

M0214

Руководство по программированию

STM32F3 и руководство по программированию Серии Cortex®-M4 STM32F

Введение

Это руководство по программированию предоставляет информацию для приложения и программного обеспечения на уровне системы разработчика. Оно дает полное описание модели программирования процессора STM32F3 И STM32F4 SERIES CORTEX®-M4, набора команд и базовых периферийных устройств.

Процессор STM32F3 И STM32F4 SERIES CORTEX®-M4 является 32-разрядным высокопроизводительным процессором, разработанным для рынка микроконтроллеров. Это предлагает существенные преимущества разработчикам, включая:

- Выдающуюся производительность обработки, объединённую с быстрой обработкой прерываний
- Улучшенную системную отладку с обширной точкой остановки и возможностями трассировки
- Эффективное ядро процессора, системы и памяти
- Сверхнизкий расход энергии с интегрированными режимами ожидания
- Безопасность платформы

Справочники

STM32F3 и Серийные таблицы данных STM32F4.

STM32F405xx/07xx, STM32F415xx/17xx, STM32F42xxx и STM32F43xxx усовершенствованный ARM®-на-основе 32-разрядного MCU (RM0090).

STM32F401xB/C и STM32F401xD/E усовершенствовали ARM®-на-основе 32-разрядного MCU (RM0368)

STM32F37xx усовершенствовал ARM®-на-основе 32-разрядного MCU (RM0313). STM32F303xB/C, STM32F303x6/8, STM32F328x8 и STM32F358xC совершенствовались ARM®-на-основе 32-разрядного MCU (RM0316).

Упомянутые выше документы доступны на www.st.com.

Это руководство по программированию применяется к продуктам, перечисленным в [Таблице 1](#).

Таблица 1. Применимые продукт

Type	Part numbers
Microcontrollers	STM32F3 Series STM32F4 Series

Содержание

1 Об этом документе.....	12
1.1 Соглашения о правилах печати.....	12
1.2 Список сокращений для регистров.....	12
1.3 О процессоре STM32 Cortex-M4 и базовых периферийных устройствах.....	13
1.3.1 Системный интерфейс уровня.....	14
1.3.2 Интегрированная конфигурируемая отладка.....	14
1.3.3 Сводка особенностей и преимуществ процессора коры-M4.....	14
1.3.4 Периферийные устройства ядра коры-M4.....	15
2 Процессор Cortex-M4.....	16
2.1 Модель программирования.....	16
2.1.1 Режим процессора и уровни полномочий для выполнения программного обеспечения.....	16
2.1.2 Стеки.....	16
2.1.3 Базовые регистры.....	17
2.1.4 Исключения и прерывания.....	25
2.1.5 Тип данных.....	25
2.1.6 Программное обеспечение микроконтроллера Кору соединённые интерфейсом со стандартом(CMSIS).....	25
2.2 Модель памяти.....	27
2.2.1 Области памяти, типы и атрибуты.....	28
2.2.2 Системное упорядочивание Памяти ,доступа памяти.....	28
2.2.3 Поведение доступа памяти.....	29
2.2.4 Упорядочивание программного обеспечения доступа памяти.....	30
2.2.5 Соединения бита.....	31
2.2.6 Порядок байтов Памяти.....	33
2.2.7 Примитив Синхронизации.....	33
2.2.8 Подсказки Программирования для примитивов синхронизации.....	35
2.3 Модель исключения.....	36
2.3.1 Состояния исключения.....	36
2.3.2 Типы Исключения.....	36
2.3.3 Обработчики исключений.....	38
2.3.4 Таблиц векторов.....	39
2.3.5 Приоритеты Исключений.....	40
2.3.6 Приоритетные группировки Прерывания.....	40
2.3.7 Записи Исключения и возврата.....	41
2.4 Обработка отказа.....	43
2.4.1 Типы отказа.....	44
2.4.2 Подъем отказа и трудно дает сбой.....	45
2.4.3 Регистры состояния отказа и отказ адреса регистра.....	46
2.4.4 Тупик.....	46
2.5 Управление электропитанием.....	46
2.5.1 Ввод режима ожидания.....	47
2.5.2 Пробуждение от режима ожидания.....	47

2.5.3 Ввод внешнего события / расширенное прерывание и ввод событий.....	48
2.5.4 Управление электропитанием, программирующие подсказки.....	48
3 Набор команд Коры-M4 STM32.....	49
3.1 Сводка набора команд.....	49
3.2 Встроенные функции CMSIS.....	57
3.3 Об описаниях инструкции.....	59
3.3.1 Операнды.....	59
3.3.2 Ограничения при использовании PC или SP.....	59
3.3.3 Гибкость вторых операндов.....	59
3.3.4 Операции Сдвига.....	61
3.3.5 Выравнивание Адреса.....	64
3.3.6 Относительные PC выражения.....	64
3.3.7 Условное выполнения.....	64
3.3.8 Выбор ширины Инструкции.....	67
3.4 Инструкции доступа памяти.....	68
3.4.1 ADR.....	69
3.4.2 LDR и STR, непосредственное смещение.....	70
3.4.3 LDR и STR, регистр смещения.....	72
3.4.4 LDR и STR, непривилегированные.....	73
3.4.5 LDR, родственник PC.....	74
3.4.6 LDM и STM.....	75
3.4.7 PUSH и POP.....	77
3.4.8 LDREX и STREX.....	78
3.4.9 CLREX.....	79
3.5 Общие инструкции обработки данных.....	80
3.5.1 ADD, ADC, SUB, SBC, и RSB.....	82
3.5.2 И, OPP, EOR, BIC, и ORN.....	84
3.5.3 ASR, LSL, LSR, ROR, и RRX.....	85
3.5.4 CLZ.....	86
3.5.5 CMP и CMN.....	87
3.5.6 MOV и MVN.....	88
3.5.7 MOVT.....	90
3.5.8 BEPCII, REV16, REVSH, и RBIT.....	91
3.5.9 SADD16 и SADD8.....	92
3.5.10 SHADD16 и SHADD8.....	93
3.5.11 SHASX и SHSAX.....	94
3.5.12 SHSUB16 и SHSUB8.....	95
3.5.13 SSUB16 и SSUB8.....	96
3.5.14 SASX и SSAX.....	97
3.5.15 TST и TEQ.....	98
3.5.16 UADD16 и UADD8.....	99
3.5.17 UASX и USAX.....	100

3.5.18 UHADD16 и UHADD8.....	101
3.5.18 UHASX и UHSAX.....	102
3.5.19 UHSUB16 и UHSUB8.....	103
3.5.20 SEL.....	104
3.5.21 USAD8.....	105
3.5.22 USADA8.....	106
3.5.23 USUB16 и USUB8.....	107
3.6 Умножители и разделители инструкции.....	108
3.6.1 MUL, MLA, и MLS.....	109
3.6.2 UMULL, UMAAL и UMLAL.....	110
3.6.3 SMLA и SMLAW.....	111
3.6.4 SMLAD.....	113
3.6.5 SMLAL и SMLALD.....	114
3.6.6 SMLSD и SMLSLD.....	116
3.6.7 SMMLA и SMMLS.....	118
3.6.8 SMMUL.....	119
3.6.9 SMUAD и SMUSD.....	120
3.6.10 SMUL и SMULW.....	121
3.6.11 UMULL, UMLAL, SMULL, и SMLAL.....	122
3.6.12 SDIV и UDIV.....	123
3.7 Насыщение инструкций.....	124
3.7.1 SSAT и USAT.....	125
3.7.2 SSAT16 и USAT16.....	126
3.7.3 QADD и QSUB.....	127
3.7.4 QASX и QSAX.....	128
3.7.5 QDADD и QDSUB.....	129
3.7.6 UQASX и UQSAX.....	130
3.7.7 UQADD и UQSUB.....	131
3.8 Упаковка и распаковка инструкций.....	132
3.8.1 PKHBT и PKHTB.....	133
3.8.2 SXT и UXT.....	134
3.8.3 SXTA и UXTA.....	135
3.9 Инструкции битового поля.....	136
3.9.1 BFC и BFI.....	137
3.9.2 SBFX и UBFX.....	138
3.9.3 SXT и UXT.....	139
3.9.4. Команды перехода и команды управления.....	140
3.9.5 B, BL, BX, и BLX.....	141
3.9.6 CBZ и CBNZ.....	143
3.9.7 IT.....	144
3.9.8 TBB и TBH.....	146
3.10 Инструкции с плавающей точкой.....	148

