается 0 | WO |
| 31:15 | - | ***Зарезервировано*** | RO |

## 21.11 Описание регистров контроллера байтовой замены (P_BYTE_CNTR)

Таблица 654 – Регистры контроллера байтовой замены

| Базовый Адрес | | Название | Описание |
|---|---|---|---|
| 0x4012_8000 | | P_BYTE | |
| **Смещение** | | | |
| 0x0000 | 0-15 | TRM0 – TRM15 | Регистр преобразования |
| 0x0020 | 8 | MUX_SET | Регистр перестановки |
| 0x0024 | 9 | MUX_VAL | Регистр чтения перестановки |

### 21.11.1 Регистр задания n-го слова данных для перестановки TRANSFORM_n

Адрес: 0x00 + 4 * n ... 0x3C.

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 31:0 | DATA | Запись в данный регистр передает n-ое слово для перестановки.<br>Чтение из данного регистра возвращает n-ое слово результата перестановки | RW |

### 21.11.2 Регистр задания перестановки MUX_SET

Адрес: 0x40.

При записи в данный регистр *(младшие 16 бит требуется записывать одновременно)* задается перестановка. Выходному байту с номером OUT_SEL ставится в соответствие входной байт с номером IN_SEL.

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 5:0 | IN_SEL | Запись задает перестановку IN_SEL → OUT_SEL<br>Данное поле только для записи, при чтении возвращается 0 | WO |
| 7:6 | - | ***Зарезервировано*** | RO |
| 13:8 | OUT_SEL | Запись задает перестановку IN_SEL → OUT_SEL<br>Данное поле только для записи, при чтении возвращается 0 | WO |
| 31:14 | - | ***Зарезервировано*** | RO |

### 21.11.3 Регистр чтения перестановки MUX_VAL

Адрес: 0x44.

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 5:0 | VAL | Данное поле только для чтения, запись игнорируется.<br>При чтении из данного поля возвращается значение перестановки для выбранного через SEL выходного байта. | RO |
| 7:6 | - | ***Зарезервировано*** | RO |
| 13:8 | SEL | Запись в данное поле задает выходной байт, для которого буде возвращено значение перестановки при чтении регистра.<br>Данное поле только для записи, при чтении возвращается 0. | WO |
| 31:14 | - | ***Зарезервировано*** | RO |

## 21.12 Описание регистров контроллера замены (S_BLOCK_CNTR)

Таблица 655 – Регистры контроллера замены

| Базовый Адрес | | Название | Описание |
|---|---|---|---|
| 0x4010_8000 | | S_BLOCK0 | Блок замены S-block0 |
| 0x4010_C000 | | S_BLOCK1 | Блок замены S-block1 |
| 0x4011_0000 | | S_BLOCK2 | Блок замены S-block2 |
| 0x4011_4000 | | S_BLOCK3 | Блок замены S-block3 |
| 0x4011_8000 | | S_BLOCK4 | Блок замены S-block4 |
| 0x4011_C000 | | S_BLOCK5 | Блок замены S-block5 |
| 0x4012_0000 | | S_BLOCK6 | Блок замены S-block6 |
| 0x4012_4000 | | S_BLOCK7 | Блок замены S-block7 |
| **Смещение** | | | |
| 0x0000 | 0 | TRANSFORM_0 | Регистр преобразования 0 |
| 0x0004 | 1 | TRANSFORM_1 | Регистр преобразования 1 |
| 0x0008 | 2 | TRANSFORM_2 | Регистр преобразования 2 |
| 0x000C | 3 | TRANSFORM_3 | Регистр преобразования 3 |
| 0x0010 | 4 | TRANSFORM_4 | Регистр преобразования 4 |
| 0x0014 | 5 | TRANSFORM_5 | Регистр преобразования 5 |

| Базовый Адрес | | Название | Описание |
|---|---|---|---|
| 0x4010_8000 | | S_BLOCK0 | Блок замены S-block0 |
| 0x4010_C000 | | S_BLOCK1 | Блок замены S-block1 |
| 0x4011_0000 | | S_BLOCK2 | Блок замены S-block2 |
| 0x4011_4000 | | S_BLOCK3 | Блок замены S-block3 |
| 0x4011_8000 | | S_BLOCK4 | Блок замены S-block4 |
| 0x4011_C000 | | S_BLOCK5 | Блок замены S-block5 |
| 0x4012_0000 | | S_BLOCK6 | Блок замены S-block6 |
| 0x4012_4000 | | S_BLOCK7 | Блок замены S-block7 |
| **Смещение** | | | |
| 0x0018 | 6 | TRANSFORM_6 | Регистр преобразования 6 |
| 0x001C | 7 | TRANSFORM_7 | Регистр преобразования 7 |
| 0x0020 | 8 | TABLE_CHANGE | Регистр изменения таблицы |
| 0x0024 | 9 | M_CNTR | Регистр памяти |

### 21.12.1 *TRANSFORM_x*

Таблица 656 – Описание бит регистра TRANSFORM_x

| Base ADDR= | **0x4010_4000** | | | | |
|---|---|---|---|---|---|
| REG Name: | TRANSFORM_0 | Offset= | **0x0000_0000** | Reset= | 0x0000_0000 |
| REG Name: | TRANSFORM_1 | Offset= | **0x0000_0004** | Reset= | 0x0000_0000 |
| REG Name: | TRANSFORM_2 | Offset= | **0x0000_0008** | Reset= | 0x0000_0000 |
| REG Name: | TRANSFORM_3 | Offset= | **0x0000_000C** | Reset= | 0x0000_0000 |
| REG Name: | TRANSFORM_4 | Offset= | **0x0000_0010** | Reset= | 0x0000_0000 |
| REG Name: | TRANSFORM_5 | Offset= | **0x0000_0014** | Reset= | 0x0000_0000 |
| REG Name: | TRANSFORM_6 | Offset= | **0x0000_0018** | Reset= | 0x0000_0000 |
| REG Name: | TRANSFORM_7 | Offset= | **0x0000_001C** | Reset= | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA [31:16] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA [15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…0 | DATA [31:0] | 0 | Запись в данный регистр передает n-ое слово для S-преобразования (результат преобразования доступен через 1 такт).<br>Запись в любой из этих регистров выбирает ячейки таблицы для изменения (см. регистр изменения таблицы).<br>Чтение из данного регистра возвращает результат S-преобразования n-го слова |

### 21.12.2 *TABLE_CHANGE*

Таблица 657 – Описание бит регистра TABLE_CHANGE

| Base ADDR= | **0x4010_4000** | | Offset= | **0x0000_0020** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|---|
| REG Name: | TABLE_CHANGE | | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TABLE_VAL_3 [4:0] | | | | | | | | TABLE_VAL_2 [4:0] | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TABLE_VAL_1 [4:0] | | | | | | | | TABLE_VAL_0 [4:0] | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31:24 | TABLE_VAL_3 | 0 | Запись в данное поле изменяет значение преобразования 3 - го (старшего) байта входного слова (адрес записи выбирается записью значение в TRANSFORM_n).<br>Чтение данного поля возвращает адрес изменяемой ячейки таблицы |
| 23:16 | TABLE_VAL_2 | 0 | Запись в данное поле изменяет значение преобразования 2 - го байта входного слова (адрес записи выбирается записью значение в TRANSFORM_n).<br>Чтение данного поля возвращает адрес изменяемой ячейки таблицы |
| 15:8 | TABLE_VAL_1 | 0 | Запись в данное поле изменяет значение преобразования 1 - го байта входного слова (адрес записи выбирается записью значение в TRANSFORM_n).<br>Чтение данного поля возвращает адрес изменяемой ячейки таблицы |
| 7:0 | TABLE_VAL_0 | 0 | Запись в данное поле изменяет значение преобразования 0 - го (младшего) байта входного слова (адрес записи выбирается записью значение в TRANSFORM_n).<br>Чтение данного поля возвращает адрес изменяемой ячейки таблицы |

## 21.13 Описание регистров контроллера прямого и обратного L-преобразования (L_BLOCK_CNTR)

Таблица 658 – Регистры контроллера прямого и обратного L-преобразования

| **Базовый Адрес** | | **Название** | **Описание** |
|---|---|---|---|
| 0x4010_4000 | | L_BLOCK | Блок прямого и обратного L преобразования |
| **Смещение** | | | |
| 0x0000 | 0 | TRANSFORM_0 | Регистр преобразования 0 |
| 0x0004 | 1 | TRANSFORM_1 | Регистр преобразования 1 |
| 0x0008 | 2 | TRANSFORM_2 | Регистр преобразования 2 |
| 0x000C | 3 | TRANSFORM_3 | Регистр преобразования 3 |
| 0x0010 | 4 | TABLE_CHANGE_0 | Регистр изменения таблицы 0 |
| 0x0014 | 5 | TABLE_CHANGE_1 | Регистр изменения таблицы 1 |
| 0x0018 | 6 | TABLE_CHANGE_2 | Регистр изменения таблицы 2 |
| 0x001C | 7 | TABLE_CHANGE_3 | Регистр изменения таблицы 3 |
| 0x0020 | 8 | SETUP | Регистр настройки преобразования |
| 0x0024 | 9 | M_CNTR | Регистр памяти |

### 21.13.1 *TRANSFORM_x*

Таблица 659 – Описание бит регистра TRANSFORM_x

| Base ADDR= | **0x4010_4000** | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | TRANSFORM_0 | | | Offset= | | | | **0x0000_0000** | | | | Reset= | | 0x0000_0000 | | |
| REG Name: | TRANSFORM_1 | | | Offset= | | | | **0x0000_0004** | | | | Reset= | | 0x0000_0000 | | |
| REG Name: | TRANSFORM_2 | | | Offset= | | | | **0x0000_0008** | | | | Reset= | | 0x0000_0000 | | |
| REG Name: | TRANSFORM_3 | | | Offset= | | | | **0x0000_000C** | | | | Reset= | | 0x0000_0000 | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA [31:16] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DATA [15:0] | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| **Бит** | **Имя** | **Значение** | **Описание** |
|---|---|---|---|
| 31…0 | DATA [31:0] | 0 | Запись в данный регистр передает n - ое слово для L / L-1 преобразования (запуск преобразования выполняется при записи в старший байт последнего регистра, результат преобразования доступен через (SET_R_COUNT + 1) тактов (см. регистр SETUP)).<br>Запись в данный регистр выбирает ячейки для изменения в n ой группы таблиц коэффициентов l преобразования (см. регистры изменения таблицы).<br>Чтение из данного регистра возвращает n-ое слово результата L / L-1 преобразования |

### 21.13.2 *TABLE_CHANGE_x*

Таблица 660 – Описание бит регистра TABLE_CHANGE_x

| Base ADDR= | **0x4010_4000** | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | TABLE_CHANGE_0 | | | Offset= | | | **0x0000_0010** | | | | Reset= | | | 0x0000_0000 | | |
| REG Name: | TABLE_CHANGE_1 | | | Offset= | | | **0x0000_0014** | | | | Reset= | | | 0x0000_0000 | | |
| REG Name: | TABLE_CHANGE_2 | | | Offset= | | | **0x0000_0018** | | | | Reset= | | | 0x0000_0000 | | |
| REG Name: | TABLE_CHANGE_3 | | | Offset= | | | **0x0000_001C** | | | | Reset= | | | 0x0000_0000 | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
| TABLE_VAL_n3 [4:0] | | | | | | | | TABLE_VAL_n2 [4:0] | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TABLE_VAL_n1 [4:0] | | | | | | | | TABLE_VAL_n0 [4:0] | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31:24 | TABLE_VAL_n3 | 0 | Запись в данное поле изменяет значение перевода a → K*a, 3-го (старшего) байта n-го входного слова (значение a выбирается записью значение в TRANSFORM_n). Изменения необходимо производить в режиме прямого преобразования с нулевым количеством преобразований R (см. регистр SETUP). Чтение данного поля возвращает текущее значение перевода a → K*a, 3-го (старшего) байта n-го входного слова (значение a выбирается записью значение в TRANSFORM_n) |
| 23:16 | TABLE_VAL_n2 | 0 | Запись в данное поле изменяет значение перевода a → K*a, 2-го байта n-го входного слова (значение a выбирается записью значение в TRANSFORM_n). Изменения необходимо производить в режиме прямого преобразования с нулевым количеством преобразований R (см. регистр SETUP). Чтение данного поля возвращает текущее значение перевода a → K*a, 2-го байта n-го входного слова (значение a выбирается записью значение в TRANSFORM_n) |
| 15:8 | TABLE_VAL_n1 | 0 | Запись в данное поле изменяет значение перевода a → K*a, 1-го байта n-го входного слова (значение a выбирается записью значение в TRANSFORM_n). Изменения необходимо производить в режиме прямого преобразования с нулевым количеством преобразований R (см. регистр SETUP). Чтение данного поля возвращает текущее значение перевода a → K*a, 1 - го байта n - го входного слова (значение a выбирается записью значение в TRANSFORM_n) |
| 7:0 | TABLE_VAL_n0 | 0 | Запись в данное поле изменяет значение перевода a → K*a, 0-го (младшего) байта n-го входного слова (значение a выбирается записью значение в TRANSFORM_n). Изменения необходимо производить в режиме прямого преобразования с нулевым количеством преобразований R (см. регистр SETUP). Чтение данного поля возвращает текущее значение перевода a → K*a, 0-го (младшего) байта n - го входного слова (значение a выбирается записью значение в TRANSFORM_n) |

### 21.13.3 *SETUP*

Таблица 661 – Описание бит регистра SETUP

| Base ADDR= | | | **0x4010_4000** | | | Offset= | | **0x0000_0020** | | | Reset= | | | 0x0000_0000 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | SETUP | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Зарезервировано | | | | | | | | | | | | | | CURR_L_DIR | SET_L_DIR |
| | | | | | | | | | | | | | | r | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | CURR_R_COUNT [4:0] | | | | | Зарезервировано | | | SET_R_COUNT [4:0] | | | | |
| | | | r | r | r | r | r | | | | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…18 | Зарезервировано | 0 | Резерв |
| 17 | CURR_L_DIR | 0 | Направление текущего, выполняемого преобразования. Запись в данное поле игнорируется |
| 16 | SET_L_DIR | 0 | Направление преобразования.<br>0 – прямое, L преобразование<br>1 – обратное, L - 1 преобразование<br>Направление выбирается на все преобразование при старте (при записи в старший байт регистра TRANSFORM_3). Изменение данного поля не влияет на результат выполняемого преобразования |
| 15…13 | Зарезервировано | 0 | Резерв |
| 12..8 | CURR_R_COUNT [4:0] | 0 | Число оставшихся R / R-1 в текущем, выполняем L / L-1 преобразовании. Нулевое значение означает что преобразование не начато или уже выполнено. Запись в данное поле игнорируется |
| 7…5 | Зарезервировано | 0 | Резерв |
| 4…0 | SET_R_COUNT [4:0] | 0 | Настройка количества R / R-1 преобразований при реализации L / L-1 преобразования. Нулевое значение – выключение преобразования.<br>Число R / R-1 преобразований выбирается на все L / L-1 преобразование при старте (при записи в старший байт регистра TRANSFORM_3). Изменение данного поля не влияет на результат выполняемого преобразования |

## 21.14 Описание регистров контроллера сопроцессора длинночисленной арифметики (ALU_CNTR)

### 21.14.1 Операции АЛУ

- **ADD** – сложение 128-битных слов без учета бита переноса предыдущей операции;
- **C_ADD** – сложение 128-битных слов с учетом бита переноса предыдущей операции;
- **O_ADD** – сложение 128-битных слов с добавлением единицы как бита переноса;
- **SHL** – сдвиг 128-битного слова на 1 бит влево, новый бит заполняется нулем;
- **C_SHL** – сдвиг 128-битного слова на 1 бит влево, новый бит заполняется битом переноса;
- **S_SHL** – сдвиг 128-битного слова на 1 бит влево, новый бит заполняется предыдущим выдвинутым битом
- **O_SHL** – сдвиг 128-битного слова на 1 бит влево, новый бит заполняется единицей;
- **SHR** – сдвиг 128-битного слова на 1 бит вправо, новый бит заполняется нулем;
- **C_SHR** – сдвиг 128 битного слова на 1 бит вправо, новый бит заполняется битом переноса;
- **S_SHR** – сдвиг 128-битного слова на 1 бит вправо, новый бит заполняется предыдущим выдвинутым битом;
- **O_SHR** – сдвиг 128-битного слова на 1 бит вправо, новый бит заполняется единицей;
- **MOV** – перемещение 128-битного слова (чтение из одного адреса и запись в другой адрес);
- **C_MOV** – перемещение 128-битного слова (чтение из одного адреса и запись в другой адрес) при условии установленного бита переноса, если бит переноса с предыдущей операции 0, перемещение не производится;
- **MUL** – умножение 128-битного слова на 128-битное слово с сохранением 256-битного результата;
- **NOP** – отсутствие действия.

### 21.14.2 *Регистры контроллера*

Таблица 662 – Регистры контроллера сопроцессора длинночисленной арифметики

| **Базовый Адрес** | | **Название** | **Описание** |
|---|---|---|---|
| 0x4010_0000 | | ALU | |
| **Смещение** | | | |
| 0x0000 | 0 | CMD_CNTR_CNT | Регистр управления счетчиком команд |
| 0x0004 | 1 | INT_FLAG | Регистр флагов прерываний |
| 0x0008 | 2 | INT_ENABLE | Регистр разрешения прерываний |
| 0x000C | 3 | ALU_STAT_CNTR | Регистр статуса и управления |
| 0x0010 | 4 | BASE_SHIFT | Регистр сдвига |
| 0x0014 | 5 | M_CNTR | Регистр памяти |

#### 21.14.2.1 *Регистр управления счетчиком команд CMD_CNTR_CNT*

Относительный адрес: 0x00.

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 9:0 | START_ADDR | Запись в данное поле задает начальный адрес серии команд для исполнения.<br>Чтение данного поля возвращает номер текущей, выполняемой серии команд | RW |
| 15:10 | SEQ_NUM | Номер серии команд для исполнения. На исполнение принимаются новые серии, только если переданный номер не совпадает с предыдущей, исполненной серией | RW |
| 25:16 | CMD_NUM | Запись в данное поле задает число команд для исполнения в новой серии.<br>Чтение данного поля возвращает число оставшихся на исполнение команд в текущей, выполняемой серии | RW |
| 30:26 | - | ***Зарезервировано*** | RO |
| 31 | BANK_SEL | Запись в данное поле задает рабочий банк следующей серии команд.<br>Чтение данного поля возвращается рабочий банк текущей, выполняемой серии.<br>Выбор одного рабочего банка автоматически подключает другой банк к AHB шине для изменения | RW |

#### 21.14.2.2 *Регистр флагов прерываний INT_FLAG*

Относительный адрес: 0x04.

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 0 | INTF_END | Флаг прерывания "выполнение командной последовательности закончено". Флаг выставляется в 1 при окончании выполнения очередной серии команд.<br>Запись 1 в данное поле сбрасывает флаг прерывания. Запись 0 не изменяет значение | RW |
| 1 | INTF_WCODE | Флаг прерывания "ошибка кода команды". Флаг выставляется в 1 при выдаче из памяти на исполнение неизвестной команды.<br>Запись 1 в данное поле сбрасывает флаг прерывания. Запись 0 не изменяет значение | RW |
| 2 | INTF_WOPER | Флаг прерывания "ошибка операнда". Флаг выставляется в 1 при неверном адресе операнда или результата.<br>Запись 1 в данное поле сбрасывает флаг прерывания. Запись 0 не изменяет значение | RW |
| 31:3 | - | ***Зарезервировано*** | RO |

Примечание – Для организации надежного сброса статусных флагов прерываний необходимо учитывать требования, изложенные в пункте 22.1.2 «Обработчики исключений».

#### 21.14.2.3 *Регистр разрешения прерываний INT_ENABLE*

Относительный адрес: 0x08.

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 0 | INTE_END | Разрешение соответствующего прерывания (***см. регистр флагов прерывания INT_FLAG).***<br>Если в данном поле стоит 1 и есть флаг в том же поле регистра INT_FLAG, модуль генерирует запрос прерывания. | RW |
| 1 | INTE_WCODE | | RW |
| 2 | INTE_WOPER | | RW |
| 31:3 | - | ***Зарезервировано*** | RO |

#### 21.14.2.4 *Регистр статуса и управления АЛУ ALU_STAT_CNTR*

Относительный адрес: 0x0C.

| Бит | Сокращенное наименование | Функция | Доступ |
|---|---|---|---|
| 0 | CARRY | Состояние бита переноса после последней операции сложения. Поле доступно только для чтения, запись в данное поле ничего не делает | RO |
| 1 | SHIFT | Значение выдвинутого бита после последней операции сдвига. Поле доступно только для чтения, запись в данное поле ничего не делает | RO |
| 2 | SET_CARRY | Задать состояние бита переноса в 1. После записи 1 в данное поле значение бита переноса становится равным 1. Запись 0 в данное поле ничего не делает. Поле доступно только для записи, чтение из данного поля всегда возвращает 0 | WO |
| 3 | SET_SHIFT | Задать состояние выдвинутого бита в 1. После записи 1 в данное поле значение выдвинутого бита становится равным 1. Запись 0 в данное поле ничего не делает. Поле доступно только для записи, чтение из данного поля всегда возвращает 0 | WO |
| 4 | CLR_CARRY | Сбросить состояние бита переноса в 0. После записи 1 в данное поле значение бита переноса становиться равным 0. Запись 0 в данное поле ничего не делает. Поле доступно только для записи, чтение из данного поля всегда возвращает 0 | WO |
| 5 | CLR_SHIFT | Сбросить состояние бита переноса в 1. После записи 1 в данное поле значение выдвинутого бита становится равным 0. Запись 0 в данное поле ничего не делает. Поле доступно только для записи, чтение из данного поля всегда возвращает 0 | WO |
| 31:6 | - | ***Зарезервировано*** | RO |

### 21.14.3 *Память инструкций*

Адрес: 0x1000.