3.10.1 VABS.....	149
3.10.2 VADD.....	150
3.10.3 VCMPE, VCMPE.....	150
3.10.4 VCVT, VCVTR с плавающей точкой и целочисленным.....	151
3.10.5 VCVT с плавающей точкой и фиксированной точкой.....	152
3.10.6 VCVTB, VCVTT.....	153
3.10.7 VDIV.....	154
3.10.8 VFMA, VFMS.....	154
3.10.9 VFNMA, VFNMS.....	155
3.10.10 VLDM.....	156
3.10.11 VLDR.....	157
3.10.12 VLMA, VLMS.....	158
3.10.13 непосредственных VMOV.....	158
3.10.14 регистров VMOV.....	159
3.10.15 скаляров VMOV к регистру ядра ARM.....	159
3.10.16 ядер ARM VMOV A регистрируются к одинарной точности.....	160
3.10. VMOV два ядра ARM регистрируются к двум одинарной точности.....	161
3.10.18 Ядро ARM VMOV регистрируется к скаляру.....	162
3.10.19 VMRS.....	162
3.10.20 VMSR.....	163
3.10.21 VMUL.....	163
3.10.22 VNEG.....	164
3.10.23 VNMLA, VNMLS, VNMUL.....	165
3.10.24 VPOP.....	166
3.10.25 VPUSH.....	166
3.10.26 VSQRT.....	167
3.10.27 VSTM.....	167
3.10.28 VSTR.....	168
3.10.29 VSUB.....	169
3.11 Разные инструкции.....	170
3.11.1BKPT.....	170
3.11.2 CPS.....	171
3.11.3DMB.....	172
3.11.4 DSB.....	172
3.11.5 ISB.....	173
3.11.6MRS.....	173
3.11.7 MSR.....	174
3.11.8 ТОЛЬКО ДЛЯ УКАЗАННЫХ ЦЕЛЕЙ.....	175
3.11.9 SEV.....	175
3.11.10 SVC.....	176
3.11.11 WFE.....	176
3.11.12 WFI.....	177

4 Базовые периферийные устройства..... 178

4.1 О периферийных устройствах ядра Коре-М4 STM32.....	178
4.2 Модуль защиты памяти (MPU).....	178
4.2.1 Атрибуты права доступа MPU.....	180
4.2.2 несоответствия MPU.....	181
4.2.3 Обновление области MPU.....	181
4.2.4 MPU разработанные полезные советы.....	184
4.2.5 MPU ввод регистр (MPU_TYPER).....	185
4.2.6 регистр команд MPU (MPU_CTRL).....	186
4.2.7 регистр числа области MPU (MPU_RNR).....	187
4.2.8 индексные регистры области MPU (MPU_RBAR).....	188
4.2.9 Атрибут области MPU и регистр размера (MPU_RASR).....	189
4.2.10 MPU регистрируют карту.....	191
4.3 Вложенный контроллер векторного прерывания (NVIC).....	193
4.3.1 Доступ к Коре-М4 регистры NVIC, используя CMSIS.....	194
4.3.2 регистра включения набора Прерывания (NVIC_ISERx).....	195
4.3.3 регистров четкого включения Прерывания (NVIC_ICERx).....	196
4.3.4 регистров ожидания набора Прерывания (NVIC_ISPRx).....	197
4.3.5 регистров четкого ожидания Прерывания (NVIC_ICPRx).....	198
4.3.6 Прерывания активных разрядных регистров (NVIC_IABRx).....	199
4.3.7 приоритетные регистры Прерывания (NVIC_IPRx).....	200
4.3.8 триггерные регистры прерывания программного обеспечения (NVIC_STIR).....	201
4.3.9 Чувствительное к уровню и импульсное прерывание.....	202
4.3.10 полезных совета проекта NVIC.....	203
4.3.11 карты регистра NVIC.....	204
4.4 Системный блок управления (SCB).....	206
4.4.1 Вспомогательный регистр команд (ACTLR).....	207
4.4.2 индексные регистры CPUID (CPUID).....	208
4.4.3 управления Прерыванием и регистр состояния (ICSR).....	210
4.4.4 регистры смещения Таблицы векторов (VTOR).....	212
4.4.5 прерывания Приложения и регистр команд сброса (AIRCR).....	212
4.4.6 Системные регистры команд (SCR).....	214
4.4.7 Конфигурация и регистр команд (CCR).....	215
4.4.8 Системные приоритеты регистров обработчика (SHPRx).....	217
4.4.9 Системное управление обработчиком и регистр состояния (SHCSR).....	219
4.4.10 Конфигурируемые регистры состояния отказа (CFSR; UFSR+BFSR+MMFSR)...	221
4.4.11 регистр состояния отказа Исползования (UFSR).....	222
4.4.12 регистра состояния отказа Шины (BFSR).....	223
4.4.13 управления Памятью дают сбой регистр адреса (MMFSR).....	224
4.4.14 Твердые регистры состояния отказа (HFSR).....	225
4.4.15 регистры адреса отказа управления Памятью (MMFAR).....	226
4.4.16 регистры адреса отказа Шины (BFAR).....	226
4.4.17 Вспомогательные регистры состояния отказа (AFSR).....	227

4.4.18 Системные полезные советы проекта блока управления.....	227
4.4.19 SCB регистрируют карту.....	228
4.5 Таймер SysTick (STK).....	230
4.5.1 Управление SysTick и регистр состояния (STK_CTRL).....	231
4.5.2 SysTick перезагрузка регистра значения (STK_LOAD).....	232
4.5.3 Регистр текущей стоимости SysTick (STK_VAL).....	233
4.5.4 Калибровка SysTick оценка регистра (STK_CALIB).....	234
4.5.5 SysTick разработка полезных советов.....	234
4.5.6 SysTick регистрируют карту.....	235
4.6 Математический сопроцессор (FPU).....	236
4.6.1 Регистр управления доступом сопроцессора (CPACR).....	237
4.6.2 регистры команд контекста С плавающей точкой (FPCCR).....	237
4.6.3 контексты С плавающей точкой адресуют регистр (FPCAR).....	239
4.6.4 регистры команд состояния С плавающей точкой (FPSCR).....	239
4.6.5 регистры команд состояния по умолчанию С плавающей точкой (FPDSCR).....	241
4.6.6 Включение FPU.....	241
4.6.7 Включение и очистка прерываний исключения FPU.....	241
5 История версии.....	244

Список таблиц

Таблица 1. Применимые продукты.....	1
Таблица 2. Сводка режима процессора, уровня полномочий выполнения, и использования стека.....	17
Таблица 3. Базовые сводки набора регистров.....	17
Таблица 4. PSR регистрирующие комбинации.....	19
Таблица 5. разрядные определения APSR.....	20
Таблица 6. разрядные определения IPSR.....	21
Таблица 7. разрядные определения EPSR.....	22
Таблица 8. PRIMASK регистрирующие разрядные определения.....	23
Таблица 9. FAULTMASK регистрирующие разрядные определения.....	23
Таблица 10. BASEPRI регистрирующие разрядные присвоения.....	24
Таблица 11. определения бита Регистра команд.....	24
Таблица 12. Упорядочивания доступов памяти.....	28
Таблица 13. поведения доступа Памяти.....	29
Таблица 14. области соединения бита памяти SRAM.....	31
Таблица 15. Периферийная область соединения бита памяти.....	31
Таблица 16. CMSIS функционируют для инструкций эксклюзивного доступа.....	35
Таблица 17. Свойства различных типов исключения.....	37
Таблица 18. поведения возврата Исключения.....	43
Таблица 19. Отказ.....	44
Таблица 20. состояния Отказа и отказ адресованных регистров.....	46
Таблица 21. инструкции Cortex-M4.....	49
Таблица 22. встроенные функции CMSIS, для генерирования некоторых инструкций Cortex-M4.....	58
Таблица 23. встроенные функции CMSIS, для получения доступа к специальным регистрам.....	58
Таблица 24. суффиксы Кода условия.....	66

Таблица 25. инструкции доступа Памяти.....	68
Таблица 26. Непосредственные, предварительно индексированных и постиндексированных диапазоны смещения.....	71
Таблица 27. <i>PC метки</i> смещения диапазонов.....	74
Таблица 28. инструкции Data processing.....	80
Таблица 29. Умножители и делители инструкций.....	108
Таблица 30. инструкции Saturating.....	124
Таблица 31. Упаковки и распаковка инструкций.....	132
Таблица 32. Инструкции, которые работают на смежных наборах битов.....	136
Таблица 33. Команды перехода и команды управления.....	140
Таблица 34. диапазоны Ответвления.....	141
Таблица 35. инструкции Floating-point.....	148
Таблица 36. инструкции Miscellaneous.....	170
Таблица 37. базовые периферийные устройства STM32 регистрирующей области.....	178
Таблица 38. сводка атрибутов Памяти.....	179
Таблица 39. TEX, C, B, и кодирование S.....	180
Таблица 40. политика Кэша для кодирования атрибута памяти.....	180
Таблица 41. кодирования AP.....	181
Таблица 42. область Памяти приписывающая для STM32.....	184
Таблица 43. значения полей PAZMEPA B качестве примера.....	190
Таблица 44. MPU регистрирующие значения сброса и карт.....	191
Таблица 45. сводка регистра NVIC.....	193
Таблица 46. доступа CMSIS функции HВИЧА.....	194
Таблица 47. разрядных присвоения IPR.....	200
Таблица 48. CMSIS функционирующие для управления NVIC.....	203
Таблица 49. Карта регистра NVIC и значения сброса.....	204
Таблица 50. Сводки системных регистров блока управления.....	206
Таблица 51. Приоритетные группировки.....	213
Таблица 52. Системные приоритеты полей обработчика отказа.....	217
Таблица 53. SCB регистрирующие значения сброса и карта.....	228
Таблица 54. Системные таймеры регистрирующие сводку.....	230
Таблица 55. SysTick регистрирующие значения сброса и карта.....	235
Таблица 56. Коры-M4F системные регистры с плавающей точкой.....	236
Таблица 57. Эффекты сравнения C плавающей точкой на флагах условия.....	240
Таблица 58. история версии Документа.....	244

Список чисел

Рисунок 1. Реализация Коры-M4 STM32.....	13
Рисунок 2. регистры ядра Процессора.....	17
Рисунок 3. APSR, IPSR и разрядные присвоения EPSR.....	19
Рисунок 4. разрядные присвоения PSR.....	19
Рисунок 5. разрядные присвоения PRIMASK.....	23
Рисунок 6. разрядные присвоения FAULTMASK.....	23
Рисунок 7. разрядные присвоения BASEPRI.....	24
Рисунок 8. карты Памяти.....	27
Рисунок 9. отображение Разрядной полосы.....	32
Рисунок 10. примеры C прямым порядком байтов.....	33
Рисунок 11. Таблицы векторов.....	39
Рисунок 12. расположения стекового фрейма Коры-M4.....	42
Рисунок 13. ASR#3.....	61
Рисунок 14. LSR#3.....	62
Рисунок 15. LSL#3.....	62

Рисунок 16. ROR #3.....	63
Рисунок 17. RRX #3.....	63
Рисунок 18 примеры Подобласти.....	183.
Рисунок 19. отображения регистра NVIC_IPRx.....	200
Рисунок 20. подрегистры CFSR.....	221

1. Об этом документе

Этот документ предоставляет информацию, запрошенную для приложения и разработки программного обеспечения на уровне системы. Он не предоставляет информацию о компонентах отладки, функциях, или работе. Этот материал для инженеров программного и аппаратного обеспечения микроконтроллера, включая тех, кто не имеет никакого опыта продуктов ARM.

1.1 Соглашения о правилах печати

Соглашения о правилах печати, используемые в этом документе:

italic -Выделяет важные примечания, представляет специальную терминологию, обозначает внутренние перекрестные ссылки, и цитаты.

< and > -Включает заменяемые сроки для ассемблерного синтаксиса, где они появляются в коде или фрагменте кода. Например: LDRSB <cond> <Rt>, [<Rn>, #<offset>]

bold- Выделения соединяющееся интерфейсом с элементами, такими как названия меню. Обозначает сигнал имени. Также используется для сроков в послужных списках, где необходимо.

monospace- Обозначает текст, который можно ввести в клавиатуру, такую как команды, файл и названия программы, и исходный код.

monospace- Обозначает разрешенное сокращение для команды или опции. Вы можете ввести подчеркнутый текст вместо полной команды или имени опции.

monospace italic- Обозначает параметры текста моношириного, где параметр должен быть замененный определенным значением.

monospace bold -Обозначает ключевые слова языка использующиеся вне примера кода.

1.2 Список сокращений для регистров

Следующие сокращения используются в описаниях регистра:

read/write (rw)- чтение-запись (rw)- Программное обеспечение может читать и записывать в эти биты.

read-only (r)- Программное обеспечение может только считать эти биты .только для

write-only (w) -Программное обеспечение может только записать в этот бит. Чтение бита возвращает значение сброса.

read/clear (rc_w1)- программное обеспечение может считать и очистить этот бит при записи 1. Запись '0' не имеет никакого эффекта на битовое значение.

read/clear (rc_w0) -программное обеспечение может считать и очистить этот бит при записи 0. Запись '1' не имеет никакого эффекта на битовое значение.

toggle (t)-Программное обеспечение может только переключить этот бит при записи '1'. Запись '0' не имеет никакого эффекта.

Reserved (Res.)- Зарезервированный бит, должен быть сохранен в значении сброса.

1.3. О процессоре STM32 Cortex-M4 и базовых периферийных устройствах

Процессор Cortex-M4 имеет высоко производительный 32-разрядный процессор, разработанный для

рыночного микроконтроллера. Это предлагает существенные преимущества разработчикам.

Выдающаяся производительность обработки объединилась с быстрой обработкой

прерываний улучшенная системная отладка с обширной точкой остановки и

возможностями трассировки. Эффективное ядро процессора, система и память,

сверхнизкий расход энергии с интегрированными режимами ожидания устойчивости

безопасности платформы, с интегрированным модулем защиты памяти (MPU).

Процессор Cortex-M4 основывается на высокоэффективном ядре процессора, с 3-этапным

конвейером архитектуры Гарварда, делая его идеальным для того, чтобы потребовать

встраиваемые приложения. Процессор поставляет исключительную эффективность питания

через эффективный набор команд и экстенсивно оптимизированный проект, обеспечивая

высокопроизводительные аппаратные средства обработки включая IEEE754-совместимую

одинарную точность, с которой умножаются вычислением с плавающей точкой, диапазон

единственного цикла и умножения SIMD и возможностей ", накапливаются", насыщая арифметику

и выделенное аппаратное подразделение.

Реализация Кору-M4 рисунка 1. STM32

Чтобы облегчить проект чувствительных к стоимости устройств, процессор Cortex-M4 реализует системные компоненты с сильной связью, которые уменьшают область процессора, в то время как значительное улучшение обработки прерываний и системы отлаживает возможности. Процессор Cortex-M4 реализует версию набора команд Thumb®, основанного на Ползунке 2 технологии, гарантируя высокую плотность кода и уменьшенные требования к памяти программы. Набор команд Кору-M4 обеспечивает исключительную производительность, ожидаемую современной 32-разрядной архитектурой, с высокой плотностью кода 8-разрядных и 16-разрядных микроконтроллеров.

Процессор Cortex-M4 близко интегрирует конфигурируемый вложенный контроллер прерывания

(NVIC), чтобы поставить продвижение отрасли прерывают производительность. NVIC включает не - маскируемое прерывание (NMI), и обеспечивает до 256 уровней приоритета прерываний.

Трудная интеграция ядра процессора и NVIC обеспечивает быстрое выполнение процедур обработки прерывания (ISR), существенно уменьшая задержку прерывания. Это достигается посредством аппаратной укладке регистров, и возможности приостановить многократные загрузкой и многократные хранилищем операции. Обработчики прерываний не требуют никаких ассемблерных тупиков, удаляя любые издержки кода из ISR. Объединяющая в цепочку хвост оптимизации также значительно уменьшает издержки, переключаясь от одного ISR до другого. Чтобы оптимизировать проекты низкого питания, NVIC интегрирует с режимами ожидания, которые включают функцию глубокого сна, которая позволяет STM32 ввести ОСТАНОВКУ или режим STDBY.

1.3.1 Системный интерфейс уровня

Процессор Cortex-M4 обеспечивает многократные интерфейсы, используя технологию AMBA®, чтобы обеспечить высокую скорость, низкие задержки доступа памяти. Это поддерживает невыровненные доступы данных и реализует атомарную побитовую обработку, которая является более быстрым периферийным средствам управления, системным спин-блокировкам и ориентированной на многопоточное исполнение обработки Булевых данных.