Память инструкций представляет собой два банка по 1024 инструкции, каждая инструкция имеет длину 32 бита. Через регистр управления CMD_CNTR_CNT возможно выбирать рабочий банк и серию инструкций для выполнения. При выборе нулевого банка рабочий, первый банк автоматически становится доступным для чтения и записи через AHB по адресу 0x0400. При выборе рабочим первого банка, на AHB подключается нулевой банк. Запись и чтение производится только 32-битными словами.

Для запуска на выполнение серии инструкций в регистр CMD_CNTR_CNT необходимо записать рабочий банк, начальный адрес серии, длину серии, и номер серии отличный от текущего. После начала выполнения серии инструкций можно передавать команду для следующей серии, запуск новой серии произойдет по завершению текущей.

В случае ошибочных инструкций или адресов операнда происходит останов выполнения серии, новые серии не будут запущены до снятия флагов ошибок INTF_WCODE и INTE_WOPER в регистре флагов прерываний ***INT_FLAG***. При снятии флагов ошибок текущая, остановленная серия прекращает работу, выбирается нулевой банк как рабочий, счетчик серий также сбрасывается в ноль. До сброса флагов ошибок из регистра CMD_CNTR_CNT можно считать адрес следующей после остановки команды и число инструкций, оставшихся в серии. Если ошибка происходит на последней команде серии и была задана настройка для новой серии команд, из CMD_CNTR_CNT будет возвращаться адрес первой команды новой серии.

#### 21.14.3.1 Формат инструкции

| Бит | Сокращенное наименование | Функция |
|---|---|---|
| 7:0 | OP_CODE | Поле содержит код операции |
| 14:8 | OPERAND_A | Поле содержит адрес операнда A, адресация 128 битными словами из регистров<br>0 адрес - Регистр A[127:0]<br>1 адрес - Регистр A[255:128] и так далее<br>Регистры умножителя MA, MB, MC, MD имеют адреса 112 – 119 |
| 15 | INV_A | Единица в данном поле инвертирует значение операнда A при его использовании в команде |
| 22:16 | OPERAND_B | Поле содержит адрес операнда B. Допустимы адреса с 0 по 111, регистры умножителя не доступны как операнд B |
| 23 | INV_B | Единица в данном поле инвертирует значение операнда B при его использовании в команде |
| 30:24 | RESULT | Адрес сохранения результата команды, доступные значения от 0 до 111. Для команды умножения регистры MA, MB, MC, MD имеют адреса 0 – 3 и выбираются 256 битными словами |
| 31 | - | ***Зарезервировано*** |

#### 21.14.3.2 Коды операций

| Мнемоника | Значение | Действие |
|---|---|---|
| NOP | 0 | отсутствие действий |
| MOV | 1 | перемещает OPERAND_A → RESULT |
| C_MOV | 2 | перемещает OPERAND_A → RESULT если бит переноса равен 1, иначе команда работает как NOP |
| SHL | 10 | сдвиг OPERAND_A влево на 1 бит, новый бит заменяется 0, результат сдвига помещается в RESULT |
| C_SHL | 11 | сдвиг OPERAND_A влево на 1 бит, новый бит заменяется битом переноса, результат сдвига помещается в RESULT |
| S_SHL | 12 | сдвиг OPERAND_A влево на 1 бит, новый бит заменяется выдвинутым битом предыдущей операции, результат сдвига помещается в RESULT |
| O_SHL | 13 | сдвиг OPERAND_A влево на 1 бит, новый бит заменяется 1, результат сдвига помещается в RESULT |
| SHR | 15 | сдвиг OPERAND_A вправо на 1 бит, новый бит заменяется 0, результат сдвига помещается в RESULT |
| C_SHR | 16 | сдвиг OPERAND_A вправо на 1 бит, новый бит заменяется битом переноса, результат сдвига помещается в RESULT |
| S_SHR | 17 | сдвиг OPERAND_A вправо на 1 бит, новый бит заменяется выдвинутым битом предыдущей операции, результат сдвига помещается в RESULT |
| O_SHR | 18 | сдвиг OPERAND_A вправо на 1 бит, новый бит заменяется 1, результат сдвига помещается в RESULT |
| ADD | 20 | сумма (OPERAND_A + OPERAND_B) → RESULT |
| C_ADD | 21 | сумма (OPERAND_A + OPERAND_B + бит переноса предыдущей операции) → RESULT |
| O_ADD | 22 | сумма (OPERAND_A + OPERAND_B + 1) → RESULT |
| MUL | 25 | произведение (OPERAND_A * OPERAND_B) → RESULT |

## 21.15 Описание регистров контроллера шлюза передачи данных между подсистемами (GATE_CNTR)

Таблица 663 – Регистры контроллера шлюза передачи данных между подсистемами

| **Базовый Адрес** | | **Название** | **Описание** |
|---|---|---|---|
| 0x4000_0000 | | GATE_0 | Контроллер шлюза, канал 0 |
| 0x4000_0080 | | GATE_1 | Контроллер шлюза, канал 1 |
| **Смещение** | | | |
| 0x0000 | 0 | FIFO_OP_TO_SF | Регистр чтения из выходного FIFO (от открытой стороны к защищенной), записываемого на открытой стороне |
| 0x0004 | 1 | FIFO_SF_TO_OP | Регистр записи во входное FIFO (от защищенной стороны к открытой), читаемое на открытой стороне |
| 0x0008 | 2 | FIFO_LEVELS | Регистр задания контрольных уровней заполненности FIFO. |
| 0x000C | 3 | FIFO_INT_MASK | Регистр задания маски разрешения источников запроса прерывания |
| 0x0010 | 4 | FIFO_INT_SOURCE | Регистр индикации и очистки источников запроса прерывания |
| 0x0014 | 5 | SF_REG0 | Регистры чтения данных, записанных на открытой стороне |
| | | … | |
| 0x0030 | 12 | SF_REG7 | |
| 0x0034 | 13 | SF_REG8 | Регистры записи данных, читаемых на открытой стороне |
| | | … | |
| 0x0050 | 20 | SF_REG15 | |
| 0x0054 | 21 | REGS_BUSY | Регистр статуса занятости входных регистров |
| 0x0058 | 22 | REGS_INT_MASK | Регистр задания маски разрешения источников запроса прерывания REGxx |
| 0x005C | 23 | REGS_INT_SOURCE | Регистр индикации и очистки источников запроса прерывания REGxx |

### 21.15.1 *FIFO_OP_TO_SF*

Таблица 664 – Описание бит регистра FIFO_OP_TO_SF

| Base ADDR= | 0x4000_0000<br>0x4000_0080 | Offset= | 0x0000_0000 | Reset= | 0x----_---- |
|---|---|---|---|---|---|
| REG Name: | FIFO_OP_TO_SF | | | | |

| 31…0 |
|---|
| FIFO_OP_TO_SF[31:0] |
| RO |

| **Бит** | **Имя** | **Доступ** | **Значение** | **Описание** |
|---|---|---|---|---|
| 31…0 | FIFO_OP_TO_SF | RO | 0x----_---- | Поле чтения из выходного FIFO, записываемого на открытой стороне.<br>Размер FIFO – 16 32-битных слов.<br>При чтении пустого FIFO возвращает случайные данные. |

### 21.15.2 *FIFO_SF_TO_OP*

Таблица 665 – Описание бит регистра FIFO_SF_TO_OP

| Base ADDR= | 0x4000_0000 | Offset= | 0x0000_0004 | Reset= | 0x----_---- |
|---|---|---|---|---|---|

| | 0x4000_0080 | | | | |
|---|---|---|---|---|---|
| REG Name: | FIFO_SF_TO_OP | | | | |
| 31…0 | | | | | |
| FIFO_SF_TO_OP[31:0] | | | | | |
| WO | | | | | |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31…0 | FIFO_SF_TO_OP | WO | 0x----_---- | Поле записи во входное FIFO, читаемое на открытой стороне. Размер FIFO – 16 32-битных слов. |

### 21.15.3 *FIFO_LEVELS*

Таблица 666 – Описание бит регистра FIFO_LEVELS

| Base ADDR= | 0x4000_0000<br>0x4000_0080 | Offset= | 0x0000_0008 | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | FIFO_LEVELS | | | | |
| 31…16 | | | | | |
| Зарезервировано | | | | | |
| | | | | | |

| 15…10 | 9…5 | 4…0 |
|---|---|---|
| Зарезервировано | FIFO_SF_TO_OP_LEVEL[4:0] | FIFO_OP_TO_SF_LEVEL[4:0] |
| | R/W | R/W |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31…10 | - | - | 0 | Зарезервировано |
| 9…5 | FIFO_SF_TO_OP_LEVEL[4:0] | R/W | 0 | Поле задания контрольного уровня заполненности по записи входного FIFO, читаемого на открытой стороне. |
| 4…0 | FIFO_OP_TO_SF_LEVEL[4:0] | R/W | 0 | Поле задания контрольного уровня заполненности по чтению выходного FIFO, записываемого на открытой стороне. |

### 21.15.4 *FIFO_INT_MASK*

Таблица 667 – Описание бит регистра FIFO_INT_MASK

| Base ADDR= | 0x4000_0000<br>0x4000_0080 | Offset= | 0x0000_000C | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | FIFO_INT_MASK | | | | |
| 31…16 | | | | | |
| Зарезервировано | | | | | |
| | | | | | |

| 15…8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Зарезервировано | FIFO_OP_TO_SF_RD_FROM_EMPTY_MASK | FIFO_OP_TO_SF_FULL_MASK | FIFO_OP_TO_SF_GREATER_MASK | FIFO_OP_TO_SF_NOT_EMPTY_MASK | FIFO_SF_TO_OP_WR_TO_FULL_MASK | FIFO_SF_TO_OP_NOT_FULL_MASK | FIFO_SF_TO_OP_LOWER_MASK | FIFO_SF_TO_OP_EMPTY_MASK |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31…8 | - | - | 0 | Зарезервировано |
| 7 | FIFO_OP_TO_SF_RD_FROM_EMPTY_MASK | R/W | 0 | Разрешение прерывания от источника запроса: было осуществлено чтение из пустого выходного FIFO |
| 6 | FIFO_OP_TO_SF_FULL_MASK | R/W | 0 | Разрешение прерывания от источника запроса: выходное FIFO полно |
| 5 | FIFO_OP_TO_SF_GREATER_MASK | R/W | 0 | Разрешение прерывания от источника запроса: выходное FIFO заполнено выше уровня, указанного в поле FIFO_OP_TO_SF_LEVEL |
| 4 | FIFO_OP_TO_SF_NOT_EMPTY_MASK | R/W | 0 | Разрешение прерывания от источника запроса: выходное FIFO не пусто |
| 3 | FIFO_SF_TO_OP_WR_TO_FULL_MASK | R/W | 0 | Разрешение прерывания от источника запроса: была осуществлена запись в заполненное входное FIFO |
| 2 | FIFO_SF_TO_OP_NOT_FULL_MASK | R/W | 0 | Разрешение прерывания от источника запроса: входное FIFO не заполнено |
| 1 | FIFO_SF_TO_OP_LOWER_MASK | R/W | 0 | Разрешение прерывания от источника запроса: входное FIFO заполнено ниже уровня, указанного в поле FIFO_SF_TO_OP_LEVEL |
| 0 | FIFO_SF_TO_OP_EMPTY_MASK | R/W | 0 | Разрешение прерывания от источника запроса: входное FIFO пусто |

### 21.15.5 *FIFO_INT_SOURCE*

Таблица 668 – Описание бит регистра FIFO_INT_SOURCE

| Base ADDR= | 0x4000_0000 0x4000_0080 | Offset= | 0x0000_0010 | Reset= | 0x0000_0005 |
|---|---|---|---|---|---|
| REG Name: | FIFO_INT_SOURCE | | | | |

| 31…16 |
|---|
| Зарезервировано |
| |

| 15…8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Зарезервировано | FIFO_OP_TO_SF_RD_FROM_EMPTY | FIFO_OP_TO_SF_FULL | FIFO_OP_TO_SF_GREATER | FIFO_OP_TO_SF_NOT_EMPTY | FIFO_SF_TO_OP_WR_TO_FULL | FIFO_SF_TO_OP_NOT_FULL | FIFO_SF_TO_OP_LOWER | FIFO_SF_TO_OP_EMPTY |
| | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31…8 | - | - | 0 | Зарезервировано |
| 7 | FIFO_OP_TO_SF_RD_FROM_EMPTY | R/W | 0 | Равно 1, если было осуществлено чтение из пустого выходного FIFO. Очищается в 0 по записи 1 в данный разряд. |
| 6 | FIFO_OP_TO_SF_FULL | R/W | 0 | Равно 1, если выходное FIFO полно. Очищается в 0 автоматически. |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 5 | FIFO_OP_TO_SF_GREATER | R/W | 0 | Равно 1, если выходное FIFO заполнено выше уровня, указанного в поле FIFO_OP_TO_SF_LEVEL. Очищается в 0 автоматически. |
| 4 | FIFO_OP_TO_SF_NOT_EMPTY | R/W | 0 | Равно 1, если выходное FIFO не пусто. Очищается в 0 автоматически. |
| 3 | FIFO_SF_TO_OP_WR_TO_FULL | R/W | 0 | Равно 1, если была осуществлена запись в заполненное входное FIFO. Очищается в 0 по записи 1 в данный разряд. |
| 2 | FIFO_SF_TO_OP_NOT_FULL | R/W | 1 | Равно 1, если входное FIFO не заполнено. Очищается в 0 автоматически. |
| 1 | FIFO_SF_TO_OP_LOWER | R/W | 0 | Равно 1, если входное FIFO заполнено ниже уровня, указанного в поле FIFO_SF_TO_OP_LEVEL. Очищается в 0 автоматически. |
| 0 | FIFO_SF_TO_OP_EMPTY | R/W | 1 | Равно 1, если входное FIFO пусто. Очищается в 0 автоматически. |

### 21.15.6 *SF_REG0…SF_REG7*

Таблица 669 – Описание бит регистров SF_REG0…SF_REG7

| Base ADDR= | 0x4000_0000<br>0x4000_0080 | Offset= | 0x0000_0014<br>0x0000_0018<br>…<br>0x0000_0030 | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | SF_REG0…<br>SF_REG7 | | | | |

| 31…0 |
|---|
| SF_REGx |
| RO |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31…0 | SF_REGx | RO | 0 | Поле чтения из регистра OP_REGx, записанного на открытой стороне. |

### 21.15.7 *SF_REG8…SF_REG15*

Таблица 670 – Описание бит регистров SF_REG8…SF_REG15

| Base ADDR= | 0x4000_0000<br>0x4000_0080 | Offset= | 0x0000_0034<br>0x0000_0038<br>…<br>0x0000_0050 | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | SF_REG8…<br>SF_REG15 | | | | |

| 31…0 |
|---|
| SF_REGxx |
| R/W |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31…0 | SF_REGxx | R/W | 0 | Поле записи в регистр, читаемый на открытой стороне из OP_REGxx. |

### 21.15.8 *REGS_BUSY*

Таблица 671 – Описание бит регистра REGS_BUSY

| Base ADDR= | 0x4000_0000<br>0x4000_0080 | Offset= | | 0x0000_0054 | | | Reset= | | 0x0000_0000 | |
|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | REGS_BUSY | | | | | | | | | |
| 31…16 | | | | | | | | | | |
| Зарезервировано | | | | | | | | | | |
| | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7…0 |
|---|---|---|---|---|---|---|---|---|
| OP_REG15_BUSY | OP_REG14_BUSY | OP_REG13_BUSY | OP_REG12_BUSY | OP_REG11_BUSY | OP_REG10_BUSY | OP_REG9_BUSY | OP_REG8_BUSY | Зарезервировано |
| RO | RO | RO | RO | RO | RO | RO | RO | |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31…16 | - | - | 0 | Зарезервировано |
| 15 | OP_REG15_BUSY | RO | 0 | Равно 1, если OP_REG15 занят. |
| 14 | OP_REG14_BUSY | RO | 0 | Равно 1, если OP_REG14 занят. |
| 13 | OP_REG13_BUSY | RO | 0 | Равно 1, если OP_REG13 занят. |
| 12 | OP_REG12_BUSY | RO | 0 | Равно 1, если OP_REG12 занят. |
| 11 | OP_REG11_BUSY | RO | 0 | Равно 1, если OP_REG11 занят. |
| 10 | OP_REG10_BUSY | RO | 0 | Равно 1, если OP_REG10 занят. |
| 9 | OP_REG9_BUSY | RO | 0 | Равно 1, если OP_REG9 занят. |
| 8 | OP_REG8_BUSY | RO | 0 | Равно 1, если OP_REG8 занят. |
| 7…0 | - | - | 0 | Зарезервировано |

### 21.15.9 *REGS_INT_MASK*

Таблица 672 – Описание бит регистра REGS_INT_MASK

| Base ADDR= | 0x4000_0000 0x4000_0080 | Offset= | 0x0000_0058 | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | REGS_INT_MASK | | | | |

| 31…16 |
|---|
| Зарезервировано |
| |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP_REG15_RD_BUSY_MASK | OP_REG14_RD_BUSY_MASK | OP_REG13_RD_BUSY_MASK | OP_REG12_RD_BUSY_MASK | OP_REG12_RD_BUSY_MASK | OP_REG12_RD_BUSY_MASK | OP_REG12_RD_BUSY_MASK | OP_REG12_RD_BUSY_MASK | OP_REG2_NEW_DATA_MASK | OP_REG2_NEW_DATA_MASK | OP_REG2_NEW_DATA_MASK | OP_REG2_NEW_DATA_MASK | OP_REG2_NEW_DATA_MASK | OP_REG2_NEW_DATA_MASK | OP_REG1_NEW_DATA_MASK | OP_REG0_NEW_DATA_MASK |
| 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 | R/W |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31 … 16 | - | - | 0 | Зарезервировано |
| 15 | OP_REG15_RD_BUSY_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG15: попытка записи в занятом состоянии. |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 14 | OP_REG14_RD_BUSY_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG14: попытка записи в занятом состоянии. |
| 13 | OP_REG13_RD_BUSY_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG13: попытка записи в занятом состоянии. |
| 12 | OP_REG12_RD_BUSY_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG12: попытка записи в занятом состоянии. |
| 11 | OP_REG11_RD_BUSY_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG11: попытка записи в занятом состоянии. |
| 10 | OP_REG10_RD_BUSY_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG10: попытка записи в занятом состоянии. |
| 9 | OP_REG9_RD_BUSY_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG9: попытка записи в занятом состоянии. |
| 8 | OP_REG8_RD_BUSY_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG8: попытка записи в занятом состоянии. |
| 7 | OP_REG7_NEW_DATA_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG7 имеет новые данные. |
| 6 | OP_REG6_NEW_DATA_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG6 имеет новые данные. |
| 5 | OP_REG5_NEW_DATA_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG5 имеет новые данные. |
| 4 | OP_REG4_NEW_DATA_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG4 имеет новые данные. |
| 3 | OP_REG3_NEW_DATA_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG3 имеет новые данные. |
| 2 | OP_REG2_NEW_DATA_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG2 имеет новые данные. |
| 1 | OP_REG1_NEW_DATA_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG1 имеет новые данные. |
| 0 | OP_REG0_NEW_DATA_MASK | R/W | 0 | Разрешение прерывания от источника запроса: OP_REG0 имеет новые данные. |

### 21.15.10 *REGS_INT_SOURCE*

Таблица 673 – Описание бит регистра REGS_INT_SOURCE

| Base ADDR= | 0x4000_0000 0x4000_0080 | Offset= | 0x0000_005C | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | REGS_INT_SOURCE | | | | |

| 31…16 |
|---|
| Зарезервировано |
| |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OP_REG15_RD_BUSY | OP_REG14_RD_BUSY | OP_REG13_RD_BUSY | OP_REG12_RD_BUSY | OP_REG12_RD_BUSY | OP_REG12_RD_BUSY | OP_REG12_RD_BUSY | OP_REG12_RD_BUSY | OP_REG2_NEW_DATA | OP_REG2_NEW_DATA | OP_REG2_NEW_DATA | OP_REG2_NEW_DATA | OP_REG2_NEW_DATA | OP_REG2_NEW_DATA | OP_REG1_NEW_DATA | OP_REG0_NEW_DATA |
| 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 | R/W |

| Бит | Имя | Доступ | Значение | Описание |
|---|---|---|---|---|
| 31 … 16 | - | - | 0 | Зарезервировано |
| 15 | OP_REG15_RD_BUSY | R/W | 0 | OP_REG15: попытка записи в занятом состоянии. Очищается при записи 1. |
| 14 | OP_REG14_RD_BUSY | R/W | 0 | OP_REG14: попытка записи в занятом состоянии. Очищается при записи 1. |
| 13 | OP_REG13_RD_BUSY | R/W | 0 | OP_REG13: попытка записи в занятом состоянии. Очищается при записи 1. |
| 12 | OP_REG12_RD_BUSY | R/W | 0 | OP_REG12: попытка записи в занятом состоянии. Очищается при записи 1. |
| 11 | OP_REG11_RD_BUSY | R/W | 0 | OP_REG11: попытка записи в занятом состоянии. Очищается при записи 1. |
| 10 | OP_REG10_RD_BUSY | R/W | 0 | OP_REG10: попытка записи в занятом состоянии. Очищается при записи 1. |
| 9 | OP_REG9_RD_BUSY | R/W | 0 | OP_REG9: попытка записи в занятом состоянии. Очищается при записи 1. |
| 8 | OP_REG8_RD_BUSY | R/W | 0 | OP_REG8: попытка записи в занятом состоянии. Очищается при записи 1. |
| 7 | OP_REG7_NEW_DATA | R/W | 0 | OP_REG7 имеет новые данные. Очищается при чтении OP_REG7. |
| 6 | OP_REG6_NEW_DATA | R/W | 0 | OP_REG6 имеет новые данные. Очищается при чтении OP_REG6. |
| 5 | OP_REG5_NEW_DATA | R/W | 0 | OP_REG5 имеет новые данные. Очищается при чтении OP_REG5. |
| 4 | OP_REG4_NEW_DATA | R/W | 0 | OP_REG4 имеет новые данные. Очищается при чтении OP_REG4. |
| 3 | OP_REG3_NEW_DATA | R/W | 0 | OP_REG3 имеет новые данные. Очищается при чтении OP_REG3. |
| 2 | OP_REG2_NEW_DATA | R/W | 0 | OP_REG2 имеет новые данные. Очищается при чтении OP_REG2. |
| 1 | OP_REG1_NEW_DATA | R/W | 0 | OP_REG1 имеет новые данные. Очищается при чтении OP_REG1. |
| 0 | OP_REG0_NEW_DATA | R/W | 0 | OP_REG0 имеет новые данные. Очищается при чтении OP_REG0. |

Примечание – Для организации надежного сброса статусных флагов прерываний необходимо учитывать требования, изложенные в пункте 22.1.2 «Обработчики исключений».