У процессора Cortex-M4 есть модуль защиты памяти (MPU), который обеспечивает мелкий модуль управление памятью, позволяя приложениям использовать многократные уровни полномочий, отделяясь и защищая код, данные и стек на основе задачи задач. Такие требования являются критическими во многих встраиваемых приложениях такой как автомобильный.

1.3.2 Интегрированная конфигурируемая отладка

Процессор Cortex-M4 реализует полное аппаратное решение для отладки. Это обеспечивает высокую системную видимость процессора и памяти или через традиционный порт JTAG или через 2-контактную *Последовательную Проводную Отладку* (SWD) порт, который идеален для устройств небольшого пакета.

Поскольку системная трассировка процессора интегрирует *Макроячейку Трассировки Инструментария* (ITM) рядом с контрольными точками данных и модулем профилирования. Чтобы включить простому и рентабельному профилированию системных событий, они генерируют, *Последовательное Проводное Средство просмотра* (SWV) которое может экспортировать поток сгенерированных программным обеспечением сообщений, трассировки данных, и информации о профилировании через единственный контакт. Дополнительная *Встроенная Макроячейка Трассировки*™ (ETM) поставляет непревзойденную трассировку инструкции получения в области, намного меньшей чем традиционные модули трассировки.

1.3.3 Сводка особенностей и преимуществ процессора коры-M4

Трудная интеграция системных периферийных устройств уменьшает область и затраты на разработку набор команд ползунка комбинирует высокую плотность кода с 32-разрядной производительностью IEEE754-совместимый FPU одинарной точности, реализованный во всем STM32F4xxx и Микроконтроллеры Кору-M4 STM32F3xxx

Оптимизация управления

питание системных компонентов интегрированных режимах ожидания для низкого расхода энергии

быстрое выполнение кода разрешает более медленные часы процессора или увеличивает время режима ожидания аппаратное подразделение и быстрый множитель детерминированная, высокоэффективная обработка прерываний для строго ограниченных во времени приложений модуль защиты памяти (MPU) для важных приложений безопасности обширная отладка и возможности трассировки:

Последовательная Проводная Отладка и Последовательная Проводная Трассировка
сокращение количество контактов, требуемых для отладки и трассировки.

1.3.4 Периферийные устройства ядра коры-M4

Периферийные устройства:

Вложенный контроллер векторного прерывания

Вложенный контроллер векторного прерывания (NVIC) является встроенным контроллером прерывания это поддерживает низкую обработку прерывания задержки.

Системный блок управления

Системный блок управления (SCB) является интерфейсом модели программирования к процессору. Это предоставляет системную информацию о реализации и системное управление, включая конфигурацию, управление, и создание отчетов системных исключений.

Системный таймер

Системный таймер, SysTick, является 24-разрядным таймером обратного отсчета. Используя это в качестве Реального времени Операционная система (RTOS) отмечает таймер или как простой счетчик.

Модуль защиты памяти

Модуль защиты Памяти (MPU) улучшает системную надежность, определяя память , атрибуты для различных областей памяти. Это обеспечивает до восьми различных областей, и дополнительную предопределенную фоновую область памяти.

Модуль с плавающей точкой

Модуль С плавающей точкой (FPU) обеспечивает IEEE754-совместимые операции на единственном - точном, 32-разрядные, значения с плавающей точкой.

2 Процессор Cortex-M4

2.1 Модель программирования

Этот раздел описывает модель программирования Кору-M4. В дополнение к отдельному ядру описания регистра, это содержит информацию о режимах процессора и уровнях полномочий для выполнения программного обеспечения и стеков.

2.1.1 Режим процессора и уровни полномочий для выполнения программного обеспечения

Режимы процессора:

Режим потока: Используется, чтобы выполнить прикладное программное обеспечение. Процессор вводит режим Потока, когда это выходит из сброса. Регистр команд проверяет привилегировано ли выполнение программного обеспечения или непривилегированно, см. [Регистр команд на странице 24](#).

Режим обработчика:

Используется, чтобы обработать исключения. Процессор возвращается, чтобы Распараллелить режим, когда это закончило обработку исключения. Выполнение программного обеспечения всегда привилегировано.

Уровни полномочий для выполнения программного обеспечения:

Непривилегированный: *непривилегированное программное обеспечение* выполняется на непривилегированном уровне и:

Имеет ограниченный доступ к MSR и инструкциям MRS, и не может использовать инструкцию CPS

Не может получить доступ к системному таймеру, NVIC, или системному блоку управления мог бы иметь ограниченный доступ к памяти или периферийным устройствам должен использовать инструкцию SVC, чтобы выполнить *вызов супервизора*, чтобы передать управление к привилегированному программному обеспечению

Привилегированный: *Привилегированное программное обеспечение* выполняется на привилегированном уровне и может использовать все инструкции и доступ ко всем ресурсам. Может записать в Регистр команд, чтобы изменить уровень полномочий для выполнения программного обеспечения.

2.1.2 Стеки

Процессор использует полный убывающий стек. Это означает, что указатель вершины стека указывает на последний сложенный элемент на памяти стека. Когда процессор продвигает новый элемент на стек, он постепенно уменьшает указатель вершины стека и затем пишет элемент в новое расположение памяти. Процессор реализует два стека, *основной стек* и *стек процесса*, с независимыми копиями указателя вершины стека, смотри [Указатель вершины стека на странице 18](#).

В режиме Потока Регистр команд проверяет, использует ли процессор основной стек или стек процесса, см. [Регистр команд на странице 24](#). В режиме Обработчика процессор всегда использует основной стек. Опции для операций процессора:

Таблица 2. Сводка режима процессора, уровня полномочий выполнения, и использования стека

Процессор режим	Используемый, чтобы выполняться	Уровень полномочий для выполнения программного обеспечения	Стек используется
Поток	Приложения	Привилегированный или непривилегированный ⁽¹⁾	Основной стек или стек процесса ⁽¹⁾
Обработчик	Обработчики исключений	Всегда привилегированный	Основной стек

1. См. [Регистр команд на странице 24](#).

2.1.3 Базовые регистры

Рисунок 2. Регистры ядра процессора

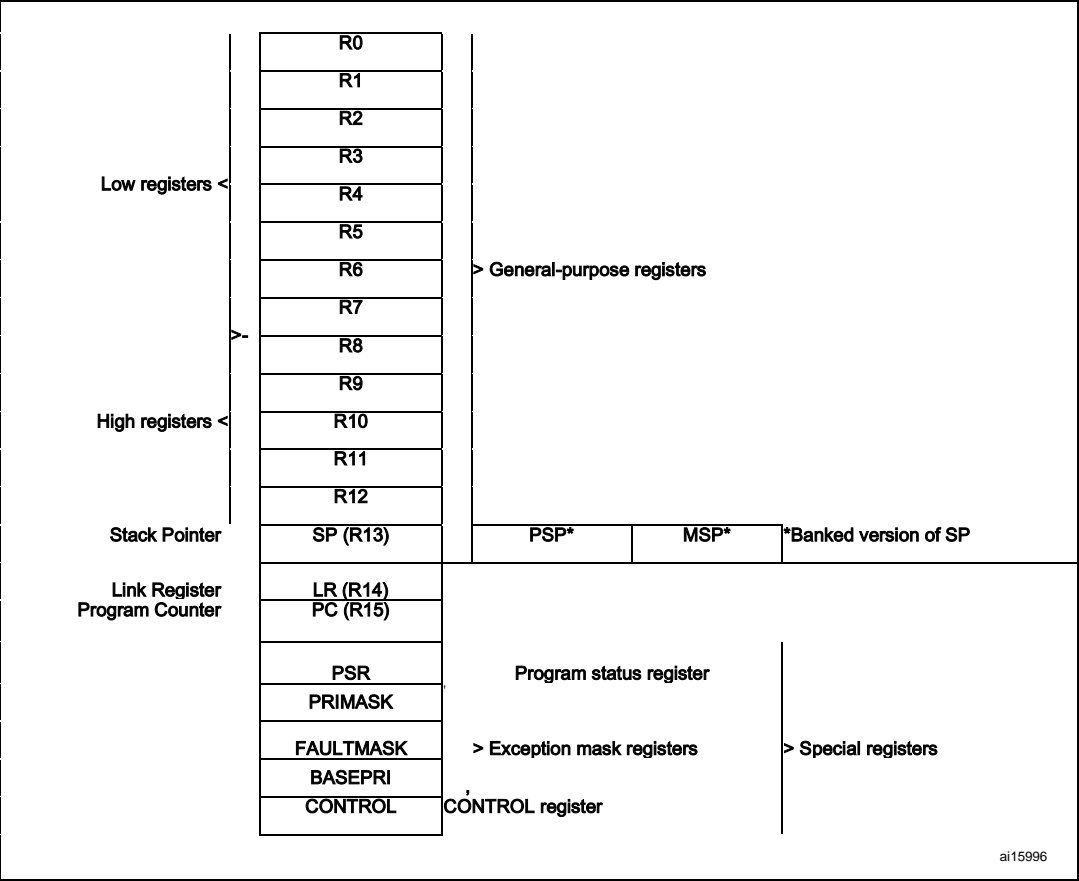


Таблица 3. Базовая сводка набора регистров

Имя	Тип (1)	Необходимые полномочия (2)	Значение сброса	Описание
R0-R12	чтение-запись	Также	Неизвестный	Регистры общего назначения на странице 18
MSP	чтение-запись	Привилегированный	См. описание	Указатель вершины стека на странице 18
PSP	чтение-запись	Также	Неизвестный	Указатель вершины стека на странице 18
LR	чтение-запись	Также	0xFFFFFFFF	Регистр ссылки на странице 18
PC	чтение-запись	Также	См. описание	Счетчик программы на странице 18

Таблица 3. Базовая сводка набора регистров (продолжение)

Имя	Тип (1)	Необходимый полномочие (2)	Значение сброса	Описание
PSR	чтение-запись	Привилегированный	0x01000000	Состояние программы регистрируется на странице 18
ASPR	чтение-запись	Также	Неизвестный	Состояние прикладной программы регистрируется на странице 20
IPSR	только для чтения	Привилегированный	0x00000000	Состояние программы прерывания регистрируется на странице 21
EPSR	только для чтения	Привилегированный	0x01000000	Состояние программы выполнения регистрируется на странице 21
PRIMASK	чтение-запись	Привилегированный	0x00000000	Приоритетная маска регистрируется на странице 23
FAULTMASK	чтение-запись	Привилегированный	0x00000000	Маска отказа регистрируется на странице 23
BASEPRI	чтение-запись	Привилегированный	0x00000000	Маска базового приоритета регистрируется на странице 24
CONTROL	чтение-запись	Привилегированный	0x00000000	Регистр команд на странице 24

1. Описывает тип доступа во время выполнения программы в режиме потока и режиме Обработчика. Доступ отладки может отличаться.
2. Запись Любого средства привилегированное и непривилегированное программное обеспечение может получить доступ к регистру.

Регистры общего назначения

R0-R12 являются 32-разрядными регистрами общего назначения для операций данных.

Указатель вершины стека

Указатель вершины стека (SP) является регистром R13. В режиме Потока, бит [1] из Регистра команд

указывает на указатель вершины стека, чтобы использовать:

0: Основной Указатель вершины стека (MSP). Это - значение сброса.

1: Указатель Стека процесса (PSP).

На сбросе процессор загружает MSP значением от адреса 0x00000000.

Регистр ссылки

Регистр Ссылки (LR) является регистром R14. Он хранит информацию возврата для подпрограмм, вызова функции, и исключения. На сбросе процессор загружает значение LR 0xFFFFFFFF.

Счетчик программы

Счетчик Программы (PC) является регистром R15. Он содержит текущий адрес программы. На сброс, процессор загружает PC значением вектора сброса, который является в адресе 0x00000004. Бит [0] из значения загружается в EPSR T-bit в сбросе и должен быть 1.

Регистр состояния программы

Регистр Состояния Программы (PSR) объединяется:

Регистр Состояния Прикладной программы (ASPR) Регистр Состояния

Программы прерывания (IPSR)

Регистр Состояния Программы выполнения (EPSR)

Эти регистры являются взаимоисключающими битовыми полями в 32-разрядном PSR. Разрядные присвоения находятся как показано в [рисунке 3](#) и [рисунке 4](#).

Рисунок 3. APSR, IPSR и разрядные присвоения EPSR

	31	30	29	28	27	26	25	24	23	20	19	16	15	10	9	8	0
APSR	N	Z	C	V	Q	Reserved				GE[3:0]			Reserved				
IPSR	Reserved													ISR_NUMBER			
EPSR	Reserved				ICI/IT		T	Reserved				ICI/IT		Reserved			

Рисунок 4. Присвоения бита PSR

	31 30 29 28 27 26 25 24 23 20 19 16 15 10															9	8	0
PSR	N	Z	C	V	Q	Reserved					GE[3:0]		ICI/IT			ISR_NUMBER		
	Reserved —																	

Получение доступа к этим регистрам индивидуально или как к комбинации любых двух или всех трех регистров, используя имя регистра в качестве параметра инструкциям MRS или MSR. Например:

Считайте все регистры, используя PSR с инструкцией MRS запишите в APSR N, Z, C, V, и биты Q, используя APSR_nzcvq с инструкцией MSR.

Комбинации PSR и атрибуты:

Таблица 4. PSR регистрируют комбинации

Регистр	Ввести	Комбинация
PSR	чтение-запись (1), (2)	APSR, EPSR, and IPSR
IEPSR	только для чтения	EPSR and IPSR
IAPSR	чтение-запись (1)	APSR and IPSR
EAPSR	чтение-запись (2)	APSR and EPSR

1. Процессор игнорирует записи к битам IPSR. 2. Чтения нуля возврата битов EPSR, и процессор игнорируют записи к этим битам

См. [MRS](#) описание инструкции [на странице 173](#) и [MSR на странице 174](#) для получения дополнительной информации о том, как получить доступ к регистрам состояния программы.

Регистр состояния прикладной программы

APSR содержит текущее состояние флагов условия от предыдущей инструкции выполнение. См. сводку регистра в [Таблице 3 на странице 17](#) для ее атрибутов. Разрядные присвоения:

Таблица 5. Определения бита APSR

Биты	Описание
Бит 31	N: Отрицательный или меньше чем флаг: 0: Результат работы был положительным, нуль, больше чем, или равный 1: результат Работы был отрицателен или меньше чем.
Бит 30	Z: Нулевой флаг: 0: Результатом работы не был нуль 1: результатом Работы был нуль.
Бит 29	C: Перенесите или заимствуйте флаг: 0: Добавляет, что работа не приводила к биту переноса или вычитала работу, приведенную к заимствованному биту 1: Добавляет, что работа привела к биту переноса, или вычитите работу, не приводил к заимствованному биту.
Бит 28	V: Флаг переполнения: 0: Работа не приводила к переполнению 1: Работа привела к переполнению.
Бит 27	Q: Переполнение DSP и флаг насыщенности: Липкий флаг насыщенности. 0: Указывает, что насыщенность не произошла так как сброшено или так как бит был последний очищенный к нулю 1: Указывает, когда инструкция SSAT ИЛИ USAT приводит к насыщенности, или указывает на переполнение DSP. Этот бит очищается, чтобы обнулить программным обеспечением, используя инструкцию MRS.
Биты 26:20	Зарезервированный.
Биты 19:16	GE [3:0]: Больше чем или Равные флаги. См. <i>SEL на странице 104</i> для получения дополнительной информации.
Биты 15:0	Зарезервированный.

Регистр состояния программы прерывания

IPSR содержит число типа исключения текущей *Процедуры обработки прерывания* (ISR). См. сводку регистра в [Таблице 3 на странице 17](#) для ее атрибутов. Разрядные присвоения:

Таблица 6. Определения бита IPSR

Биты	Описание
Биты 31:9	Зарезервированный
Биты 8:0	ISR_NUMBER: Это - число текущего исключения: 0: режим Потока 1: Зарезервированный 2: NMI 3: Твердый отказ 4: управление Памятью дает сбой 5: отказ Шины 6: отказ Использования 7: Зарезервированный.... 10: Зарезервированный 11: SVCcall 12: Зарезервированный для Отладки 13: Зарезервированный 14: PendSV 15: SysTick 16: IRQ0 (1)..... 83: IRQ81 (1)смотри <i>типы Исключения на странице 36</i> для получения дополнительной информации.

1. См. справочник продукта STM32 / таблица данных для получения дополнительной информации об отображении прерывания

Регистр состояния программы выполнения

EPSR содержит бит состояния Ползунка, и биты режима выполнения для любого:

Если Затем (IT) инструкция прерывистая-Continuable Инструкция (ICI) поле для прерванной многократной загрузки или хранилище многоадресная команда.