## 21.16 Описание регистров контроллера OTP (OTP_CNTR)

Таблица 674 – Регистры контроллера OTP

| Базовый Адрес | | Название | Описание |
|---|---|---|---|
| 0x1011_0000 | | OTP_CNTR | |
| **Смещение** | | | |
| 0x0000 | 0 | STAT_CNTR | Регистр статуса и управления |
| 0x0004 | 1 | OTP_STAT | Регистр статуса блоков |
| 0x0008 | 2 | DELAY_0 | Регистр задержки 0 |
| 0x000c | 3 | DELAY_1 | Регистр задержки 1 |
| 0x0010 | 4 | RW_CMD | Регистр команды чтения-записи |
| 0x0014 | 5 | READ_DATA | Регистр считанных данных |
| 0x0018 | 6 | WRITE_PROTECT | Регистр защиты от записи |
| 0x001c | 7 | READ_PROTECT | Регистр защиты от чтения |
| 0x0020 | 8 | - | Зарезервировано |
| 0x0024 | 9 | - | Зарезервировано |

| Базовый Адрес | | Название | Описание |
|---|---|---|---|
| 0x1011_0000 | | OTP_CNTR | |
| **Смещение** | | | |
| 0x0028 | 10 | TEST_OPT | Регистр флагов тестовых опций |

### 21.16.1 Регистр статуса и управления STAT_CNTR

Таблица 675 – Описание бит регистра STAT_CNTR

| Base ADDR= | | | **0x1011_0000** | | | Offset= | | | **0x0000_0000** | | | Reset= | | | 0x0000_0000 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| REG Name: | | | STAT_CNTR | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | | |
| Зарезервировано | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | | DIR_EN | READ_PCRC | WRITE_PCRC | TEST_CRC | RW_ERROR | OTP_PLOCK | OTP_LOCK | BUSY | FSM_STATE | | |
| | | | r | r | rw | r | r | r | r | r | r | r | r | r | r |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…11 | Зарезервировано | 0 | Резерв |
| 10 | DIR_EN | 0 | Включение тестового режима. Не рекомендуется использовать. Всегда записывать 0. |
| 9 | READ_PCRC | 0 | Флаг состояния целостности данных регистра защиты от чтения (READ_PROTECT) |
| 8 | WRITE_PCRC | 0 | Флаг состояния целостности данных регистра защиты от записи (WRITE_PROTECT) |
| 7 | TEST_CRC | 0 | Флаг состояния целостности данных регистра специальных тестовых опций (TEST_OPT) |
| 6 | RW_ERROR | 0 | Флаг ошибки чтения или записи:<br>- при попытке чтения или записи во время установленного сигнала BUSY<br>- при попытке писать в защищенную от записи область или читать из защищенной от чтения области |
| 5 | OTP_PLOCK | 0 | Флаг частичной блокировки OTP от записи.<br>Единица в данном флаге означает что некоторые банки блока OTP недоступны для записи. |
| 4 | OTP_LOCK | 0 | Флаг блокировки OTP от записи.<br>Единица в данном флаге означает что все банки блока OTP недоступны для записи. |
| 3 | BUSY | 0 | Флаг занятости OTP.<br>OTP выполняет какие-либо действия и не готово принять очередной запрос на чтение или запись |
| 2…0 | FSM_STATE | 0 | Состояние конечного автомата управлением OTP:<br>0 - бездействие<br>1 - инициализация<br>2 - готовность к работе<br>3 - ожидание окончания чтения через регистры<br>4 - ожидание окончания чтения через шину<br>5 - ожидание окончания записи<br>6 - тестовый режим прямого управления OTP |

### 21.16.2 Регистр статуса блоков OTP_STAT_REG

Таблица 676 – Описание бит регистра OTP_STAT_REG

| Base ADDR= | **0x1011_0000** | Offset= | **0x0000_0004** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | OTP_STAT_REG | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OTP_3_BUSY | OTP_3_LOCK | Зарезер-вирован | OTP_3_FSM_STATE | | | | | OTP_2_BUSY | OTP_2_LOCK | Зарезер-вирован | OTP_2_FSM_STATE | | | | |
| r | r | | r | r | r | r | r | r | r | | r | r | r | r | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OTP_1_BUSY | OTP_1_LOCK | Зарезер-вировано | OTP_1_FSM_STATE | | | | | OTP_0_BUSY | OTP_0_LOCK | Зарезер-вировано | OTP_0_FSM_STATE | | | | |
| r | r | | r | r | r | r | r | r | r | r | r | r | r | r | r |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31 | OTP_3_BUSY | 0 | Флаг занятости: OTP 3 выполняет какие-либо действия и не готова принять очередной запрос на чтение или запись |
| 30 | OTP_3_LOCK | 0 | Флаг состояния аппаратной блокировки от записи блока OTP 0 |
| 29 | Зарезервировано | 0 | Резерв |
| 28...24 | OTP_3_FSM_STATE | 0 | Состояние конечного автомата блока OTP:<br>0 - бездействие<br>1 - переход в режим энергосбережения<br>2 - подготовка сброса<br>3 - сброс<br>4 - окончание сброса<br>5 - активный режим, ожидание чтения-записи<br>6 - пауза режима чтения после записи<br>7 - подготовка чтения<br>8 - чтение<br>9 - окончание чтения<br>10 - подготовка записи<br>11 - сохранение данных для записи<br>12 - окончание сохранения данных<br>13 - подготовка программирования<br>14 - подготовка схемы повышения напряжения<br>15 - программирование<br>16 - выключение схемы повышения напряжения<br>17 - ожидание окончания программирования<br>18 - ожидание окончания записи<br>19 - подготовка тестового режима прямого управления<br>20 - тестовый режим прямого управления |
| 23 | OTP_2_BUSY | 0 | Аналогично OTP_3_BUSY, но для блока OTP 2 |
| 22 | OTP_2_LOCK | 0 | Аналогично OTP_3_LOCK, но для блока OTP 2 |
| 21 | Зарезервировано | 0 | Резерв |
| 20…16 | OTP_2_FSM_STATE | 0 | Аналогично OTP_3_FSM_STATE, но для блока OTP 2 |
| 15 | OTP_1_BUSY | 0 | Аналогично OTP_3_BUSY, но для блока OTP 1 |
| 14 | OTP_1_LOCK | 0 | Аналогично OTP_3_LOCK, но для блока OTP 1 |
| 13 | Зарезервировано | 0 | Резерв |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 12…8 | OTP_1_FSM_STATE | 0 | Аналогично OTP_3_FSM_STATE, но для блока OTP 1 |
| 7 | OTP_0_BUSY | 0 | Аналогично OTP_3_BUSY, но для блока OTP 0 |
| 6 | OTP_0_LOCK | 0 | Аналогично OTP_3_LOCK, но для блока OTP 0 |
| 5 | Зарезервировано | 0 | Резерв |
| 4…0 | OTP_0_FSM_STATE | 0 | Аналогично OTP_3_FSM_STATE, но для блока OTP 0. |

### 21.16.3 Регистр задержки DELAY_0_REG

Таблица 677 – Описание бит регистра DELAY_0_REG

| Base ADDR= | **0x1011_0000** | Offset= | **0x0000_0008** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | DELAY_0_REG | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | | | | | | DELAY_01US | | | | | | | |
| | | | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Зарезервировано | | | DELAY_50NS | | | | | Зарезервировано | | | | DELAY_20NS | | | |
| | | | rw | rw | rw | rw | rw | | | | | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…25 | Зарезервировано | 0 | Резерв |
| 24…16 | DELAY_1US | 0 | Параметр задержки для формирования паузы 1 мкс. Величина задержки задается в тактах основной частоты работы модуля, пауза равна N+1 такту, где N – заданное в поле число. И должна быть не меньше 1 мкс и не более 2 мкс |
| 15…13 | Зарезервировано | 0 | Резерв |
| 12…8 | DELAY_50NS | 0 | Параметр задержки для формирования паузы 50 нс. Величина задержки задается в тактах основной частоты работы модуля, пауза равна N+1 такту, где N – заданное в поле число. И должна быть не меньше 50 нс |
| 7…4 | Зарезервировано | 0 | Резерв |
| 3…0 | DELAY_20NS | 0 | Параметр задержки для формирования паузы 20 нс. Величина задержки задается в тактах основной частоты работы модуля, пауза равна N+1 такту, где N – заданное в поле число. И должна быть не меньше 20 нс |

### 21.16.4 Регистр задержки DELAY_1_REG

Таблица 678 – Описание бит регистра DELAY_1_REG

| Base ADDR= | **0x1011_0000** | Offset= | **0x0000_000C** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | DELAY_1_REG | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| Зарезервировано | | | | | | | | | | | | | | | DELAY_16US |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DELAY_16US | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…17 | Зарезервировано | 0 | Резерв |
| 16…0 | DELAY_16US | 0 | Параметр задержки для формирования паузы 16 мкс. Величина задержки задается в тактах основной частоты работы модуля, пауза равна N+1 такту, где N – заданное в поле число. И должна быть не меньше 16 мкс и не больше 17 мкс |

### 21.16.5 Регистр команды чтения-записи RW_CMD_REG

Для правильной работы операций записи/чтения необходимо установить значения всех задержек в регистрах DELAY_0_REG и DELAY_1_REG.

Таблица 679 – Описание бит регистра RW_CMD_REG

| Base ADDR= | **0x1011_0000** | Offset= | **0x0000_0010** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | RW_CMD_REG | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RW_ERROR | Зарезер-вировано | WRITE | READ | WDATA_3 | WDATA_2 | WDATA_1 | WDATA_0 | Зарезервировано | | | | | | | ADDR |
| rw | | rw | rw | rw | rw | rw | rw | | | | | | | | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADDR | | | | | | | | | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31 | RW_ERROR | 0 | Ошибка выполнения последней операции чтения или записи. Ошибка возникает при попытке чтения или записи во время установленного сигнала **BUSY**.<br>При попытке писать в защищенную от записи область или читать из защищенной от чтения области |
| 30 | Зарезервировано | 0 | Резерв |
| 29 | WRITE | 0 | Команда на запись в указанный адрес, при наличии 1 в данном поле и 0 в поле команды чтения производиться процедура записи. Если |

| | | | |
|---|---|---|---|
| | | | выбранная область не защищена от записи OTP не занято, произойдет запись данных. Окончание контролируется по флагу BUSY в регистре статуса |
| 28 | READ | 0 | Команда на чтение из указанного адреса, при наличии 1 в данном поле и 0 в поле команды записи производиться процедура чтения. Если выбранная область не защищена от чтения и OTP не занято, произойдет чтение данных, считанные данные будут возвращены в регистр считанных данных. Окончание контролируется по флагу BUSY в регистре статуса |
| 27 | WDATA_3 | 0 | Данные для записи в OTP 3: 0 – не производить запись. Начальное состояние OTP – 0, записать можно только 1, стереть 1 нельзя |
| 26 | WDATA_2 | 0 | Аналогично WDATA_3 но для OTP 2 |
| 25 | WDATA_1 | 0 | Аналогично WDATA_3 но для OTP 1 |
| 24 | WDATA_0 | 0 | Аналогично WDATA_3 но для OTP 0 |
| 23…17 | Зарезервировано | 0 | Резерв |
| 16…0 | ADDR | 0 | Битовый адрес записи и чтения. Устанавливается для всех блоков OTP. Запись производиться битами, чтение производиться байтами. |

### 21.16.6 Регистр считанных данных READ_DATA_REG

Таблица 680 – Описание бит регистра READ_DATA_REG

| Base ADDR= | **0x1011_0000** | Offset= | **0x0000_0014** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | READ_DATA_REG | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RDATA_3 | | | | | | | | RDATA_2 | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RDATA_1 | | | | | | | | RDATA_0 | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…24 | RDATA_3 | 0 | Данные считанные из 3 банка OTP в случае успешного завершения процедуры чтения. При ошибках в поле заноситься константа 0xEF |
| 23…16 | RDATA_2 | 0 | Аналогично RDATA_3 но для OTP 2 |
| 15…8 | RDATA_1 | 0 | Аналогично RDATA_3 но для OTP 1 |
| 7…0 | RDATA_0 | 0 | Аналогично RDATA_3 но для OTP 0 |

### 21.16.7 Регистр защиты от записи WRITE_PROTECT_REG

Таблица 681 – Описание бит регистра WRITE_PROTECT_REG

| Base ADDR= | **0x1011_0000** | Offset= | **0x0000_0018** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | WRITE_PROTECT_REG | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WP_OTP_3 | | | | | | | | WP_OTP_2 | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WP_OTP_1 | | | | | | | | WP_OTP_0 | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…24 | WP_OTP_3 | 0 | Каждый бит регистра отвечает за область памяти в 2048 байт. 0 бит – за адреса 0 – 2047, 1 бит – за адреса 2048 – 4095 и так далее. 1 в бите запрещает запись в данный регион памяти. Биты защиты можно только установить, сбросить биты защиты можно только общим сбросом по питанию. Сброс процессора через регистры управления не снимает установленные биты.<br>При штатной записи данных в регистр, автоматически рассчитывается контрольная сумма для нового значения. По этой сумме контролируется целостность данных, на случай модификаций бит защиты сторонними методами |
| 23…16 | WP_OTP_2 | 0 | Аналогично WP_OTP_3 но для OTP 2 |
| 15…8 | WP_OTP_1 | 0 | Аналогично WP_OTP_3 но для OTP 1 |
| 7…0 | WP_OTP_0 | 0 | Аналогично WP_OTP_3 но для OTP 0 |

### 21.16.8 Регистр защиты от чтения READ_PROTECT_REG

Таблица 682 – Описание бит регистра READ_PROTECT_REG

| Base ADDR= | **0x1011_0000** | Offset= | **0x0000_001C** | Reset= | 0x0000_0000 |
|---|---|---|---|---|---|
| REG Name: | READ_PROTECT_REG | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RP_OTP_3 | | | | | | | | RP_OTP_2 | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RP_OTP_1 | | | | | | | | RP_OTP_0 | | | | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…24 | RP_OTP_3 | 0 | Каждый бит регистра отвечает за область памяти в 2048 байт. 0 бит – за адреса 0 – 2047, 1 бит – за адреса 2048 – 4095 и так далее. 1 в бите запрещает чтение из данного региона памяти. Биты защиты можно только установить, сбросить биты защиты можно только общим сбросом по питанию. Сброс процессора через регистры управления не снимает установленные биты.<br>При штатной записи данных в регистр, автоматически рассчитывается контрольная сумма для нового значения. По этой сумме контролируется целостность данных, на случай модификаций бит защиты сторонними методами |
| 23…16 | RP_OTP_2 | 0 | Аналогично WP_OTP_3 но для OTP 2 |
| 15…8 | RP_OTP_1 | 0 | Аналогично WP_OTP_3 но для OTP 1 |
| 7…0 | RP_OTP_0 | 0 | Аналогично WP_OTP_3 но для OTP 0 |

### 21.16.9 Регистр флагов тестовых опций TEST_OPT

Таблица 683 – Описание бит регистра TEST_OPT

Инициализируется автоматически из ячейки OTP с адресом 0 (securityword). Для того что бы запретить опцию необходимо записать в соответствующий бит слова OTP единицу.

| Base ADDR= | **0x1011_0000** | Offset= | **0x0000_0028** | Reset= | ~securityword |
|---|---|---|---|---|---|
| REG Name: | TEST_OPT | | | | |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ATPG_EN [31:16] | | | | | | | | | | | | | | | |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ATPG_EN [15:8] | | | | | | | | DT_ACK_EN1 | DT_RESET_REL3 | DT_ACK_EN0 | DT_RESET_REL2 | DT_REQ_EN1 | DT_RESET_REL1 | DT_REQ_EN0 | DT_RESET_REL0 |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |

| Бит | Имя | Значение | Описание |
|---|---|---|---|
| 31…8 | ATPG_EN | ~securityword[31:8] | 1 в любом бите поля securityword [31:8] запрещает ATPG и переключение выводов PC [31:30] в режим SWD |
| 7 | DT_ACK_EN1<br>DT_DP_ENABLE3 | ~securityword[7] | 1 в соответствующем бите securityword запрещает подтверждение включения модуля DAP и запрещает работу модуля DP |
| 6 | DT_RESET_REL3<br>DT_DP_ENABLE2 | ~securityword[6] | 1 в соответствующем бите securityword запрещает снятие сигнала сброса с модуля DAP и запрещает работу модуля DP |
| 5 | DT_ACK_EN0 | ~securityword[5] | 1 в соответствующем бите securityword запрещает подтверждение включения модуля DAP |
| 4 | DT_RESET_REL2 | ~securityword[4] | 1 в соответствующем бите securityword запрещает снятие сигнала сброса с модуля DAP |
| 3 | DT_REQ_EN1 | ~securityword[3] | 1 в соответствующем бите securityword запрещает запрос включения модуля DAP |
| 2 | DT_RESET_REL1 | ~securityword[2] | 1 в соответствующем бите securityword запрещает снятие сигнала сброса с модуля DAP |
| 1 | DT_REQ_EN0<br>DT_DP_ENABLE1 | ~securityword[1] | 1 в соответствующем бите securityword запрещает запрос включения модуля DAP и запрещает работу модуля DP |
| 0 | DT_RESET_REL0<br>DT_DP_ENABLE0 | ~securityword[0] | 1 в соответствующем бите securityword запрещает снятие сигнала Сброс с модуля DAP и запрещает работу модуля DP |

# 22 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** – исключении начало обрабатываться процессором, но появилось новое исключении в состоянии Pending от того же источника.

## 22.1 Прерывания и исключения (Cortex-M0)

### 22.1.1 *Типы исключений*

Исключения бывают следующих типов:
- RESET;
- NON MASKABLE INTERRUPT (NMI);
- Hard Fault;
- SVCall;
- PendSV;
- SysTick;
- Прерывания (IRQ).

#### 22.1.1.1 RESET

RESET вызывается при включении питания и горячем сбросе. Модель исключений трактует RESET как специальная форма 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 в Thread режиме.

#### 22.1.1.2 NON MASKABLE INTERRUPT (NMI)

Не маскируемое прерывание (NMI) может быть вызвано периферией или установлено программой. Это самое высокоприоритетное исключение после сброса. Всегда разрешено и имеет фиксированный приоритет -2.