См. сводку регистра в [Таблице 3 на странице 17](#) для атрибутов EPSR. Бит присвоения:

Таблица 7. Определения бита EPSR

Биты	Описание
Биты 31:27	Зарезервированный.
Биты 26:25, 15:10	ICI: прерывистые-continuable биты инструкции, см. <i>инструкции Interruptible-continuable на странице 22.</i>
Биты 26:25, 15:10	IT: Указывает на биты режима выполнения инструкции IT, см. <i>IT на странице 144.</i>
Бит 24	T: Бит состояния ползунка.
Биты 23:16	Зарезервированный.
Биты 9:0	Зарезервированный.

Попытки считать EPSR непосредственно через прикладное программное обеспечение, используя инструкцию MSR всегда возвращают нуль. Попытки записать EPSR использование инструкции MSR в прикладном программном обеспечении игнорируются. Обработчики отказа могут исследовать значение EPSR в сложном PSR, чтобы указать на работу, которая ошибается. См. [Раздел 2.3.7: запись Исключения и возврат на странице 41.](#)

Прерывистые-continuable инструкции

Когда прерывание происходит во время выполнения LDM STM, ПРОДВИНЬТЕ, POP, VLDM, VSTM, VPUSH, или инструкция VPOP, процессор:

A diagram showing a 32-bit field labeled "Reserved". The field is represented as a horizontal bar with vertical tick marks at positions 31, 30, 29, 28, 27, 26, 25, and 24, indicating bit boundaries. The number "31" is at the left end and "10" is at the right end of the bar.

Таблица 8. PRIMASK регистрируют разрядные определения

Биты	Описание
Биты 31:1	Зарезервированный
Бит 0	PRIMASK: 0: Никакой эффект 1: Предотвращает активацию всех исключений с конфигурируемым приоритетом.

Регистр маски отказа

Регистр FAULTMASK предотвращает активацию всех исключений за исключением

Немаскируемого

Прерывание (NMI). См. сводку регистра в [Таблице 3 на странице 17](#) для ее атрибутов. [Рисунок 6](#) показывает разрядные присвоения.

Рисунок 6. Присвоения бита FAULTMASK

31								1 0
Reserved								

Таблица 9. FAULTMASK регистрируют разрядные определения

Биты	Функция
Биты 31:1	Зарезервированный
Бит 0	FAULTMASK: 0: Никакой эффект 1: Предотвращает активацию всех исключений за исключением NMI.

Процессор очищает бит FAULTMASK к 0 на выходе от любого обработчика исключений кроме обработчика NMI.

Регистр маски базового приоритета

Регистр BASEPRI определяет минимальный приоритет для обработки исключения. Когда BASEPRI устанавливается в ненулевое значение, он предотвращает активацию всех исключений с тем же самым или более низким приоритетным уровнем как значение BASEPRI. См. сводку регистра в [Таблице 3 на странице 17](#) для ее атрибутов. [Рисунок 7](#) показывает разрядные присвоения.

Рисунок 7. Присвоения бита BASEPRI

31					8	7		0
Reserved						BASEPRI		

Таблица 10. BASEPRI регистрируют разрядные присвоения

Биты	Функция
Биты 31:8	Зарезервированный
Биты 7:4	BASEPRI [7:4] Приоритетные биты маски (1) 0x00: никакой Ненулевой эффект: определяет базовый приоритет для обработки исключения. Процессор не обрабатывает исключения с приоритетным значением, больше чем или равный BASEPRI
Биты 3:0	Зарезервированный

. 1. Это поле подобно приоритетным полям в приоритетных регистрах прерывания. См. [приоритетные регистры Прерывания \(NVIC_IPRx\) на странице 200](#) для получения дополнительной информации. Помните, что более высокие приоритетные значения полей соответствуют более низким приоритетам исключения.

Регистр команд

Регистр команд управляет используемым стеком и уровнем полномочий для программного обеспечения, когда процессор находится в режиме Потока и указывает, является ли состояние FPU активным. См. сводку регистра в [Таблице 3 на странице 17](#) для ее атрибутов.

Таблица 11. Определения бита регистра команд

Биты	Функция
Биты 31:3	Зарезервированный
Бит 2	FPCA: Указывает ли в настоящий момент активный контекст с плавающей точкой: 0: Никакой контекст с плавающей точкой активный 1: активный контекст с плавающей точкой. Кора-M4 использует этот бит, чтобы определить, сохранить ли состояние с плавающей точкой, обрабатывая исключение.
Бит 1	SPSEL: Активный выбор указателя вершины стека. Выбирает текущий стек: 0: MSP является текущим указателем вершины стека 1: PSP является текущим указателем вершины стека. В режиме Обработчика этот бит читает как ноль и игнорирует записи. Кора-M4 обновляет этот бит автоматически по возврату исключения.
Бит 0	nPRIV: уровень полномочий режима Потока. Определяет уровень полномочий режима Потока. 0: Привилегированный 1: непривилегированный.

Режим обработчика всегда использует MSP, таким образом, процессор игнорирует явные записи к активному биту указателя вершины стека Регистра команд когда в режиме Обработчика. Механизмы записи и возврата исключения обновляют Регистр команд. В среде ОС рекомендуется, чтобы потоки, работающие в режиме Потока, использовали стек процесса и ядро и обработчик исключений использует основной стек. По умолчанию, режим Потока использует MSP. Переключить указатель вершины стека, используемый в режим Потока к PSP, также:

Используйте инструкцию MSR, чтобы установить Активный бит указателя вершины стека в 1, смотри [MSR на странице 174](#). выполните возврат исключения, чтобы Распараллелить режим соответствующим EXC_RETURN оцените, см. [поведение возврата Исключения на странице 43](#). Изменяя указатель вершины стека, программное обеспечение должно сразу использовать инструкцию ISB после инструкция MSR. Это гарантирует, что инструкции после ISB выполняют использование нового указателя вершины стека. См. [ISB на странице 173](#)

2.1.4 Исключения и прерывания

Процессор Cortex-M4 поддерживает системные исключения и прерывания. Процессор и *Вложенный Контроллер Векторного прерывания* (NVIC) располагает по приоритетам и обрабатывает все исключения. Исключение изменяет нормальный поток управления программным обеспечением. Процессор использует режим обработчика, чтобы обработать все исключения за исключением сброса. См. [запись Исключения на странице 41](#) и [возврат Исключения на странице 43](#) для получения дополнительной информации. Регистры NVIC управляют обработкой прерываний. См. [Вложенный контроллер векторного прерывания \(NVIC\) на странице 193](#) для получения дополнительной информации.

2.1.5 Типы данных

Процессор:

Поддерживает следующие типы данных:

-32-разрядные слова-

16-разрядные полуслова-

8-разрядные байты

управляет всеми доступами памяти как прямым порядком байтов. См. [области Памяти, типы и атрибуты на странице 28](#) для получения дополнительной информации.

2.1.6 Программное обеспечение микроконтроллера Кору соединяет интерфейсом со стандартом (CMSIS)

Для системы микроконтроллера Кору-M4, *Интерфейса программного обеспечения*

Микроконтроллера Кору Стандарт (CMSIS) определяет:

Распространенный способ к:

- Периферийным регистрам доступа
- Определителям вектора исключения

Имена:

- Регистры базовых периферийных устройств
- Базовые векторы исключения

Независящий от устройств интерфейс для ядер RTOS, включает канал отладки

CMSIS включает определения адреса и структуры данных для базовых периферийных устройств в процессоре Cortex-M4.

CMSIS упрощает разработку программного обеспечения, включая повторному использованию шаблона кода и комбинация CMSIS-совместимых компонентов программного обеспечения от различных поставщиков промежуточного программного обеспечения. Поставщики программного обеспечения могут развернуть CMSIS, чтобы включать их периферийные определения и функции доступа для всех периферийных устройств.

Этот документ включает имена регистра, определенные CMSIS, и дает короткие описания функций CMSIS, которые адресуют ядро процессора и базовые периферийные устройства.

Этот документ использует краткие названия регистра, определенные CMSIS. В нескольких случаях они отличаются от архитектурных кратких названий, которые могли бы использоваться в других документах.

Следующие разделы дают больше информации о CMSIS:

[Раздел 2.5.4: Управление электропитанием, программирующее подсказки на странице 48](#)

[встроенные функции CMSIS на странице 57 регистры включения набора прерывания \(NVIC_ISERx\) на странице 195](#)

[NVIC, программирующий подсказки на странице 203](#)

22 Модель памяти

Этот раздел описывает карту памяти процессора, поведение доступов памяти, и соединяющие бит функции. У процессора есть фиксированная карта памяти, которая обеспечивает до 4 Гбайт адресуемой памяти.