NMI не может быть:
- замаскировано или предотвращено от активации из другого исключения;
- прерывает любые исключения кроме RESET.

#### 22.1.1.3 Hard Fault

Hard Fault исключение происходит при ошибке во время обработки исключений или потому, что исключение не может быть обработано каким-либо другим механизмом. Hard Fault имеет фиксированный приоритет -1, означающий, что он имеет больший приоритет чем любое из исключений с конфигурируемым приоритетом.

#### 22.1.1.4 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.

#### 22.1.1.5 PendSV

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

#### 22.1.1.6 SysTick

Исключение SysTick является исключением, генерируемым 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.

#### 22.1.1.7 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 684 – Таблица различных типов исключений

| Номер исключения | IRQ номер | Тип | Приоритет | Адрес вектора обработчика (смещение) | Активация |
|---|---|---|---|---|---|
| 1 | - | RESET | -3, наивысший | 0x0000_0004 | Асинхронный |
| 2 | -14 | NMI | -2 | 0x0000_0008 | Асинхронный |
| 3 | -13 | Hard Fault | -1 | 0x0000_000C | - |
| 4-10 | - | Reserved | - | 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.

### 22.1.2 *Обработчики исключений*

Для обработки исключений используются:

**Interrupt Service Routines (ISRs)** – Прерывания с IRQ0 по IRQ31 обрабатываются ISRs.

Если программный код, сбрасывающий флаг прерывания на уровне периферийного блока, находится в конце функции обработчика прерывания, ядро может не успеть выполнить очистку флага в момент выхода из обработчика, из-за чего прерывание будет вызвано повторно. Чтобы этого избежать, необходимо после сброса

флага прерывания сделать фиктивное чтение, например, регистра статусных флагов периферийного блока в отдельную переменную.

**Fault Handlers** – Обрабатываются только Fault Handlers.

**System Handlers** – NMI, PendSV, SVCall, SysTick и HardFault обрабатываются System Handlers.

#### 22.1.2.1 Таблица векторов

Таблица векторов содержит указатель стека, вектор входа по RESET и стартовые адреса обработчиков, также называемых векторами. На рисунке 227 представлена последовательность векторов в таблице. Младший бит всех векторов должен быть равен 1, указывая на то, что обработчик выполняется в Thumb-режиме.

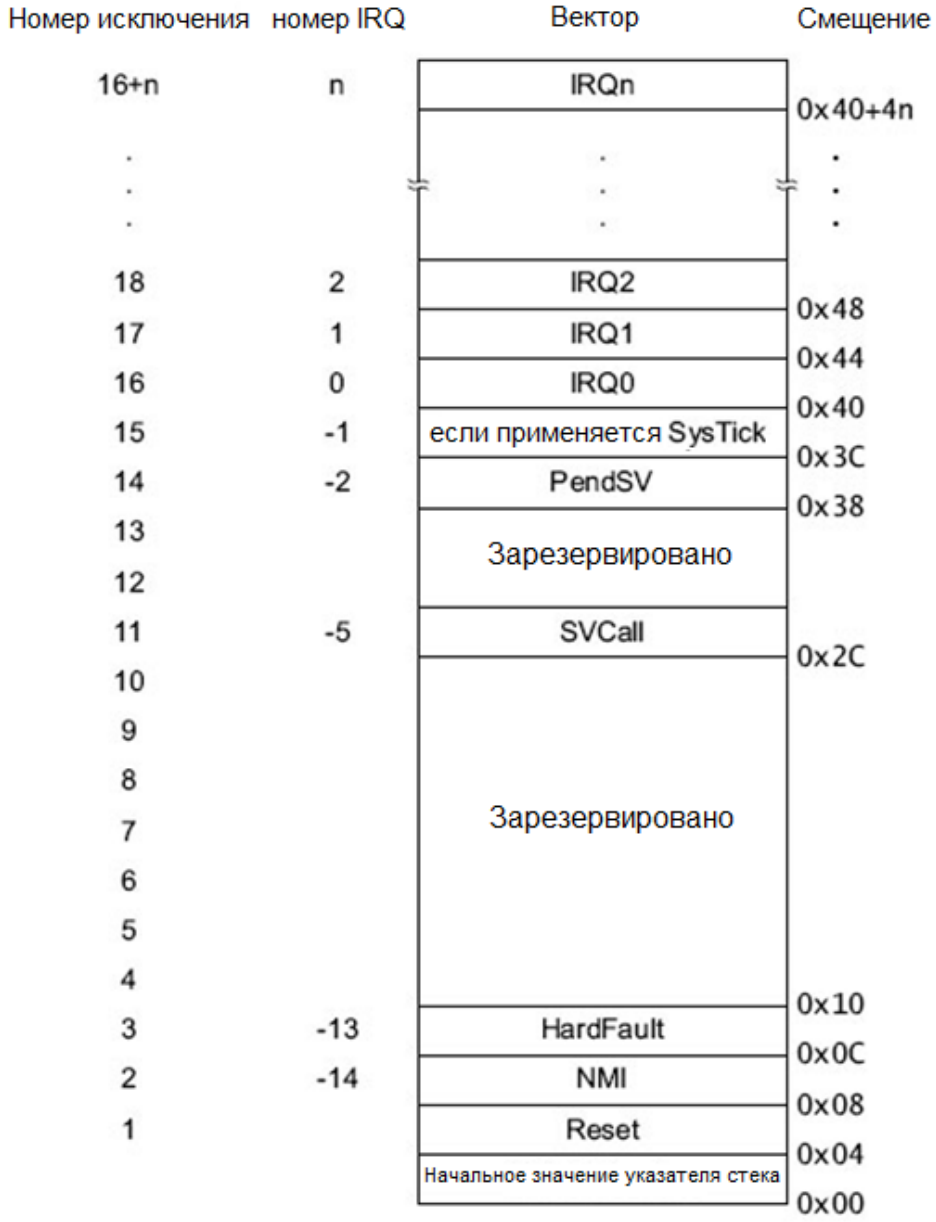



Рисунок 227 – Таблица векторов

При системном сбросе, таблица векторов располагается по фиксированному адресу 0x00000000.

### 22.1.3 *Приоритеты исключений*

Меньшее 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 192 с шагом 64. 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.

### 22.1.4 *Вход в обработчик и выход из обработчика*

При описании используются следующие термины:

#### 22.1.4.1 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. В случае, если внутри обработчика исключения возникает прерывание более высокого приоритета возникает ситуация, называемая вложенным исключением. Подробнее данный вопрос рассмотрен в подразделе «Вход в процедуру обработки исключения».

#### 22.1.4.2 Возврат

Возврат из 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 подразделе «Возврат из обработчика исключения».

#### 22.1.4.3 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 присутствуют исключения, требующие вызова обработчика, восстановление состояния процессора из стека не производится, а управление передается непосредственно на новый обработчик.

#### 22.1.4.4 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.

#### 22.1.4.5 Вход в процедуру обработки исключения

Вызов процедуры обработки исключения возникает в случае наличия необработанных исключительных ситуаций с достаточным приоритетом и выполнения 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.

Для того, чтобы исключительная ситуация имела достаточный приоритет, необходимо, чтобы уровень ее приоритета был выше значений, заданных в регистрах маскирования. В противном случае исключение находится в состоянии ожидания, процедура его обработки не вызывается.

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.

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 значений 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 обработчика высокоприоритетной исключительной ситуации без изменения состояния отложенного прерывания в соответствии с правилами, изложенными в разделе «Запоздавшее исключение».

#### 22.1.4.6 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.

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 содержать 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 685.

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 685 – Возврат из обработчика исключения

| EXC_RETURN [3:0] | Описание |
|---|---|
| bXXX0 | Резерв. |
| B0001 | Возврат в режим обработчика.<br>Восстановление контекста осуществляется из стека MSP.<br>Дальнейшая работа осуществляется со стеком MSP. |
| B0011 | Резерв. |
| B01X1 | Резерв. |
| B1001 | Возврат в режим приложения.<br>Восстановление контекста осуществляется из стека MSP.<br>Дальнейшая работа осуществляется со стеком MSP. |
| B1101 | Возврат в режим приложения.<br>Восстановление контекста осуществляется из стека PSP.<br>Дальнейшая работа осуществляется со стеком PSP. |
| B1X11 | Резерв. |

### 22.1.5 *Управление электропитанием*

В процессоре Cortex-M0 предусмотрены следующие режимы ожидания (пониженного энергопотребления):

**Sleep** – останов синхросигнала для процессора;

**Sleep deep**– останов синхросигнала для процессора, PLL и Flash.

Выбор процессором конкретного режима ожидания определяется значением бита SLEEPDEEP регистра SCR (см. подраздел «SCB->SCR»).

Далее в разделе описаны механизмы перехода в режим пониженного энергопотребления и условия выхода из этого режима.

#### 22.1.5.1 Переход в режим пониженного энергопотребления

Система может формировать ложные сигналы событий, выводящие процессор из ожидания, например, они возникают при работе отладчика. Следовательно, программное обеспечение должно быть способно перевести процессор обратно в указанный ожидания. Для этого можно, например, организовать в программе пустой цикл.

#### 22.1.5.2 Ожидание прерывания

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

#### 22.1.5.3 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.

#### 22.1.5.4 Переход в режим ожидания по выходу из обработчика исключения (режим sleep-on-exit)

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 процессора используется только для обработки исключений.

#### 22.1.5.5 Выход из состояния ожидания

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

#### 22.1.5.6 Выход из ожидания по команде WFI и в режиме sleep-on-exit

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. 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.

#### 22.1.5.7 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 оно запрещено или имеет приоритет, недостаточно высокий для запуска обработчика.

Более подробная информация о регистре SCR представлена в разделе «SCB->SCR».

#### 22.1.5.8 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
```

## 22.2 Прерывания и исключения (Cortex-M4)

### 22.2.1 *Типы исключений*

Исключения бывают следующих типов:
- RESET;
- NON MASKABLE INTERRUPT (NMI);
- Hard Fault;
- Memory Management fault;
- Bus Fault;
- Usage Fault;
- SVCall;
- PendSV;
- SysTick;
- Прерывания (IRQ).

#### 22.2.1.1 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.

#### 22.2.1.2 NON MASKABLE INTERRUPT (NMI)

Немаскируемое прерывание (NMI) может быть вызвано периферией или установлено программой. Это самое высокоприоритетное исключение после сброса. Всегда разрешено и имеет фиксированный приоритет - 2.

Примечание – Данное прерывание не реализовано.

NMI не может быть:
- замаскировано или предотвращено от активации из другого исключения;
- прерывает любые исключения, кроме RESET.

#### 22.2.1.3 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.

#### 22.2.1.4 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.

#### 22.2.1.5 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.

#### 22.2.1.6 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.

#### 22.2.1.7 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.

#### 22.2.1.8 PendSV

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

#### 22.2.1.9 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

#### 22.2.1.10 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 686 – Различные типы исключений

| Номер исключения | Номер IRQ | Тип | Приоритет | Адрес вектора обработчика (смещение) | Активация |
|---|---|---|---|---|---|
| 1 | - | RESET | -3, наивысший | 0x0000_0004 | Асинхронный |
| 2 | -14 | NMI | -2 | 0x0000_0008 | Асинхронный |
| 3 | -13 | Hard Fault | -1 | 0x0000_000C | - |
| 4 | -12 | Memory Management Fault | Конфигурируемый | 0x0000_0010 | Синхронный |
| 5 | -11 | Bus Fault | Конфигурируемый | 0x0000_0014 | Синхронный/ Асинхронный |
| 6 | -10 | Usage Fault | Конфигурируемый | 0x0000_0018 | Синхронный |
| 7-10 | - | - | - | Зарезервировано | - |
| 11 | -5 | SVCall | Конфигурируемый | 0x0000_002C | Синхронный |
| 12-13 | - | - | - | Зарезервировано | - |
| 14 | -2 | PendSV | Конфигурируемый | 0x0000_0038 | Асинхронный |
| 15 | -1 | SysTick | Конфигурируемый | 0x0000_003C | Асинхронный |
| 16 и выше | 0 и выше | IRQ | Конфигурируемый | 0x0000_0040 и выше | Асинхронный |

Для асинхронных исключений, кроме RESET, процессор может выполнить другие инструкции между возникновением сигнала исключения и входом в обработчик.

Программа в Privileged режиме может запретить прерывания, имеющие конфигурируемый приоритет.

### 22.2.2 *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.

#### 22.2.2.1 Таблица векторов

Таблица векторов содержит указатель стека, вектор входа по RESET и стартовые адреса обработчиков, также называемых векторами. Рисунок 228 представляет последовательность векторов в таблице. Младший бит всех векторов должен быть равен 1, указывая на то, что обработчик выполняется в Thumb режиме.

| Exception number | IRQ number | Offset | Vector |
|---|---|---|---|
| 47 | 31 | 0×00BC | IRQ31 |
| . | . | . | |
| . | . | . | |
| . | . | 0×004C | |
| 18 | 2 | 0×0048 | IRQ2 |
| 17 | 1 | 0×0044 | IRQ1 |
| 16 | 0 | 0×0040 | IRQ0 |
| 15 | -1 | 0×003C | Systick |
| 14 | -2 | 0×0038 | PendSV |
| 13 | | | Reserved |
| 12 | | | Reserved for Debug |
| 11 | -5 | 0×002C | SVCall |
| 10 | | | Reserved |
| 9 | | | |
| 8 | | | |
| 7 | | | |
| 6 | -10 | 0×0018 | Usage fault |
| 5 | -11 | 0×0014 | Bus fault |
| 4 | -12 | 0×0010 | Memory management fault |
| 3 | -13 | 0×000C | Hard fault |
| 2 | -14 | 0×0008 | Reserved |
| 1 | | 0×0004 | Reset |
| | | 0×0000 | Initial SP value |

Рисунок 228 – 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.

### 22.2.3 *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.

#### 22.2.3.1 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.

### 22.2.4 *Вход в обработчик и выход из обработчика*

При описании используются следующие термины:

#### 22.2.4.1 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».

#### 22.2.4.2 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»

#### 22.2.4.3 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.

#### 22.2.4.4 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.

#### 22.2.4.5 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)».

#### 22.2.4.6 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 687.

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 687 – Возврат из обработчика исключения

| EXC_RETURN[3:0] | Описание |
|---|---|
| bXXX0 | Зарезервирован |
| b0001 | Возврат в режим обработчика.<br>Восстановление контекста осуществляется из стека MSP.<br>Дальнейшая работа осуществляется со стеком MSP |
| b0011 | Зарезервирован |
| b01X1 | Зарезервирован |
| b1001 | Возврат в режим приложения.<br>Восстановление контекста осуществляется из стека MSP.<br>Дальнейшая работа осуществляется со стеком MSP |
| b1101 | Возврат в режим приложения.<br>Восстановление контекста осуществляется из стека PSP.<br>Дальнейшая работа осуществляется со стеком PSP |
| b1X11 | Зарезервирован |

### 22.2.5 *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.

#### 22.2.5.1 Типы отказов

В таблице 688 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 688 – Отказы

| Отказ | Обработчик | 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.

#### 22.2.5.2 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.

#### 22.2.5.3 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 689.

Таблица 689 – 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”<br>“Регистр адреса отказа доступа к памяти” |
| Отказ доступа к шине | BFSR | BFAR | “Регистр состояния отказов доступа к шине”<br>“Регистр адреса отказа доступа к шине” |
| Отказ, вызванный ошибками программирования | UFSR | - | “Регистр состояния отказов, вызванных ошибками программирования” |

#### 22.2.5.4 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.

### 22.2.6 *Управление электропитанием*

В процессоре Cortex-M4 предусмотрены следующие режимы ожидания (пониженного энергопотребления):

- Deep Sleep;
- Sleep;
- Standby.

Выбор процессором конкретного режима ожидания определяется значением бита SLEEPDEEP регистра SCR (см. “SCB->SCR”).

Далее в разделе описаны механизмы перехода в режим пониженного энергопотребления и условия выхода из этого режима.

#### 22.2.6.1 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.

#### 22.2.6.2 Ожидание прерывания

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

#### 22.2.6.3 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 работу без перехода в режим ожидания.

#### 22.2.6.4 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.

#### 22.2.6.5 Выход из состояния ожидания

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

***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.

***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 “SCB->SCR”.

#### 22.2.6.6 Рекомендации по программированию режима энергопотребления

В стандарте ANSI языка С 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
```

# 23 Контроллеры прерываний NVIC

## 23.1 Контроллер прерываний NVIC (Cortex-M0)

В разделе описан векторный контроллер прерываний с возможностью вложения (NVIC – Nested Vectored Interrupt Controller) и используемые им регистры.

Контроллер обеспечивает поддержку:

- программное задание уровня приоритета в диапазоне от 0 до 192 с шагом 64 независимо каждому прерыванию. Более высокое значение соответствует меньшему приоритету, таким образом, уровень 0 отвечает наивысшему приоритету прерывания;
- срабатывание сигнала прерывания по импульсу и по уровню;
- передача управления из одного обработчика исключения на 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 таблице 690.

Таблица 690 – Обобщенная информация о регистрах контроллера NVIC

| Адрес | Имя | Тип | Значение после сброса | Описание |
|---|---|---|---|---|
| 0xE000E100 | ISER | RW | 0x00000000 | Регистр разрешения прерываний |
| 0xE000E180 | ICER | RW | 0x00000000 | Регистр запрета прерывания |
| 0xE000E200 | ISPR | RW | 0x00000000 | Регистр перевода прерывания в состояние ожидания обслуживания |
| 0xE000E280 | ICPR | RW | 0x00000000 | Регистр сброса состояния ожидания обслуживания |
| 0xE000E400 – 0xE000E41C | IPR0-7 | RW | 0x00000000 | Регистр приоритета прерываний |

### 23.1.1 *Логика работы прерываний контроллера NVIC.*

В данном разделе описывается функционирование контроллера NVIC при поступлении на его вход запросов прерываний IRQ от различных модулей периферии микроконтроллера.

Первоначальным условием работы прерывания является его разрешение в модуле NVIC. За это отвечают регистры:

ISER – за разрешение прерываний,

ICER – за запрет прерываний.

В случае, когда соответствующий запрос разрешен (при данном условии рассмотрены все диаграммы в разделе), и приходит сигнал активации прерывания – запрос IRQ request, то возникает признак отложенного прерывания IRQ pending. Данный признак переводит прерывание в состояние ожидания его обработки ядром.

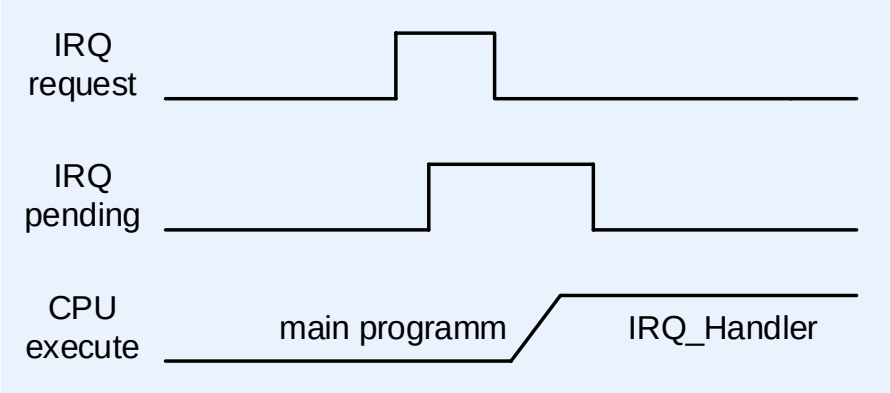



Рисунок 229 – Выставление отложенного запроса на прерывание и последующая его обработка

Pending биты выставляются в регистрах ISPR/ICPR, которые в свою очередь позволяют программно управлять признаком отложенного прерывания. ISPR – для установки pending бит, ICPR – для сброса соответственно. Если после прихода запроса на прерывание IRQ request, сбросить pending бит в регистре ICPR до того, как ядро приступит к его обработке, то прерывание будет проигнорировано - рисунок 230.

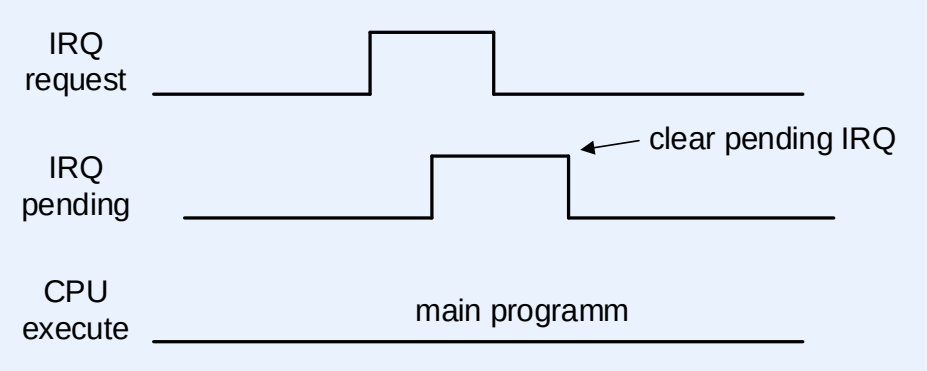



Рисунок 230 – Сброс признака отложенного прерывания, до обработки ядром

Если произойдёт снятие запроса IRQ request от источника, «защелкивание» признака отложенного прерывания гарантирует отработку его ядром в соответствии с приоритетом. – рисунок 231. Сам IRQ pending признак снимается автоматически, когда прерывание становится активным, о чём сигнализирует признак IRQ active.