Рисунок 8. Карта памяти

Области для SRAM и периферийных устройств включают области разрядной полосы. Соединение бита обеспечивает атомарные операции для разрядных данных, см. [Раздел 2.2.5: соединение бита на странице 31](#).

Процессор резервирует области *Частной периферийной шины* (PPB) адресное пространство для ядра

периферийные регистры, см. [Раздел 4.1: О периферийных устройствах ядра Кори-M4 STM32 на странице 178](#).

2.2.1 Области памяти, типы и атрибуты

Карта памяти и программирование MPU разделяют карту памяти на области.

У каждой области есть определенный тип памяти, и у некоторых областей есть дополнительные атрибуты памяти. Тип памяти и атрибуты определяют поведение доступов к области.

Типы памяти:

Нормальный Процессор может переупорядочить транзакции для эффективности, или выполнить спекулятивные чтения.

Устройство Процессор сохраняет порядок транзакции относительно другого транзакции к Устройству или Строго упорядоченной памяти.

Строго упорядоченный Процессор сохраняет порядок транзакции относительно всех другой транзакции.

Различные требования упорядочивания для Устройства и Строго упорядоченной памяти означают, что система памяти может буферизовать запись к памяти Устройства, но не должна буферизовать запись к Строго - упорядоченной памяти.

Дополнительные атрибуты памяти включают:

Не выполнится Никогда (XN) Означает, что процессор предотвращает доступы инструкции. Любая попытка выбрать инструкцию причин области XN исключает отказ управления памятью.

2.2.2 Системное упорядочивание памяти доступов памяти

Для большинства доступов памяти, вызванных явными инструкциями доступа памяти, памятью система не гарантирует, что порядок, в котором доступы полные соответствия порядку программы инструкций, обеспечивая это не влияет на поведение последовательности инструкции. Обычно, если корректное выполнение программы зависит от двух доступов памяти, завершающихся в порядке программы, программное обеспечение должно вставить инструкцию барьера памяти между инструкциями доступа памяти, смотри [Раздел 2.2.4: упорядочивание программного обеспечения доступов памяти на странице 30](#).

Однако, система памяти действительно гарантирует некоторое упорядочивание доступов к Устройству и Строго упорядоченной памяти. Для двух инструкций A1 и A2 доступа памяти, если A1 происходит перед A2 в порядке программы, упорядочивание доступов памяти, вызванных двумя инструкциями:

Таблица 12. Упорядочивание доступов памяти (1)

A1	A2			
	Нормальный доступ	Устройство		Строго упорядоченный доступ
		Необщий доступ	Общий	
Нормальный доступ	-	-	-	-
Доступ устройства, необщий	-	<	-	<
Доступ устройства, общий	-	-	<	<
Строго упорядоченный доступ	-	<	<	<

1. - средства, что система памяти не гарантирует упорядочивания доступов.

<средство, что доступы наблюдаются в порядке программы, то есть, A1, всегда наблюдается перед A2.

2.2.3 Поведение доступов памяти

Поведение доступов к каждой области в карте памяти:

Таблица 13. Поведение доступа памяти

Адрес диапазон	Память область	Память ввести	XN	Описание
0x00000000-0x1FFFFFFF	Код	Нормальный (1)	-	Исполнимая область для кода программы. Может также поместить данные здесь
0x20000000-0x3FFFFFFF	SRAM	Нормальный (1)	-	Исполнимая область для данных. Может также поместить код здесь. Эта область включает разрядную полосу и разрядные области псевдонима полосы, см. <i>Таблицу 14 на странице 31</i> .
0x40000000-0x5FFFFFFF	Периферийное устройство	Устройство (1)	XN ⁽¹⁾	Эта область включает разрядную полосу и разрядные области псевдонима полосы, см. <i>Таблицу 15 на странице 31</i> .
0x60000000-0x9FFFFFFF	Внешняя RAM	Нормальный (1)	-	Исполнимая область для данных.
0xA0000000-0xDFFFFFFF	Внешнее устройство	Устройство (1)	XN ⁽¹⁾	Внешняя память Устройства
0xED000000-0xED0FFFFF	Частная Периферийная Шина	Строго - упорядоченный (1)	XN ⁽¹⁾	Эта область включает НВИЧА, Системный таймер, и системный блок управления
0xED100000-0xFFFFFFF	Память отображала периферийные устройства	Устройство (1)	XN ⁽¹⁾	Эта область включает все стандартные периферийные устройства STM32.

1. См. [области Памяти, типы и атрибуты на странице 28](#) для получения дополнительной информации.

Код, SRAM, и внешние области RAM могут содержать программы. Однако, рекомендуется, чтобы программы всегда использовали область Кода. Это, потому что у процессора есть отдельные шины, которые позволяют выборкам инструкции и доступам данных произойти одновременно. MPU может переопределить поведение доступа памяти по умолчанию, описанное в этом разделе. Для больше информации, см. [модуль защиты Памяти \(MPU\) на странице 178](#).

2.2.4 Упорядочивание программного обеспечения доступов памяти

Порядок инструкций в процессе выполнения программы не всегда гарантирует порядок соответствующие транзакции памяти. Это, потому что:

Процессор может переупорядочить некоторые доступы памяти, чтобы улучшить эффективность, обеспечивая это не влияет на поведение последовательности инструкции. У процессора есть многократные интерфейсы шины. У памяти или устройств в карте памяти есть различные состояния ожидания некоторых доступов памяти буферизуются или спекулятивные. [Раздел 2.2.2: системное упорядочивание Памяти доступов памяти на странице 28](#) описывает случаи где системные гарантии памяти порядок доступов памяти. Иначе, если порядок доступов памяти является критическим, программное обеспечение должно включать инструкции барьера памяти, чтобы вызвать упорядочивание. Процессор обеспечивает следующие инструкции барьера памяти:

DMB Барьер Памяти Данных (DMB), инструкция гарантирует, что выдающиеся транзакции памяти завершаются перед последующими транзакциями памяти. См. [DMB на странице 172](#).

DSB Барьер Синхронизации Данных (DSB) инструкция гарантирует что выдающийся транзакции памяти, полные перед последующими инструкциями, выполняются. См. [DSB на странице 172](#).

ISB Барьер Синхронизации Инструкции (ISB) гарантирует что эффект всех завершенные транзакции памяти являются распознаваемыми последующими инструкциями. См. [ISB на странице 173](#).

Использование инструкции барьера памяти , например:

Таблица векторов. Если программа изменяет запись в таблице векторов, и затем включает соответствующее исключение, используйте инструкцию DMB между операциями. Это гарантирует, что, если исключение берется, сразу будучи включенным, процессор использует новый вектор исключения.

Самоизменение кода. Если программа содержит код самоизменения, используйте ISB инструкция сразу после модификации кода в программе. Это гарантирует, что последующее выполнение инструкции использует обновленную программу.

Переключение карты памяти. Если система содержит карту памяти, переключающую механизм, используйте инструкцию DSB после переключения карты памяти в программе. Это гарантирует, что последующее выполнение инструкции использует обновленную карту памяти.

Динамическое приоритетное изменение исключения. Приоритет исключения должен измениться когда исключение находится на рассмотрении или является активным, используйте инструкции DSB после изменения. Это гарантирует, что изменение вступает в силу на завершении инструкции DSB.

Используя семафор в мультиосновной системе. Если система содержит больше чем один мастер шины, например, если другой процессор присутствует в системе, каждый процессор, должен использовать инструкцию DMB после любых семафорных инструкций, чтобы гарантировать, что другие мастера шины видят транзакции памяти в порядке, в котором они выполнялись.

Доступы памяти к Строго упорядоченной памяти, такие как системный блок управления, не делают потребуйте использования инструкций DMB.

Для программирования MPU, используйте DSB, сопровождаемый инструкцией ISB, или исключите возврат гарантии, что новая конфигурация MPU используется последующими инструкциями.

2.2.5 Соединение бита

Область разрядной полосы отображает каждое слово в области *псевдонима разрядной полосы* к единственному биту в *разрядной полосе области*. Область разрядной полосы занимает самый низкий 1 Мбайт SRAM и периферийную область памяти.