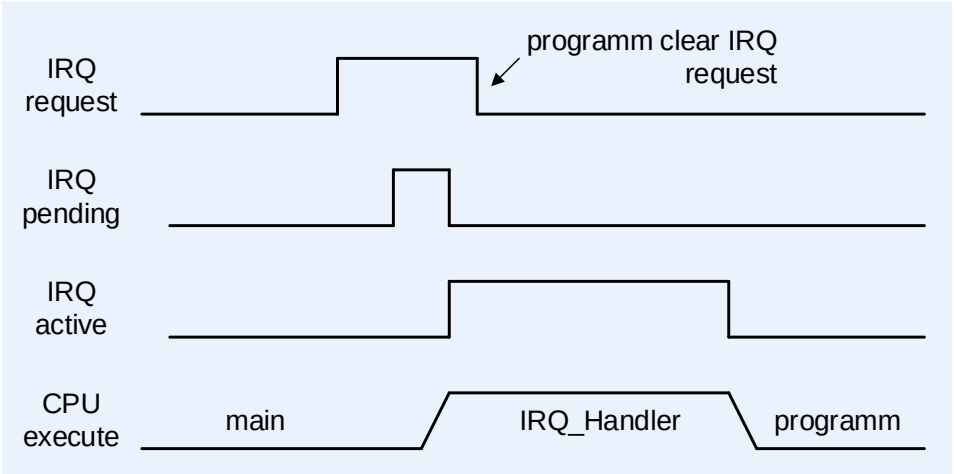



Рисунок 231 – Сброс признака отложенного прерывания, до обработки ядром

После того как прерывание стало активным, повторно запустить обработчик того же прерывания будет невозможно до тех пор, пока не будет завершена процедура

обработки прерывания командой выхода из исключения. После выполнения команды выхода происходит сброс признака активности IRQ active.

При удержании источником на входе NVIC запроса на обработку IRQ request, по окончании обработки прерывания и снятия признака активного прерывания IRQ active, происходит повторное выставление признака отложенного прерывания IRQ pending – «защелкивание» pending бита, сброс которого в дальнейшем инициирует повторную активность и обработку того же исключения – рисунок 232.

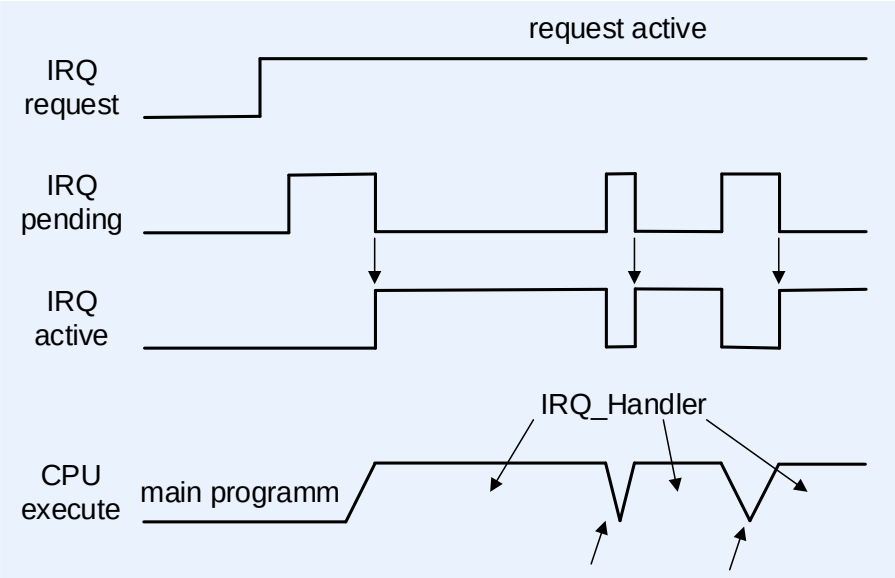



Рисунок 232 – Повторная обработка прерываний при удержании запроса от источника

Необходимо учитывать, что если источник прерываний выдает многократную установку и снятие запроса IRQ request на входе контроллера NVIC, то в таком случае только первый запрос выставляет признак отложенного прерывания IRQ pending, а остальные запросы до начала процедуры обработки прерывания (в момент активного признака отложенного прерывания) будут проигнорированы ядром – рисунок 233.

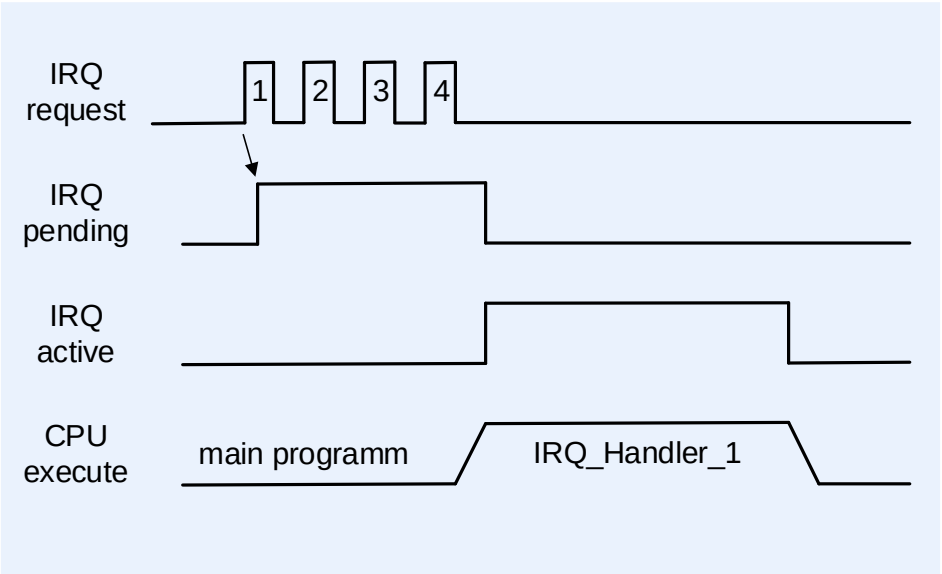



Рисунок 233 – Многократная установка снятие запроса IRQ request

Если запрос на прерывание пришел в момент активного прерывания, то в такой ситуации уже будут отработаны оба запроса на прерывание. В отличие от случая, изображенного на рисунке 233, запрос приходит тогда, когда признак отложенного прерывания IRQ pending уже сброшен, и новый запрос как раз его выставляет, что в дальнейшем позволяет провести повторную обработку прерывания – рисунок 234.

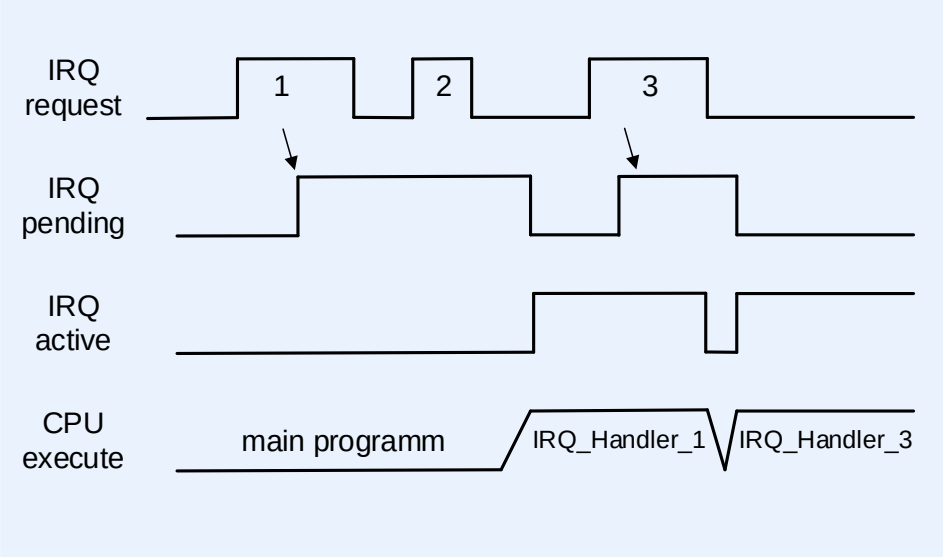



Рисунок 234 – Повторная установка запроса на прерывание в момент выполнения обработчика исключения

Выставление признака отложенного прерывания возможно даже в тех случаях, когда соответствующее прерывание запрещено. Все отложенные прерывания будут отражены в ISPR/ICPR, и в случае разрешения таких прерываний регистром ISPR, ядро тут же приступит к их обработке. Рекомендуется перед разрешением соответствующего прерывания убедиться в отсутствии признака отложенного запроса и, при необходимости, сбросить его.

### 23.1.2 *Регистр разрешения прерываний*

Регистр ISER предназначен для разрешения прерываний (запись) и определения, какие из прерываний разрешены (чтение). Более подробная информация представлена в таблице 690.

Распределение битов регистра представлено на рисунке 235.

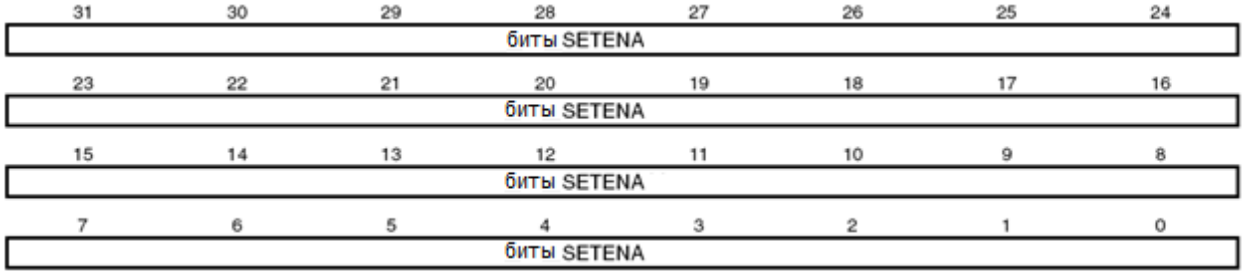



Рисунок 235 – Распределение битов регистра ISER

Таблица 691 – Регистр разрешения прерываний

| Биты | Поле | Функция |
|---|---|---|
| 31…0 | SETENA | Биты разрешения прерывания.<br>**При записи:**<br>1 – разрешение прерывания;<br>0 – не оказывает влияния.<br>**При чтении:**<br>1 – прерывание разрешено;<br>0 – прерывание запрещено. |

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.

### 23.1.3 *Регистр запрета прерываний*

Регистр ICER предназначен для запрета прерываний (запись) и определения, какие из прерываний разрешены (чтение). Более подробная информация представлена в таблице 690.

Распределение битов регистра представлено на рисунке 236.




Рисунок 236 – Распределение битов регистра ICER

Таблица 692 – Регистр запрета прерываний

| Биты | Поле | Функция |
|---|---|---|
| 31…0 | CLRENA | Биты запрещения прерывания.<br>**При записи:**<br>1 – запрещает прерывание;<br>0 – не оказывает влияния.<br>**При чтении:**<br>1 – прерывание разрешено;<br>0 – прерывание запрещено. |

### 23.1.4 *Регистр установки состояния ожидания для прерывания*

Регистр ISPR предназначен для принудительного перевода прерываний в состояние ожидания обслуживания (запись) и определения, какие из прерываний находятся в этом состоянии (чтение).

Более подробная информация представлена в таблице 690.

Распределение битов регистра представлено на рисунке 237.

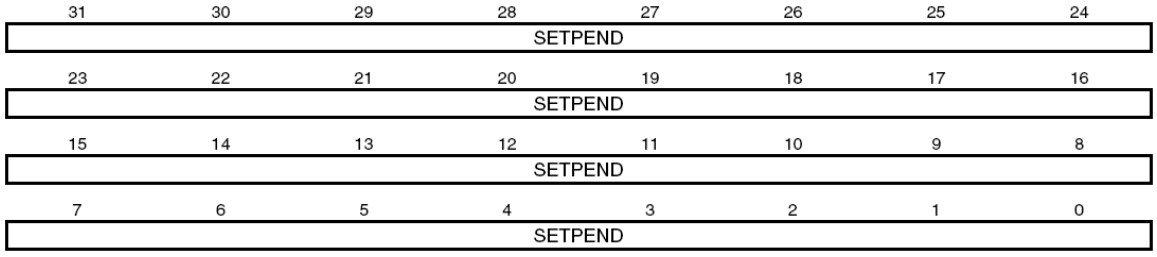



Рисунок 237 – Распределение битов регистра ISPR

Таблица 693 – Регистр установки состояния ожидания для прерывания

| Биты | Поле | Функция |
|---|---|---|
| 31…0 | SETPEND | **При записи:**<br>1 – перевод прерывания в состояние ожидания;<br>0 – не оказывает влияния.<br>**При чтении:**<br>1 – прерывание в состоянии ожидания обслуживания;<br>0 – прерывание не в состоянии ожидания обслуживания. |

Запись 1 в бит регистра ISPR, соответствующий:
– прерыванию, уже ожидающему обслуживания – не влияет на работу системы;
– запрещенному прерыванию – переводит его в состояние ожидания.

### 23.1.5 *Регистр сброса состояния ожидания для прерывания*

Регистр ICPR предназначен для принудительного сброса состояния ожидания обслуживания прерывания (запись) и определения, какие из прерываний находятся в состоянии ожидания (чтение).

Более подробная информация представлена в таблице 690.

Распределение битов регистра представлено на рисунке 238.

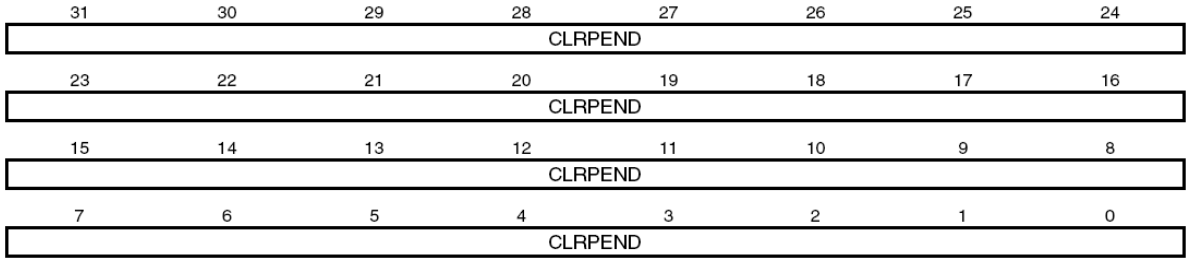



Рисунок 238 – Распределение битов регистра ICPR

Таблица 694 – Регистр сброса состояния ожидания обслуживания

| Биты | Поле | Функция |
|---|---|---|
| 31…0 | CLRPEND | **При записи:**<br>1 – сбрасывает состояние ожидания обслуживания;<br>0 – не оказывает влияния.<br>**При чтении:**<br>1 – прерывание в состоянии ожидания обслуживания;<br>0 – прерывание не в состоянии ожидания обслуживания. |

Запись 1 в разряд регистра ICPR, соответствующего прерыванию в активном состоянии, не влияет на работу системы.

### 23.1.6 *Регистры приоритета прерываний*

Регистры IPR0-IPR7 представляют собой набор 8-битовых полей, каждое из которых соответствует одному прерыванию. Регистры доступны пословно. Обобщенная информация об их характеристиках представлена в таблице 690.

Каждый из регистров содержит четыре поля приоритета, которые отображаются на четыре элемента массива PRI[0] ... PRI[31] CMSIS, как показано 239.

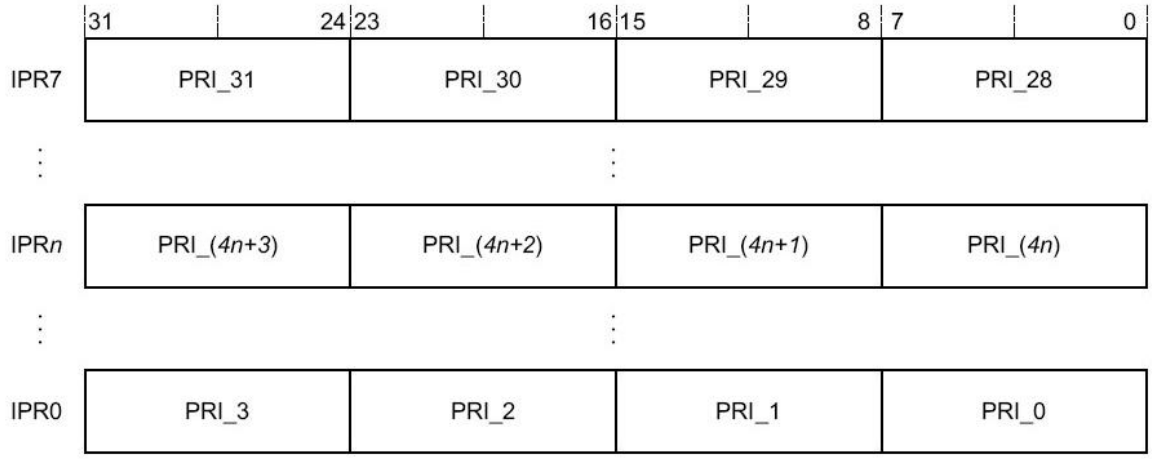



Рисунок 239 – Назначение бит

Каждое поле содержит значение приоритета в диапазоне от 0 до 192, причем меньшие значения соответствуют более высокому приоритету соответствующего прерывания. Процессор обеспечивает доступ только к битам [7:6] приоритета, биты [5:0] при чтении всегда равны нулю, а при записи игнорируются. Поэтому, например, запись 255 в регистр запишется как 192.

Для того, чтобы определить номер регистра IPR и смещение данных в регистре необходимо выполнить следующие операции:

для заданного номера прерывания N 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].

### 23.1.7 *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. подраздел «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.

### 23.1.8 *Аппаратное и программное управление прерываниями*

Процессор Cortex-M0 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 не активно;
- контроллер прерываний NVIC обнаруживает передний фронт сигнала запроса прерывания;
- программное обеспечение осуществляет запись в соответствующий разряд регистра ISPR0 или соответствующего значения в регистр 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.

### 23.1.9 *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.

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 695).

Таблица 695 – Функции CMSIS для управления контроллером прерываний

| **Функция** | **Описание** |
|---|---|
| void NVIC_EnableIRQ(IRQn_t IRQn) | Разрешить IRQn |
| void NVIC_DisableIRQ(IRQn_t IRQn) | Запретить IRQn |
| uint32_t NVIC_GetPendingIRQ (IRQn_t IRQn) | Вернуть истину, если прерывание IRQn ожидает обслуживания, ложь – 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 |
| 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.

## 23.2 Контроллер прерываний NVIC (Cortex-M4)

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 таблице 696.

Таблица 696 – Обобщенная информация о регистрах контроллера NVIC

| Адрес | Название | Тип | Доступ | Значение после сброса | Описание |
|---|---|---|---|---|---|
| 0xE000E100 | NVIC | | | | Контроллер прерываний NVIC |
| 0x000 | ISER[0] | RW | Привилегированный | 0x00000000 | Регистр разрешения прерываний ISER |
| … | | | | | |
| 0x01C | ISER[7] | | | | |
| … | | | | | |
| 0x080 | ICER[0] | RW | Привилегированный | 0x00000000 | Регистр запрета прерываний ICER |
| … | | | | | |
| 0x09C | ICER[7] | | | | |
| … | | | | | |
| 0x100 | ISPR[0] | RW | Привилегированный | 0x00000000 | Регистр установки состояния ожидания для прерывания ISPR |
| … | | | | | |
| 0x11C | ISPR[7] | | | | |
| … | | | | | |
| 0x180 | ICPR[0] | RW | Привилегированный | 0x00000000 | Регистр сброса состояния ожидания для прерывания ICPR |
| … | | | | | |
| 0x19C | ICPR[7] | | | | |
| … | | | | | |
| 0x200 | IABR[0] | RO | Привилегированный | 0x00000000 | Регистр активных прерываний IABR |
| … | | | | | |
| 0x21C | IABR[7] | | | | |
| … | | | | | |
| 0x300 | IP[3],IP[2],IP[1],IP[0] | RW | Привилегированный | 0x00000000 | Регистр приоритета прерываний IP |
| … | | | | | |
| 0x3F0 | IP[239], IP[238], IP[237], IP[236] | | | | |
| … | | | | | |
| 0xE00 | STIR | WO | В зависимости от конфигурации*) | 0x00000000 | Регистр программного формирования прерываний STIR |

* Более подробную информацию см. в описании регистра.

### 23.2.1 *Логика работы прерываний контроллера NVIC.*

В данном разделе описывается функционирование контроллера NVIC при поступлении на его вход запросов прерываний IRQ от различных модулей периферии микроконтроллера.

Первоначальным условием работы прерывания является его разрешение в модуле NVIC. За это отвечают регистры:

ISER – за разрешение прерываний,

ICER – за запрет прерываний.

В случае, когда соответствующий запрос разрешен (при данном условии рассмотрены все диаграммы в разделе), и приходит сигнал активации прерывания – запрос IRQ request, то возникает признак отложенного прерывания IRQ pending. Данный признак переводит прерывание в состояние ожидания его обработки ядром.

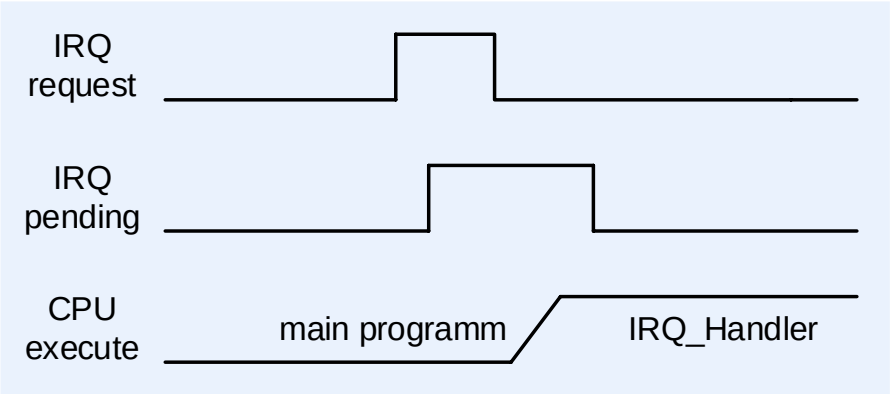



Рисунок 240 – Выставление отложенного запроса на прерывание и последующая его обработка

Pending биты выставляются в регистрах ISPR/ICPR, которые в свою очередь позволяют программно управлять признаком отложенного прерывания. ISPR – для установки pending бит, ICPR – для сброса соответственно. Если после прихода запроса на прерывание IRQ request, сбросить pending бит в регистре ICPR до того, как ядро приступит к его обработке, то прерывание будет проигнорировано - рисунок 230.

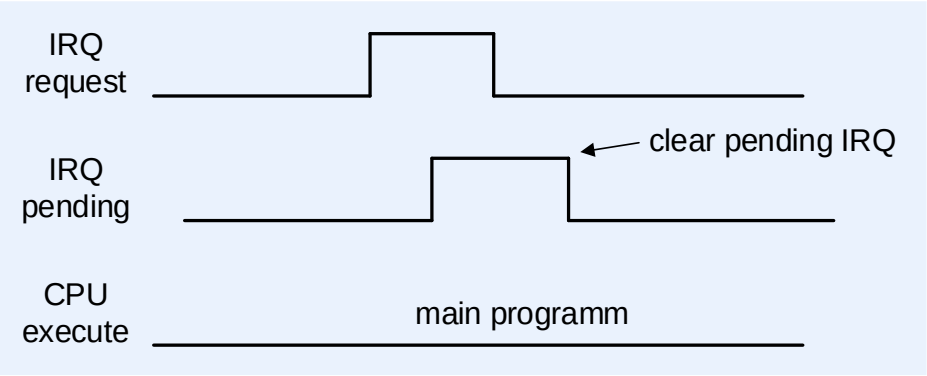



Рисунок 241 – Сброс признака отложенного прерывания, до обработки ядром

Если произойдёт снятие запроса IRQ request от источника, «защелкивание» признака отложенного прерывания гарантирует отработку его ядром в соответствии с приоритетом. – рисунок 231. Сам IRQ pending признак снимается автоматически, когда прерывание становится активным, о чём сигнализирует признак IRQ active. Информация об активности соответствующего прерывания содержится в регистре IABR[x].

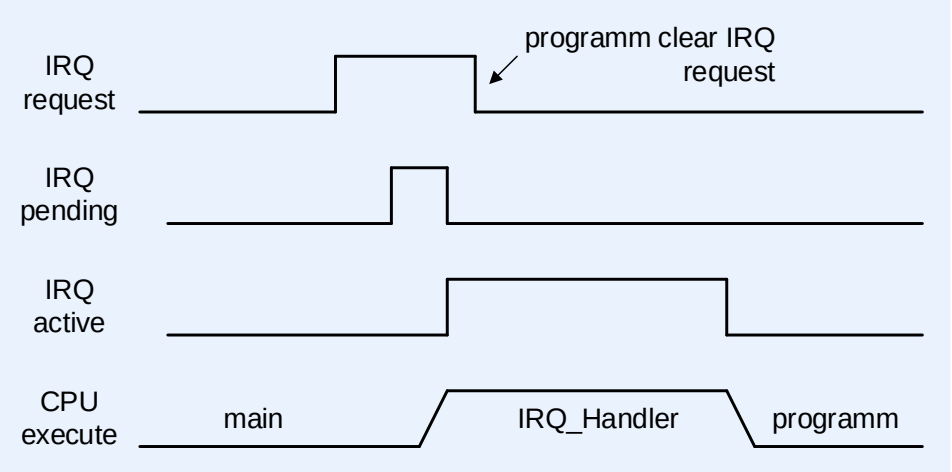



Рисунок 242 – Сброс признака отложенного прерывания, до обработки ядром

После того как прерывание стало активным, повторно запустить обработчик того же прерывания будет невозможно до тех пор, пока не будет завершена процедура обработки прерывания командой выхода из исключения. После выполнения команды выхода происходит сброс признака активности IRQ active.

При удержании источником на входе NVIC запроса на обработку IRQ request, по окончании обработки прерывания и снятия признака активного прерывания IRQ active, происходит повторное выставление признака отложенного прерывания IRQ pending – «защелкивание» pending бита, сброс которого в дальнейшем инициирует повторную активность и обработку того же исключения – рисунок 232.

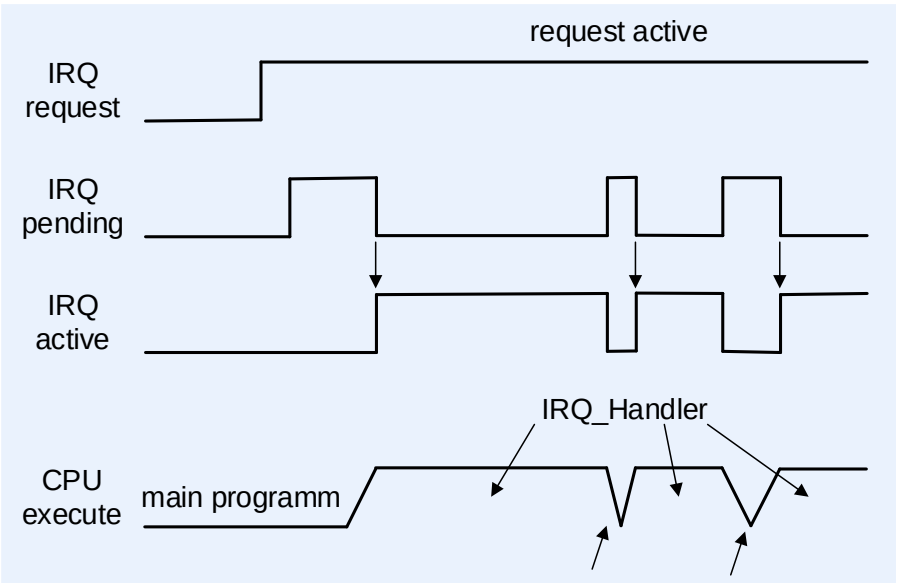



Рисунок 243 – Повторная обработка прерываний при удержании запроса от источника

Необходимо учитывать, что если источник прерываний выдает многократную установку и снятие запроса IRQ request на входе контроллера NVIC, то в таком случае только первый запрос выставляет признак отложенного прерывания IRQ pending, а остальные запросы до начала процедуры обработки прерывания (в момент активного признака отложенного прерывания) будут проигнорированы ядром – рисунок 233.

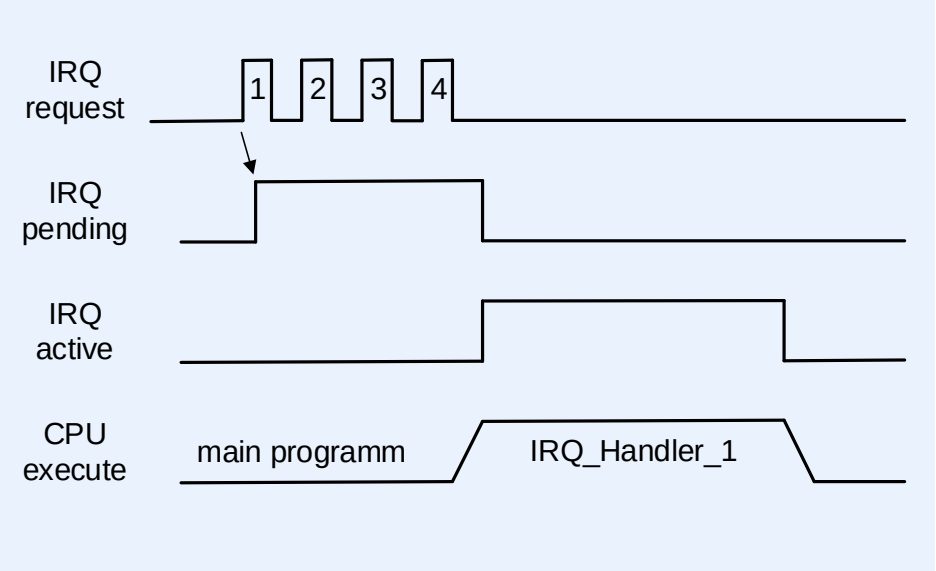



Рисунок 244 – Многократная установка снятие запроса IRQ request

Если запрос на прерывание пришел в момент активного прерывания, то в такой ситуации уже будут отработаны оба запроса на прерывание. В отличие от случая, изображенного на рисунке 244, запрос приходит тогда, когда признак отложенного прерывания IRQ pending уже сброшен, и новый запрос как раз его выставляет, что в дальнейшем позволяет провести повторную обработку прерывания – рисунок 234.

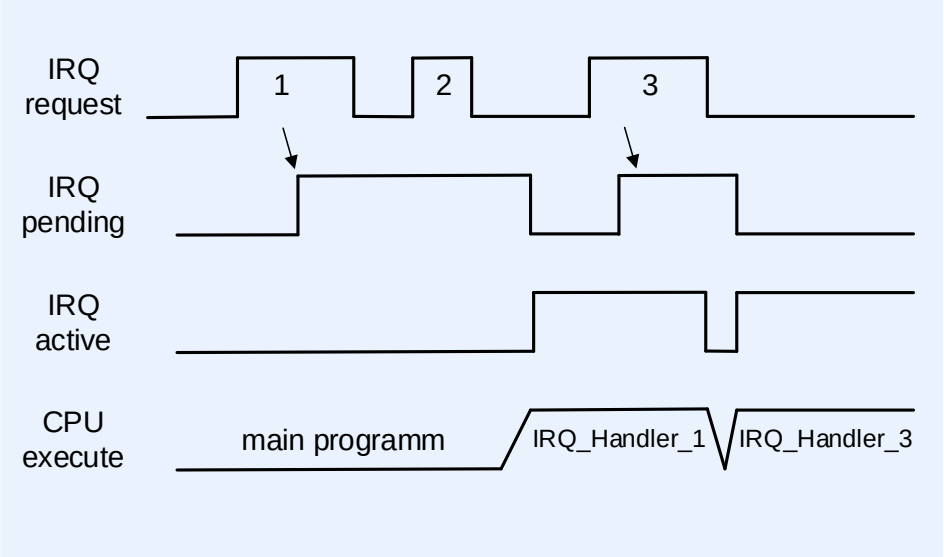



Рисунок 245 – Повторная установка запроса на прерывание в момент выполнения обработчика исключения

Выставление признака отложенного прерывания возможно даже в тех случаях, когда соответствующее прерывание запрещено. Все отложенные прерывания будут отражены в ISPR/ICPR, и в случае разрешения таких прерываний регистром ISPR, ядро тут же приступит к их обработке. Рекомендуется перед разрешением соответствующего прерывания убедиться в отсутствии признака отложенного запроса и, при необходимости, сбросить его.

### 23.2.2 *Регистр разрешения прерываний*

Регистр ISER предназначен для разрешения прерываний (запись) и определения, какие из прерываний разрешены (чтение). Более подробная информация представлена в таблице 690.

Распределение битов регистра представлено на рисунке 235.

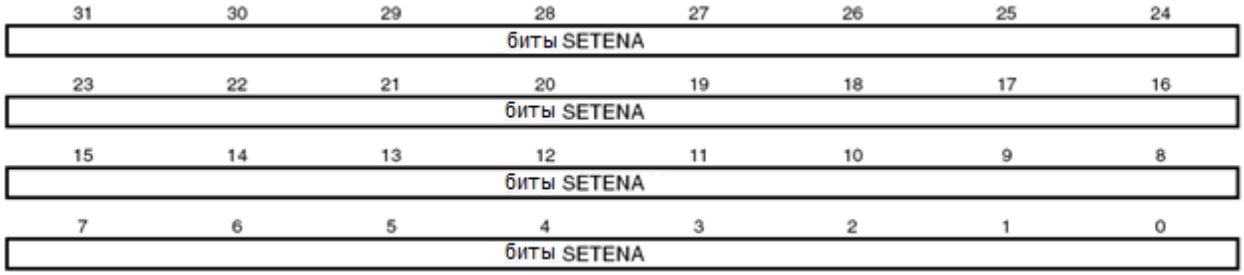



Рисунок 246 – Распределение битов регистра ISER

Таблица 697 – Регистр разрешения прерываний

| Биты | Поле | Функция |
|---|---|---|
| 31…0 | SETENA | Биты разрешения прерывания.<br>**При записи:**<br>1 – разрешение прерывания;<br>0 – не оказывает влияния.<br>**При чтении:**<br>1 – прерывание разрешено;<br>0 – прерывание запрещено. |

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.

### 23.2.3 *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 таблице 698 показано отображение прерываний (номеров запросов IRQ) на регистры прерываний и соответствующие переменные CMSIS, для которых предусмотрено по одному биту на прерывание.

Таблица 698 – 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

#### 23.2.3.1 NVIC->ISER[x]

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

Таблица 699 – 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.

#### 23.2.3.2 NVIC->ICER[x]

Регистр запрета прерываний

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

Таблица 700 – 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.

#### 23.2.3.3 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 701 – 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.

#### 23.2.3.4 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 702 – 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.

#### 23.2.3.5 NVIC->IABR[x]

Регистр активных прерываний

Регистр ICPR0 показывает, какие из прерываний находятся в активном состоянии. Этот регистр доступен только для чтения (Таблица 703).

Таблица 703 – 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.

### 23.2.3.6 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 704 – 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].

### 23.2.3.7 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 705 – 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.

### 23.2.4 *Прерывания, срабатывающие по уровню сигнала*

Процессор способен обрабатывать прерывания, сформированные по уровню сигнала. Формирование запроса на прерывание по уровню происходит при условии удержания сигнала не менее двух тактов процессорного ядра.

Прерывание такого типа считается активным до тех пор, пока периферийное устройство не снимет активный уровень сигнала запроса. Как правило, это происходит после соответствующего обращения процедуры обработки прерывания к периферийному устройству.

После того, как процессор передал управление на обработчик, он автоматически снимает признак ожидания обслуживания прерывания (см. 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.

### 23.2.5 *Аппаратное и программное управление прерываниями*

Процессор Cortex-M4 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.

### 23.2.6 *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 706 – 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.

# 24 Блок управления системой

Блок управления системой SCB обеспечивает доступ к информации о конфигурации и управление работой системы. Регистры блока управления системой представлены в таблице 707.

Таблица 707 – 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 |

## 24.1 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].

### 24.1.1 *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 708.

Таблица 708 – 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.

### 24.1.2 *SCB->CPUID*

Регистр идентификации процессора

Регистр CPUID содержит информацию о модели процессора, версии и варианте его реализации. Подробная информация о регистре представлена в таблице 709.

Таблица 709 – 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.

### 24.1.3 *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 710 – 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 | R/W |
| **Сброс** | 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.

### 24.1.4 *SCB->VTOR*

Регистр смещения таблицы векторов прерываний

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

Таблица 711 – Регистр смещения таблицы векторов прерываний

| **Номер** | 31, 30 | 29…7 | 6...0 |
|---|---|---|---|
| **Доступ** | U | R/W | R/W |
| **Сброс** | 0 | 0 | 0 |
| | **-** | **TBLOFF** | **Reserved** |

TBLOFF – смещение базового адреса таблицы векторов относительно нижней границы карты распределения памяти. Собственно, смещение хранится в битах [28:7]. Бит [29] определяет, размещена ли таблица в области кода или в области памяти SRAM: 0 = область кода, 1 = SRAM. Бит [29] может также обозначаться как TBLBASE.

При установке значения TBLOFF требуется обеспечить выравнивание базового адреса таблицы векторов. Минимальный размер выравнивания – по границе блока из 32 слов, достаточен для хранения 16 векторов прерываний. Для поддержки большего количества прерываний необходимо увеличить размер выравнивания до ближайшей степени двойки, большей или равной размеру таблицы. Например, для хранения 21 вектора прерываний таблицу следует выровнять по границе блока из 64 слов, так как ее объем составляет 37 слов, а ближайшая степень двойки, большая или равная 37, равна 64.

Учитывая описанные выше требования по выравниванию, разряды [6…0] смещения всегда равны нулю.

### 24.1.5 *SCB->AIRCR*

Регистр управления прерываниями и программного сброса

Регистр AIRCR позволяет группировать исключения по приоритетам, задавать порядок следования байтов в слове (endian) при доступе к данным, а также управлять процессом сброса системы.

Для записи данных в регистр необходимо установить его поле VECTKEY в значение 0x05FA, в противном случае попытка записи будет проигнорирована процессором.

Таблица 712 – 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 713.

Таблица 713 – Группировка приоритетов прерываний

| **PRIGROUP** | **Значение приоритета в поле PRI_N[7:0]** | | | **Общее количество** | |
|---|---|---|---|---|---|
| | **Положение двоичной точки** | **Биты номера группы** | **Биты номера подгруппы** | **Групп** | **подгрупп** |
| 0b000, 0b001, 0b010, 0b011, 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.

### 24.1.6 *SCB->SCR*

Регистр управления системой

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

Таблица 714 – 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.

### 24.1.7 *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 715 – 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 UDI:

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”).

### 24.1.8 *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 716 – 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 717 – 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 718 – 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 719 – 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

### 24.1.9 *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 720 – Регистр управления и состояния системных обработчиков

| **Номер** | 31...19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 |
|---|---|---|---|---|---|---|---|---|---|
| **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 |
| | **-** | **USGFAULTENA** | **BUSFAULTENA** | **MEMFAULTENA** | **SVCALLPENDED** | **BUSFAULTPENDED** | **MEMFAULTPENDED** | **USGFAULTPENDED** | **SYSTICKACT** |

| **Номер** | 10 | 9 | 8 | 7 | 6...4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| **Доступ** | 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 |
| | **PENDSVACT** | **-** | **MONITORACT** | **CVCALLAVCT** | **-** | **USGFAULTACT** | **-** | **BUSFAULTACT** | **MEMFAULTACT** |

USGFAULTENA разрешение обработки отказов, вызванных ошибками программирования, 1 – разрешено, 0 – запрещено.

BUSFAULTENA разрешение обработки отказа доступа к шине, 1 – разрешено, 0 – запрещено.

MEMFAULTENA разрешение обработки отказа доступа к памяти, 1 – разрешено, 0 – запрещено.

SVCALLPENDED признак ожидания обработки вызова SVC, возвращает 1, если вызов ожидает обработки.

BUSFAULTPENDED признак ожидания обработки отказа доступа к шине, возвращает 1, если отказ ожидает обработки.

MEMFAULTPENDED признак ожидания обработки отказа доступа к памяти, возвращает 1, если отказ ожидает обработки.

USGFAULTPENDED признак ожидания обработки отказа, вызванного ошибками программирования, возвращает 1, если отказ ожидает обработки.

SYSTICKACT признак активности обработчика исключения SysTick, возвращает 1, если обработчик активен.

PENDSVACT признак активности обработчика исключения PendSV, возвращает 1, если обработчик активен.

MONITORACT признак активности монитора отладчика, возвращает 1, если монитор отладчика активен.

SVCALLACT признак активности обработчика вызова SVC, возвращает 1, если обработчик активен.

USGFAULTACT признак активности обработчика отказа, вызванного ошибкой программирования, возвращает 1, если обработчик активен.

BUSFAULTACT признак активности обработчика отказа доступа к шине, возвращает 1, если обработчик активен.

MEMFAULTACT признак активности обработчика отказа доступа к памяти, возвращает 1, если обработчик активен.

*Примечания*

*1 Установка бита разрешения в 1 разрешает обработку исключения, установка в 0 – запрещает.*

*2 Чтение 1 из бита-признака активности свидетельствует об активности исключения, 0 – о его неактивности. Существует возможность записи значения в данный бит для принудительного перевода исключения в активное состояние, однако при этом следует предпринять меры предосторожности, описанные далее в разделе;*

*3 Чтение 1 из бита-признака ожидания свидетельствует о том, что исключение находится в состоянии ожидания обработки. Существует возможность принудительного перевода исключения в состояние ожидания путем записи 1 в данный бит.*

Если в системе возникло исключение (отказ), обработчик которого запрещен, процессор формирует запрос на обработку тяжелого отказа.

Существует возможность принудительного перевода того или иного системного исключения в состояние ожидания обработки или активное состояние путем записи в соответствующий разряд регистра SHCSR.

Например, ядро операционной системы может осуществлять запись в биты – признаки активности для того, чтобы осуществить переключение контекста со сменой типа обрабатываемого исключения.

Программа, меняющая значение бит – признаков активности исключения, должна обеспечить необходимую корректировку содержимого стека, в противном случае процессор может сгенерировать отказ. Необходимо убедиться, что программа сохраняет и впоследствии корректно восстанавливает текущее значение признаков активности исключений.

После разрешения системных обработчиков все дальнейшие манипуляции с битами регистра необходимо производить, последовательно выполняя операции чтения, модификации и обратной записи, гарантирующие изменение только необходимых разрядов регистра.

### 24.1.10 *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 721 – 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.

### 24.1.10.1 Поле MMFSR

Регистр состояния отказов доступа к памяти

Регистр MMFSR содержит набор флагов, указывающих на различные причины отказа доступа к памяти.

Таблица 722 – Регистр состояния отказов доступа к памяти

| **Номер** | 7 | 6, 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **Доступ** | RO | U | RO | RO | U | RO | RO |
| **Сброс** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | **MMARVALID** | **-** | **MSTKERR** | **MUNSTKERR** | **-** | **DACCVIOL** | **IACCVIOL** |

MMARVALID признак корректности значения в регистре адреса отказа доступа к памяти (MMAR): 0 = значение в MMAR не содержит корректный адрес отказа, 1 = содержит.

В случае если произошла эскалация отказа доступа к памяти, обработчик 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.

### 24.1.10.2 Поле BFSR

Регистр состояния отказов доступа к шине

Регистр BFSR содержит набор флагов, указывающих на различные причины отказа доступа к шине:

Таблица 723 – Регистр состояния отказов доступа к шине

| **Номер** | 7 | 6, 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **Доступ** | RO | U | RO | RO | RO | RO | RO |
| **Сброс** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | **BFRVALID** | | **STKERR** | **UNS TKERR** | **IMPRECI SERR** | **PRECI SERR** | **IBU SERR** |

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.

### 24.1.10.3 Поле UFSR

Регистр состояния отказов, вызванных ошибками программирования

Регистр UFSR содержит набор флагов, указывающих на различные причины отказа.

Таблица 724 – 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 |
| | **-** | **DIV BYZERO** | **UN ALIGNED** | **-** | **NOCP** | **INV PC** | **INV STATE** | **UNDEF INSTR** |

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.

## 24.1.11 *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 725 – 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.

### 24.1.12 *SCB->MMFAR*

Регистр адреса отказа доступа к памяти

Регистр MMFAR содержит адрес, при обращении по которому возникла ошибка управления памятью.

Таблица 726 – 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”.

### 24.1.13 *SCB->BFAR*

Регистр адреса отказа доступа к шине

Регистр BFAR содержит адрес, при обращении по которому возникла ошибка доступа к шине.

Таблица 727 – 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”.

### 24.1.14 *Рекомендации по программированию блока управления системой*

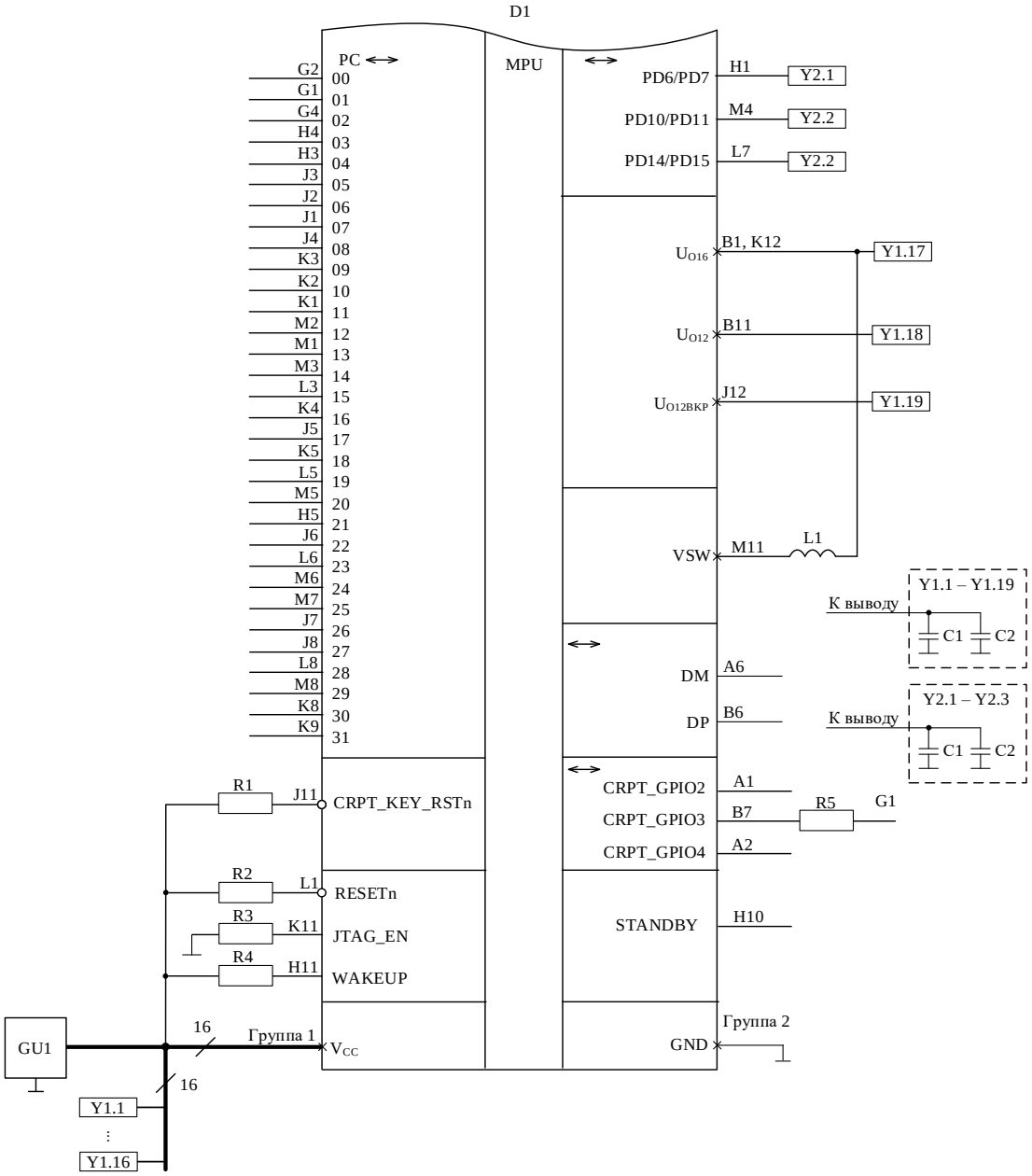
Необходимо убедиться, что программа использует для обращения к регистрам блока управления системой доступ по корректно выровненным адресам. Обращение ко всем регистрам, за исключением CFSR и SHPR1-SHPR3, должно быть выровнено по границе слова. Регистры CFSR и SHPR1-SHPR3 допускают как побайтный доступ, 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.

# 25 Типовая схема включения




D1 – контролируемая микросхема;
GU1 – источник напряжения питания, от 3,0 до 3,6 В;
L1 – индуктивность 3,3 мкГн ± 10 %;
R1 – R5 – резисторы сопротивлением 10 кОм ± 10 %;
Y1, Y2 – элементы схемы.

Элементы схемы Y1.1 – Y1.19.
C1 – конденсатор емкостью 10 мкФ ± 10 %, напряжение 10 В;
C2 – конденсатор емкостью 0,1 мкФ ± 10 %, напряжение 10 В.

Элементы схемы Y2.1 – Y2.3.
C1 – конденсатор емкостью 4,7 мкФ ± 10 %, напряжение 10 В;
C2 – конденсатор емкостью 0,1 мкФ ± 10 %, напряжение 10 В.

Группы выводов:
– Группа 1: A11, B12, C6, C7, F3, F10, G3, H2, H9, J10, K6, K7, L2, L4, L12, M12;
– Группа 2: E6, E7, F5 - F8, G5 - G8, H6, H7, L11, M10.

Рисунок 247 – Типовая схема включения

# 26 Электрические параметры микросхем

Таблица 728 – Электрические параметры микросхем при приемке и поставке

| Наименование параметра, единица измерения, режим измерения | Буквенное обозначение параметра | Норма параметра | | Температура среды, °С |
|---|---|---|---|---|
| | | не менее | не более | |
| Выходное напряжение низкого уровня, В, на выходах портов PA, PB, PC, PD6/PD7, PD10/PD11, PD14/PD15, CPRT_GPIO[7:5], STANDBY при работе в цифровом режиме | $U_{OL}$ | 0[1)] | 0,4 | 25, 85, – 40 |
| Выходное напряжение высокого уровня, В, на выходах портов PA, PB, PC, PD6/PD7, PD10/PD11, PD14/PD15, CPRT_GPIO[7:5], STANDBY при работе в цифровом режиме | $U_{OH}$ | 2,4 | 3,6[1)] | |
| Ток утечки низкого уровня, мкА, на выходах портов PA, PB, PC, PD6/PD7, PD10/PD11, PD14/PD15, STANDBY, CPRT_GPIO[7:5], CRPT_KEY_RSTn, RESETn, WAKEUP, JTAG_EN при работе в цифровом режиме | $I_{ILL}$ | – 10 | 10 | |
| Ток утечки высокого уровня, мкА, на выходах портов PA, PB, PC, PD6/PD7, PD10/PD11, PD14/PD15, CPRT_GPIO[7:5], CRPT KEY_RSTn, RESETn, STANDBY при работе в цифровом режиме | $I_{ILH}$ | – 10 | 10 | |
| Входной ток высокого уровня, мкА, на входах WAKEUP, JTAG_EN с резистором доопределения до нуля | $I_{IH}$ | 100 | 500 | |
| Динамический ток потребления, А, при $f_{CLK}$ = 160 МГц, разрешено тактирование всех периферийных блоков | $I_{CCO}$ | 0 | 1,0 | |
| Статический ток потребления в режиме пониженного энергопотребления, мА | $I_{CCS}$ | 0 | 10 | |
| **АЦП** | | | | |
| Дифференциальная нелинейность АЦП, ЕМР, (однополярный режим) | $E_{LD_ADC}$ | – 1 | 3 | 25, 85, – 40 |
| Интегральная нелинейность АЦП, ЕМР, (однополярный режим) | $E_{L_ADC}$ | – 5 | 5 | |
| Ошибка смещения нуля АЦП, % ПШ, (однополярный режим) | $E_{IO_ADC}$ | – 1 | 1 | |
| Погрешность полной шкалы АЦП, % ПШ, (однополярный режим) | $E_{G_ADC}$ | – 1 | 1 | |
| **ЦАП** | | | | |
| Минимальное выходное напряжение ЦАП, В, при DAC_DATA = 0x000 | $U_{O_DAC_min}$ | – | 0,08 | 25, 85, – 40 |
| Максимальное выходное напряжение ЦАП, В, при DAC_DATA = 0xFFF | $U_{O_DAC_max}$ | $U_{REFp_DAC}$ – 0,08 | – | |
| Дифференциальная нелинейность ЦАП, ЕМР | $E_{LD_DAC}$ | – 1 | 3 | |
| Интегральная нелинейность ЦАП, ЕМР | $E_{L_DAC}$ | – 5 | 5 | |
| Ошибка смещения нуля ЦАП, % ПШ | $E_{IO_DAC}$ | – 1 | 1 | |
| Погрешность полной шкалы ЦАП, % ПШ | $E_{G_DAC}$ | – 1 | 1 | |

| Наименование параметра, единица измерения, режим измерения | Буквенное обозначение параметра | Норма параметра | | Температура среды, °С |
|---|---|---|---|---|
| | | не менее | не более | |
| **Генератор частоты и PLL** | | | | |
| Выходная частота LSI RC-генератора, кГц, при LSI_TRIM = 0x40 | $f_{O_LSI}$ | 20 | 60 | 25, 85, – 40 |
| Выходная частота HSI RC-генератора, МГц при HSI_TRIM = 0x40 | $f_{O_HSI}$ | 5 | 11 | |
| Выходная частота PLL, МГц,<br>– минимальная; | $f_{O_PLL}$ | – | 40 | |
| – максимальная | | 160 | – | |
| **USB** | | | | |
| Выходное напряжение низкого уровня, В, в режиме Full Speed | $U_{OL_USB}$ | – | 0,3 | 25, 85, – 40 |
| Выходное напряжение высокого уровня, В, в режиме Full Speed | $U_{OH_USB}$ | 2,8 | – | |

[1)] Норма параметра обеспечивается конструкцией микросхемы

Микросхемы должны быть устойчивы к воздействию статического электричества с потенциалом не менее 2000 В*.

* Уточняется в ходе ОКР.

# 27 Предельно-допустимые и предельные параметры

Таблица 729 – Предельно-допустимые электрические режимы эксплуатации и предельные электрические режимы микросхем

| Наименование параметра, единица измерения | Буквенное обозначение параметра | Предельно-допустимый режим | | Предельный режим | |
|---|---|---|---|---|---|
| | | не менее | не более | не менее | не более |
| Напряжение питания, В | $U_{CC}$ | 3,0 | 3,6 | – | 4,0 |
| **Схемы ввода-вывода** | | | | | |
| Входное напряжение низкого уровня, В, на входах портов PA, PB, PC, PD6/PD7, PD10/PD11, PD14/PD15, CPRT_GPIO[7:5], RESETn, WAKEUP, CRPT_KEY_RSTn, JTAG_EN при работе в цифровом режиме | $U_{IL}$ | 0 | 0,8 | – 0,3 | – |
| Входное напряжение высокого уровня, В, на входах портов PA, PB, PC, PD6/PD7, PD10/PD11, PD14/PD15, CPRT_GPIO[7:5], RESETn, WAKEUP, CRPT_KEY_RSTn, JTAG_EN при работе в цифровом режиме | $U_{IH}$ | 2,0 | $U_{CC}$ | – | $U_{CC}$ + 0,3 |
| Выходной ток низкого уровня, мА, на выходах портов PA, PB, PC, PD6/PD7, PD10/PD11, PD14/PD15, CPRT_GPIO[7:5], STANDBY при работе в цифровом режиме | $I_{OL}$ | – | 4,0[1)] | – | 8,0[1)] |
| Выходной ток высокого уровня, мА, на выходах портов PA, PB, PC, PD6/PD7, PD10/PD11, PD14/PD15, CPRT_GPIO[7:5], STANDBY при работе в цифровом режиме | $I_{OH}$ | – 4,0[2)] | – | – 8,0[2)] | – |
| Емкость нагрузки, пФ, на выходах портов PA, PB, PC, PD6/PD7, PD10/PD11, PD14/PD15, CPRT_GPIO[7:5], STANDBY при работе в цифровом режиме | $C_{L_IO_max}$ | – | 50 | – | 50 |
| **Генератор частоты и PLL** | | | | | |
| Частота следования импульсов тактовых сигналов, МГц:<br>– основных ядер; | $f_{CLK}$ | 0,03 | 160 | – | – |
| – защищенного ядра | | 0,03 | 130 | – | – |
| Частота следования импульсов тактовых сигналов, МГц, на выводах PA27 – PA30, сконфигурированных в аналоговый режим, при:<br>– HSEx_BYP = «0»; | $f_{C_HSE}$ | 2 | 30 | – | – |
| – HSEx_BYP = «1» | | 1 | 160 | – | – |
| Частота следования импульсов тактовых сигналов, кГц, на выводах PA5, PA6, сконфигурированных в аналоговый режим, при:<br>– LSE_BYP = «0»; | $f_{C_LSE}$ | 32 | 33 | – | – |
| – LSE_BYP = «1» | | 0 | 1000 | – | – |
| Входная частота PLL, МГц | $f_{C_PLL}$ | 1 | 40 | – | – |
| Частота фазового детектора PLL, МГц | $f_{PFD_PLL}$* | 1 | 40 | – | – |
| Внутренняя частота PLL, МГц | $f_{INT_PLL}$* | 150 | 300 | – | – |

| Наименование параметра, единица измерения | Буквенное обозначение параметра | Предельно-допустимый режим | | Предельный режим | |
|---|---|---|---|---|---|
| | | не менее | не более | не менее | не более |
| **АЦП** | | | | | |
| Входное напряжение верхней границы опоры АЦП, В | $U_{REFp_ADC}$ | 2,40 | 2,55 | – | $U_{CC}$ + 0,3 |
| Входное напряжение нижней границы опоры АЦП, В | $U_{REFn_ADC}$ | 0 | 0,05 | – 0,3 | – |
| Диапазон входных напряжений, В | $U_I$ | $U_{REFn_ADC}$ | $U_{REFp_ADC}$ | – 0,3 | $U_{CC}$ + 0,3 |
| Частота тактирования АЦП, МГц | $f_{C_ADC}$ | 1 | 64 | – | – |
| **ЦАП** | | | | | |
| Входное напряжение верхней границы опоры ЦАП, В | $U_{REFp_DAC}$ | 2,4 | $U_{CC}$ | – 0,3 | $U_{CC}$ + 0,3 |
| Входное напряжение нижней границы опоры ЦАП, В | $U_{REFn_DAC}$ | 0 | 0,3 | – 0,3 | $U_{CC}$ + 0,3 |
| Резистивная нагрузка ЦАП, кОм | $R_{L_DAC}$ | 10 | – | – | – |
| Емкостная нагрузка ЦАП, нФ | $C_{L_DAC}$ | – | 1 | – | – |
| **Компаратор** | | | | | |
| Напряжение на входах компаратора, В | $U_{I_COMP}$ | 0 | $U_{CC}$ | – 0,3 | $U_{CC}$ + 0,3 |
| **USB** | | | | | |
| Входное напряжение низкого уровня, В, на выводах DN, DP в режиме Full Speed | $U_{IL_USB}$ | – | 0,8 | – | – |
| Входное напряжение высокого уровня, В, на выводах DN, DP в режимах Full Speed | $U_{IH_USB}$ | 2,0 | – | – | – |

* См. раздел «Блоки умножения тактовой частоты PLLn».

Примечание – Не допускается одновременное задание нескольких предельных режимов.

---

[1] Суммарный выходной ток должен быть не более 100 мА в предельно-допустимом режиме и не более 150 мА в предельном режиме.

[2] Суммарный выходной ток должен быть не менее минус 100 мА в предельно-допустимом режиме и не менее минус 150 мА в предельном режиме

# 28 Справочные данные

Предельная температура p-n перехода кристалла 150 °С.

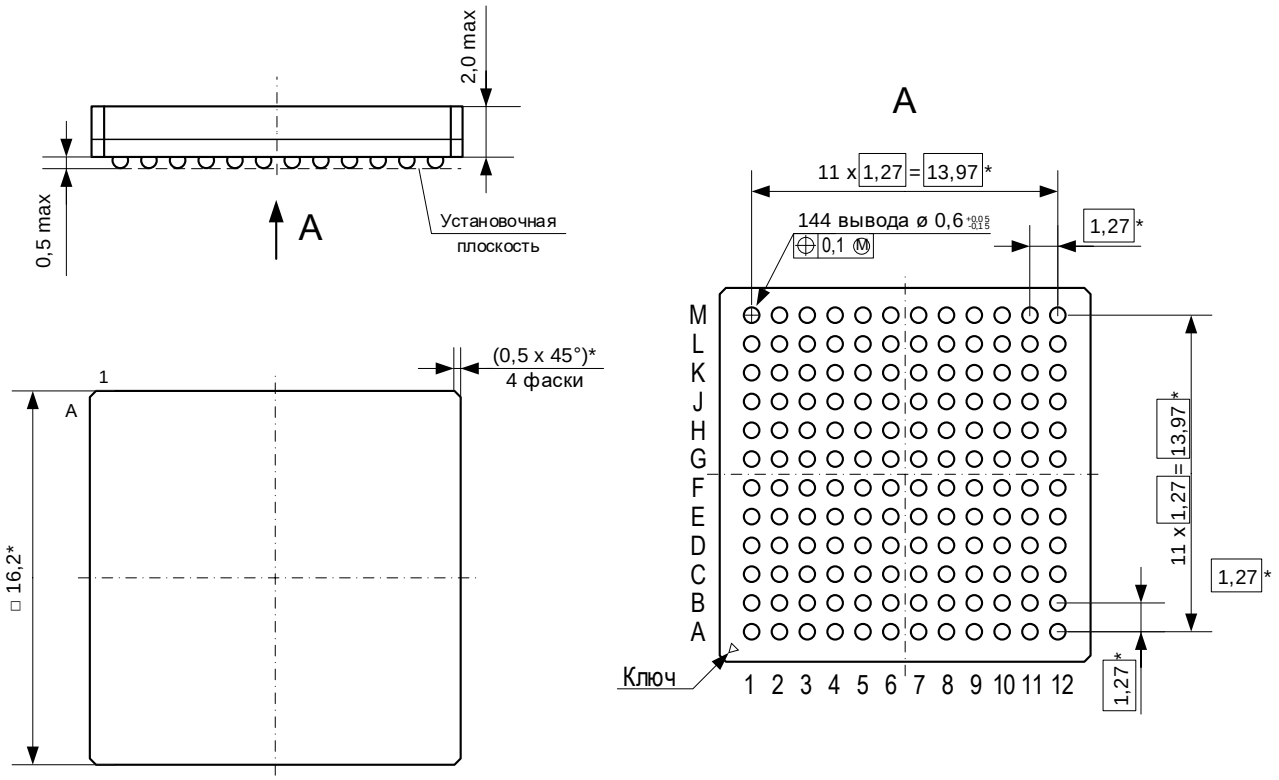
Таблица 14 – Справочные параметры микросхем

| Наименование параметра, единица измерения, режим измерения | Буквенное обозначение параметра | Норма параметра | | | Температура среды, °С |
|---|---|---|---|---|---|
| | | не менее | типовая | не более | |
| Статический ток потребления в режиме STANDBY, мА, регулятор напряжения выключен, BKP работает | $I_{CCS_BKP}$ | - | 1,7 | 10 | 25 |
| Динамический ток потребления, мА, при $f_{CLK}$ = 8 МГц, включено одно вычислительное ядро в режиме пустого цикла while(1){}; | $I_{CCO_D1}$ | - | 15 | 50 | 25 |
| Динамический ток потребления, мА, при $f_{CLK}$ = 160 МГц, включены оба вычислительных ядра в режиме пустого цикла while(1){}; | $I_{CCO_D2}$ | - | 80 | 100 | 25 |
| Динамический ток потребления, мА, при $f_{CLK}$ = 160 МГц, включены оба вычислительных ядра, разрешено тактирование всех периферийных блоков. Включен блок криптографического сопроцессора | $I_{CCO_D3}$ | - | 330 | 500 | 25 |
| Статический ток потребления IO, мкА | $I_{CCS_IO}$ | - | - | ** | 25 |
| Динамический ток потребления IO, мА | $I_{CCO_IO}$ | - | - | * | 25 |
| Статический ток потребления BKP, мкА | $I_{CCS_BKP}$ | - | - | * | 25 |
| Динамический ток потребления BKP, мА | $I_{CCO_BKP}$ | - | - | * | 25 |
| Статический ток потребления ANA, мкА | $I_{CCS_ANA}$ | - | - | * | 25 |
| Динамический ток потребления ANA, мА | $I_{CCO_ANA}$ | - | - | * | 25 |
| Статический ток потребления DAC, мкА | $I_{CC_DAC}$ | - | - | * | 25 |
| Динамический ток потребления DAC, мА | $I_{CCO_DAC}$ | - | - | * | 25 |
| Статический ток потребления ADC, мкА | $I_{CCS_ADC}$ | - | - | * | 25 |
| Динамический ток потребления ADC, мА | $I_{CCO_ADC}$ | - | - | * | 25 |
| Статический ток потребления BG, мкА | $I_{CCS_BG}$ | - | - | * | 25 |
| Динамический ток потребления BG, мА | $I_{CCO_BG}$ | - | - | * | 25 |
| **USB** | | | | | |
| Динамический ток потребления, мА | $I_{CCO_USB}$ | | | | 25 |
| – в режиме Full speed USB; | | 3,8 | - | 4,2 | |
| – в режиме Suspend USB | | 0,1 | - | 3 | |
| Статический ток потребления USB, мкА | $I_{CCS_USB}$ | - | - | * | 25 |
| **АЦП** | | | | | |
| Коэффициент нелинейных искажений, дБ, режим внутреннего источника опорного напряжения 2,5 В, частота семплирования 1 МВыб/с, входной синусоидальный сигнал с частотой 10,1 кГц, работа от внутреннего регулятора HLDO: | THD | | | | 25 |
| – однополярный режим работы; | | - | 75,9 | - | |
| – дифференциальный режим работы | | - | 76,8 | - | |

* Уточняется в ходе ОКР.

| Наименование параметра, единица измерения, режим измерения | Буквенное обозначение параметра | Норма параметра | | | Температура среды, °С |
|---|---|---|---|---|---|
| | | не менее | типовая | не более | |
| Соотношение сигнал/шум, дБ, режим внутреннего источника опорного напряжения 2,5 В, частота семплирования 1 МВыб/с, входной синусоидальный сигнал с частотой 10,1 кГц, работа от внутреннего регулятора HLDO: | SNDR | | | | 25 |
| – однополярный режим работы; | | - | 63,2 | - | |
| – дифференциальный режим работы | | - | 66,3 | - | |
| **ЦАП** | | | | | |
| Время включения ЦАП, мкс | $t_{ON_DAC}$ | - | - | 12 | 25 |
| Время установления ЦАП из $U_{DACMAX}$ в $U_{DACMIN}$, мкс | $t_{SHL_DAC}$ | - | - | 10 | 25 |
| Время установления ЦАП из $U_{DACMAX}$ в $U_{DACMAX}/2$, мкс | $t_{SMH_DAC}$ | - | - | 15 | 25 |
| Время установления ЦАП из $U_{DACMIN}$ в $U_{DACMAX}/2$, мкс | $t_{SML_DAC}$ | - | - | 15 | 25 |
| **Компаратор** | | | | | |
| Напряжение гистерезиса компаратора, мВ | $U_{HYST_CMP}$ | 10 | - | 30 | 25 |
| Напряжение смещения нуля компаратора, мВ | $U_{OFS_CMP}$ | 5 | - | 6 | 25 |
| Время задержки мультиплексоров выбора источников сравнения, нс (с момента изменения selp или seln до установки сигнала на входах компаратора) | $t_{C_MPMX}$ | - | - | 500 | 25 |
| Время включения компаратора, мкс | $t_{ON}$ | - | - | 100 | 25 |
| Время задержки переключения компаратора, нс | $t_{dtran}$ | - | - | 500 | 25 |
| **ШИМ высокого разрешения (HRPWM)** | | | | | |
| Длительность спада выходных сигналов, нс | $t_{F_PWM}$ | - | - | 3 | 25 |
| Длительность нарастания выходных сигналов, нс | $t_{R_PWM}$ | - | - | 3 | 25 |
| Время установления внутренней DLL, мкс, (от сигнала en до появления флага rdy) | $t_{SU_PWM}$ | - | - | 150 | 25 |
| **Система питания** | | | | | |
| Уровень напряжения питания для выключения сброса по снижению питания, В | $U_{POROFF}$ | 2,6 | - | 2,8 | 25 |
| Уровень напряжения питания для включения сброса по повышению питания, В | $U_{PORON}$ | 2,7 | - | 2,9 | 25 |
| Уровень напряжения питания для срабатывания схемы защиты от превышения допустимого уровня питания, В | $U_{OVROFF}$ | 3,6 | - | 3,8 | 25 |
| Уровень напряжения питания для выключения схемы защиты от превышения допустимого уровня питания, В | $U_{OVRON}$ | 3,6 | - | 3,8 | 25 |
| Выходное напряжение регулятора HLDO, В | $U_{OH_LDO}$ | 1,35 | - | 1,70 | 25 |
| Выходное напряжение регулятора DCDC, В | $U_{O_DCDC}$ | 1,35 | - | 1,70 | 25 |
| Выходное напряжение регулятора LLDO, В | $U_{O_LLDO}$ | 0,95 | - | 1,30 | 25 |
| Выходное напряжение регулятора BLDO, В | $U_{O_BLDO}$ | 1,00 | - | 1,30 | 25 |
| **Flash-память** | | | | | |
| Время хранения информации, лет | $t_{GS}$ | 10 | - | - | 85 |
| Число циклов записи/стирания данных | $N_{PR}$ | 10000 | - | - | 85 |

# 29 Габаритный чертеж микросхемы




* Размеры для справок.

Рисунок 248 – Микросхема в корпусе BGA144

# 30 Информация для заказа

| Обозначение микросхемы | Маркировка | Тип корпуса | Температурный диапазон, °С | Международное обозначение |
|---|---|---|---|---|
| К1986ВК01GI | MDR1201GI | BGA144 | от – 40 до 85 | MDR1201GI |

# Лист регистрации изменений

| № п/п | Дата | Версия | Краткое содержание изменения | №№ изменяемых листов |
|---|---|---|---|---|
| 1 | 05.08.2021 | 0.1.0 | Введена впервые | |
| 2 | 20.08.2021 | 0.2.0 | Обновлен раздел 20 «Программная модель защищенной подсистемы…» | 707-738 |
| 3 | 09.09.2021 | 0.3.0 | Дополнено описание контроллера USART (ISO7816).<br>Исправлена нумерация подразделов в разделе 20 | 296-329<br>769-784 |
| 4 | 21.02.2022 | 0.4.0 | Исправление опечаток<br>Основные характеристики. Периферийные блоки 2xCortex- M4F – дополнена информация об аналоговых АЦП<br>Подразделы «Использование DMA», «Запрос сеанса OTG», «Динамическое измерение размера FIFO» исключены<br>Разделы «USB контроллер канального уровня» и «Контроллер SDIO» перенесены в раздел «Блоки контроллеров интерфейсов передачи данных»<br>Добавлены описание блока и регистров CRC<br>Подраздел 3.6 обновлен<br>Раздел 6 – REG_60_TMRx заменено на REG_60_SYSx<br>Таблица 3 – добавлено примечание<br>Рисунки 20, 21 обновлены<br>Таблица 20 – адреса скорректированы<br>Подраздел 10.3 – алгоритмы для сторожевого таймера скорректированы<br>Подраздел 12.1.8 обновлен<br>Подраздел 12.1.9 – напряжение 1,25 В используется для тестовых целей<br>«Контроллер Ethernet (EthernetMAC)» – данные о режиме отладки удалены<br>Рисунок 119 – «Буфер 8кБ» -> «Буфер 32кБ»<br>Подраздел 13.5.9 – дополнена информация о режиме работы ОУ-ОУ<br>Подраздел 13.6 дополнен<br>Раздел 15 – описание обновлено, информация о DMA исключена<br>Рисунок 174 – Flash ключей –> RAM ключей<br>Рисунок 183, таблица 54 – добавлены названия<br>Подраздел 16.4 – добавлена информация о режиме standby<br>Подраздел 16.12 – скорректирована информация о статистическом тестировании<br>Рисунок 184 – добавлено название<br>Подраздел 16.13 – исключена информация о возможности использования 1 стоп-бита<br>Рисунок 189 – изменено состояние флага ТС после 3 фрейма<br>Подраздел 16.13.2.4 – удалены таблицы определения погрешностей. Добавлена таблица определения погрешностей (при fPCLK = 130 МГц)<br>Рисунок 203 – изменено условие передачи Data 3<br>Подраздел 16.14 скорректирован<br>Таблица 63 – адрес 0x4008_4000 зарезервирован<br>Подраздел 17.8 – описание регистра MIL_CLK_CTRL изменено<br>Таблица 69 – добавлено имя бита 20<br>Таблица 77 – скорректированы названия регистров. А также смещение регистров BLDO_CTRL(1, 2) | По тексту<br>3<br><br><br><br><br><br><br><br>29<br>38<br>40<br>67, 68<br>69<br>106<br><br>155<br>155<br><br>197<br><br>197<br>226<br><br>226<br>277<br><br>280<br>297, 299<br>282<br><br>299<br><br>300<br>300<br><br>306<br><br>311<br><br><br><br>323<br>324<br>330<br>342<br><br>367<br>420 |

| № п/п | Дата | Версия | Краткое содержание изменения | №№ изменяемых листов |
|---|---|---|---|---|
| | | | Подразделы 17.13.2, 17.13.4-17.13.8, 17.13.13 – таблицы дополнены | 422-429, 432-433 |
| | | | Подраздел 17.17.8 – описание битов [31:30] изменено | 508 |
| | | | Подраздел 17.18.2 – описание битов TX_READY, RX_READY скорректировано | 517 |
| | | | Таблицы 144, 145 – добавлено примечание | 539, 540 |
| | | | Таблица 167 – изменено описание бита EN_INT | 554 |
| | | | Подраздел 17.22.1 – описание битов DIV[6:0] дополнено | 558 |
| | | | Подраздел 17.26 – описание регистра ADC_BG_CTRL дополнено | 579 |
| | | | Подраздел 17.27.8 обновлен | 626 |
| | | | Подраздел 17.32 – описание регистров IntrTx, TxMaxP, TxCSR, RxMapP, RxCSR, RxCount, FIFOSize, FIFOx изменено. TxFIFOsz, RxFIFOsz, TxFIFOadd, RxFIFOadd зарезервированы | 681 |
| | | | Подраздел 17.32.1.2 – бит D6 зарезервирован | 684 |
| | | | Подраздел 17.32.2.4 – бит D1 зарезервирован | 694 |
| | | | Подраздел 18.16 – биты 8, 9 зарезервированы | 776 |
| | | | Добавлены разделы «Прерывания и исключения» и «Контроллеры прерываний NVIC» | 785, 802 |
| | | | Разделы 22, 23 приведены в соответствие с ТУ | 831, 833 |
| | | | Добавлен раздел «Справочные данные»» | 835 |
| 5 | 14.03.2022 | 0.4.1 | Исправление на габаритном чертеже | 837 |
| 6 | 01.04.2022 | 0.5.0 | Исправление опечаток | По тексту |
| | | | Рисунок 1 обновлен | 19 |
| | | | Добавлен разделы «Условно-графическое изображение», «Указания по применению и эксплуатации» | 20, 30 |
| | | | Таблица 1 обновлена, примечание исправлено | 22 |
| | | | Таблица «Назначение выводов по блокам» добавлена | 26 |
| | | | Таблица 18 – добавлены входы CS[7:0], CLOCK, OCLK | 72 |
| | | | Подраздел 9.6 – добавлены диаграммы записи на внешней шине | 73 |
| | | | Подраздел 13.2.1.8, дополнен | 147 |
| | | | Подразделы 14.3.6, 14.3.7 скорректированы | 206, 207 |
| | | | Рисунок 178 обновлен | 285 |
| | | | Таблица 123, 125 – порядок выводов BWEn, CS, CSn исправлены | 469, 471 |
| | | | Таблицы 127, 128 – примечания исключены | 474, 475 |
| | | | Подраздел 19.17.8 – описание бита DBGRF_EN скорректировано | 516 |
| | | | Рисунок 232 – ГЧ обновлен | 847 |
| 7 | 05.08.2022 | 0.6.0 | Исправление опечаток | По тексту |
| | | | Названия разделов 10, 11, 16, 17 скорректированы | |
| | | | Названия подразделов 9.1-9.10, 12.1-12.3, 13.1-13.4, 14.1-14.4, 15.1-15.9, 18.3, 18.5, 18.7.1-18.7.5, 18.8-18.13, 19.6-19.36 скорректированы | |
| | | | Таблицы подраздела 19.16.1 «Программная модель контроллера портов ввода-вывода (PORTx)» перенесены в раздел 3 «Описание выводов» | |
| | | | Структурная схема и УГО скорректированы | |
| | | | Обозначения выводов типа PWR скорректированы | По тексту |
| | | | Подраздел 5.3 – добавлено примечание | |
| | | | Подразделы 5.4, 7.4 – скорректирована информация о задержке $T_{PORSTn}$ | |
| | | | Подраздел 7.1.5 скорректирован | |
| | | | Подразделы 8.9, 8.11 дополнены | 60, 61, 63 |
| | | | Таблица 23 скорректирована | 67 |

| № п/п | Дата | Версия | Краткое содержание изменения | №№ изменяемых листов |
|---|---|---|---|---|
| | | | Подраздел 11.2 – дополнена информация об отключении режима LOCKSTEP | 119 |
| | | | Подраздел 13.1.2.2 дополнен | 133 |
| | | | Рисунок 57 скорректирован | 135 |
| | | | Подраздел 14.1 «Контроллер АЦП» дополнен | 166 |
| | | | Подраздел 14.1.9 скорректирован | |
| | | | Подраздел 14.1.10 «Входные сопротивление и емкость» добавлен | 174 |
| | | | Подраздел 14.3 дополнен | 176 |
| | | | Подраздел 14.4 скорректирован | |
| | | | Подраздел 15.1.4.7 «SSPTNBSYINTR» дополнен | 198 |
| | | | Подраздел 15.2.13 «Прерывания» – UARTTNBSYINTR: добавлено условие активности прерывания | 214 |
| | | | Подраздел 15.5.7 скорректирован | 242 |
| | | | Подраздел 19.12.2 – описание битов 3…0 скорректировано | 435 |
| | | | Подраздел 19.13.5 – бит 12 скорректирован | 445 |
| | | | Таблица 126 – описание битов скорректировано | 472 |
| | | | Таблица 159-161 – биты 13: «…по наличию данных…» → «…по отсутствию данных…» | 557-559 |
| | | | Подраздел 19.22.5 – описание бита INT скорректировано | 563 |
| | | | Подраздел 19.22.7 –описание бита CLR_INT скорректировано | 564 |
| | | | Подраздел 19.25.1 – скорректированы описания битов CMP_SELP, CMP_SELN | 580 |
| | | | Добавлена типовая схема включения | 847 |
| 8 | 14.01.2026 | 0.7.0 | Исправлен диапазон частот HSE0, HSE1 в режиме резонатора; | 2 |
| | | | Рисунки 1, 4, 8, 10 – заменены; | 20,42,46,50 |
| | | | Рисунки19 – 22 скорректированы; | 81–83 |
| | | | Пункты 7.1.1, 7.1.2, 7.1.4, 15.1.4, 15.4.1, 15.6.7 – 15.6.9 скорректированы; | 49,50,226,250, 291-293 |
| | | | Подраздел 7.2 скорректирован; | 51 |
| | | | Подраздел 7.3 – скорректирован рисунок 12 | 53 |
| | | | Подразделы 7.4, 12.2, 13.1, 13.3, 13.4, 14.2, 15.7, 18.9, 18.11 обновлены; | 53,127,131–163, 179–193,201,294 –297,365,366,368 |
| | | | Добавлен подраздел 8.16; | 72 |
| | | | Подраздел 9.3 – исправлен размер информационной памяти с 12 на 48 Кбайт; | 78 |
| | | | Подпункт 9.3.1.2, подраздел 9.7 скорректирован; | 80,90 |
| | | | Подразделы 12.2, 12.4 дополнены; | 127,129 |
| | | | Рисунки 45, 90, 111, 116 – 126, 128 – 130 скорректированы; | 127,177,198,209–220,222–224 |
| | | | Пункты 15.1.2, 15.4.8, 15.6.9, 15.6.10, 22.1.2 дополнены; | 210,258,293,294, 839 |
| | | | Подпункт 15.1.4.4 скорректирован; | 227 |
| | | | Добавлен подраздел 15.5 Контроллер CAN FD; | 264–284 |
| | | | Подраздел 18.8 – добавлены особенности работы MUL; | 359 |
| | | | Добавлен раздел 19 Системный таймер SysTick; | 395–397 |
| | | | Таблица 158 – скорректирована; | 402 |
| | | | Пункты 20.8.24 – 20.8.26 – скорректированы; | 436–439 |
| | | | Подраздел 20.9 – зарезервированы 0x0000_002C, 0x0000_0040, 0x0000_0060, 0x0000_0074, скорректировано описание FIRSTEVENT, LASTEVENT; | 444 |
| | | | Пункты 20.9.2, 20.9.3, 20.9.6, 20.9.7, 20.9.10 – 20.9.13, 20.9.15, 20.9.16, 20.9.18, 20.9.19, | 446,447,449,454–458,463,466,470, 471,496,497,503– |

| № п/п | Дата | Версия | Краткое содержание изменения | №№ изменяемых листов |
|---|---|---|---|---|
| | | | 20.13.1, 20.13.3, 20.13.8 – 20.13.13, 20.18.14, 20.31.3 – 20.31.8, 20.31.10 – 20.31.12 скорректированы;<br>Таблица 288– скорректирована;<br>Подраздел 20.20 – скорректирована таблица 343;<br>Пункт 20.20.3 – заменено «16 бит» на «32 бит»;<br>Подразделы 20.22, 20.26, 20.30, 20.31, 21.4 обновлены;<br><br><br>Таблицы 416 – 419, 428 – 431 скорректированы;<br>Раздел 28 добавлены параметры для flash-памяти | 506,569,713–717,719–721<br>558<br>584<br>586<br>605–608,625–630,694–721,786–788<br>636,640,641<br>894 |