У карты памяти есть две области псевдонима на 32 Мбайта, которые отображаются на две области разрядной полосы на 1 Мбайт:

Доступы к SRAM на 32 Мбайта искажают карту области к области разрядной полосы SRAM на 1 Мбайт, как показано в [Таблице 14](#) Доступы к периферийному устройству на 32 Мбайта искажают карту области к периферийной разрядной полосе на 1 Мбайт области, как показано в [Таблице 15](#).

Таблица 14. Области соединения бита памяти SRAM

Адрес диапазон	Память область	Инструкция и доступы данных
0x20000000-0x200FFFFF	Область разрядной полосы SRAM	Прямые доступы к этому диапазону памяти ведут себя как доступы памяти SRAM, но эта область является также битом, адресуемым через псевдоним разрядной полосы.
0x22000000-0x23FFFFFF	Псевдоним разрядной полосы SRAM	Доступы данных к этой области повторно отображаются на разрядную область полосы. Операция записи выполняется, поскольку "чтение изменяет запись". Доступы инструкции не повторно отображаются.

Таблица 15. Периферийные области соединения бита памяти

Адрес диапазон	Память область	Инструкция и доступы данных
0x40000000-0x400FFFFF	Периферийная область разрядной полосы	Прямые доступы к этому диапазону памяти ведут себя как периферийные доступы памяти, но эта область является также битом, адресуемым через псевдоним разрядной полосы.
0x42000000-0x43FFFFFF	Периферийный псевдоним разрядной полосы	Доступы данных к этой области повторно отображаются на область разрядной полосы. Операция записи выполняется, поскольку "чтение изменяет запись". Доступы инструкции не повторно отображаются.

Отметьте:

Доступ слова к SRAM или периферийным областям псевдонима разрядной полосы отображается на единственный бит в SRAM или периферийной области разрядной полосы. Разрядные доступы полосы могут использовать байт, полуслово, или передачи слова. Разрядный размер передачи полосы соответствует размеру передачи инструкции, делающей разрядный доступ полосы.

Следующая формула показывает, как область псевдонима отображается на область разрядной полосы:

```
bit_word_offset = (byte_offset x 32) + (bit_number x 4)
bit_word_addr = bit_band_base + bit_word_offset
```

Где:

Bit_word_offset является позицией целевого бита в области памяти разрядной полосы. Bit_word_addr является адресом слова в области памяти псевдонима, которая отображается на предназначенный бит.

Bit_band_base является начальным адресом области псевдонима.

Byte_offset является числом байта в области разрядной полосы, которая содержит предназначенный бит.

Bit_number является позицией двоичного разряда, 0-7, предназначенного бита.

Рисунок 9 на странице 32 показывает примеры разрядной полосы, отображающейся между областью псевдонима разрядной полосы SRAM и областью разрядной полосы SRAM:

Слово псевдонима в 0x23FFFFED отображается на бит [0] из байта разрядной полосы в 0x200FFFFF: $0x23FFFFED = 0x20000000 + (0xFFFF * 32) + (0 * 4)$.

Слово псевдонима в 0x23FFFFFC отображается на бит [7] из байта разрядной полосы в 0x200FFFFF: $0x23FFFFFC = 0x20000000 + (0xFFFF * 32) + (7 * 4)$.

Слово псевдонима в 0x22000000 отображается на бит [0] из байта разрядной полосы в 0x20000000: $0x22000000 = 0x20000000 + (0 * 32) + (0 * 4)$.

Слово псевдонима в 0x2200001C отображается на бит [7] из байта разрядной полосы в 0x20000000: $0x2200001C = 0x20000000 + (0 * 32) + (7 * 4)$.

Рисунок 9. Отображение разрядной полосы

Непосредственно получая доступ к области псевдонима

Запись в слово в области псевдонима обновляет единственный бит в области разрядной полосы. Бит [0] из значения, записанного слову в области псевдонима, определяет значение, записанное предназначенным битом в области разрядной полосы. Запись значения с битом [0] набор к 1 записи 1 к биту - бит полосы, и при записи значения с битом [0] набор к 0 записям 0 к биту разрядной полосы. Биты [31:1] слова псевдонима не имеют никакого эффекта на бит разрядной полосы. У записи 0x01 есть то же самое

эффект как пишущий 0xFF. Запись 0x00 имеет тот же самый эффект как пишущий 0x0E.

Чтение слова в области псевдонима:

0x00000000 указывает, что предназначенный бит в области разрядной полосы

обнуляется 0x00000001 указывает, что предназначенный бит в области разрядной полосы устанавливается в 1

Непосредственно доступ к [Поведению](#) области разрядной полосы [доступов памяти на странице 29](#) описывает поведение прямого байта, полуслова, или доступы слова к областям разрядной полосы.

2.2.6

Порядок байтов памяти

Процессор просматривает память как линейный набор байтов, пронумерованных в порядке возрастания от нуля. Например, байты 0-3 содержат первое сохраненное слово, и байты 4-7 содержат второе сохраненное слово.

Формат с прямым порядком байтов

В формате с прямым порядком байтов процессор хранит младший значащий байт слова в самом низком - пронумерованный байт, и старший значащий байт в байте с самым высоким номером.

См. [рисунок 10](#) для примера.

Рисунок 10. Пример с прямым порядком байтов

31	24	23	16	15	8	7	0
B3		B2		B1		B0	

2.2.7 Примитивы синхронизации

Набор команд Коры-M4 включает пары *примитивов синхронизации*. Они обеспечивают не блокирование механизма, которые могут использовать поток или процесс, чтобы получить эксклюзивный доступ к расположению памяти. Программное обеспечение может использовать их, чтобы выполнить гарантируемое чтение "обновления памяти, изменяют запись" последовательность, или для семафорного механизма.

Пара примитивов синхронизации включает:

Инструкция Load-Exclusive: Используемый, чтобы считать значение расположения памяти, запрашивая эксклюзивный доступ к тому расположению.

Инструкция Store-Exclusive: Используемый, чтобы попытаться записать в то же самое расположение памяти, возврат бита состояния к регистру. Если этот бит: 0: поток или процесс получали эксклюзивный доступ к памяти, и запись успешно выполняется 1: поток или процесс не получали эксклюзивный доступ к памяти, и никакая запись не выполняется.

Пары инструкций Load-Exclusive и Store-Exclusive:

Инструкции LDREX и STREX слова инструкции LDREXH и STREXH полуслова байтовые команды LDREXB и STREXB.

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

Чтобы выполнить гарантируемое "чтение изменяют запись" расположения памяти, программное обеспечение должно:

1. Использовать инструкцию Load-Exclusive, чтобы считать значение расположения.
2. Обновить значение, как требуется.
3. Использовать инструкцию Store-Exclusive, чтобы попытаться записать новое значение обратно к памяти расположения.
4. Протестировать возвращенный бит состояния. Если этот бит: 0: "чтение изменяет запись", завершенную успешно, 1: Никакая запись не выполнялась. Это указывает, что значение, возвращенное в шаге 1, могло бы быть устаревшим. Программное обеспечение должно повторить последовательность "чтение, изменяют запись",

Программное обеспечение может использовать примитивы синхронизации, чтобы реализовать семафоры следующим образом:

1. Использовать инструкцию Load-Exclusive, чтобы читать из семафорного адреса, чтобы проверить, свободен ли семафор.
2. Если семафор свободен, использовать Монопольное хранилище, чтобы записать, что требование оценивает семафорному адресу.
3. Если возвращенный бит состояния от шага 2 указывает, что Монопольное хранилище, за которым следуют тогда программное обеспечение, требовало семафора. Однако, если отказавшее Монопольное хранилищем, другой процесс, возможно, требовал семафора после того, как программное обеспечение выполняло шаг 1.

Кора-M4 включает монитор эксклюзивного доступа, который тегирует факт что процессор выполнил инструкцию Load-Exclusive. Если процессор является частью многопроцессорной системы, система также глобально тегирует расположения памяти, адресуемые эксклюзивными доступами каждого процессора.

Процессор удаляет свой тег эксклюзивного доступа если:

Это выполняет инструкцию CLREX

ЭТО выполняет инструкцию Store-Exclusive, независимо от того, успешно выполняется ли запись. исключение происходит. Это означает, что процессор может разрешить семафорные конфликты между различными потоками.

В многопроцессорной реализации, выполняется:

Инструкция CLREX удаляет только локальный тег эксклюзивного доступа для процессора монопольного хранилища инструкции, или исключения. удаляет локальные теги эксклюзивного доступа, и глобальный эксклюзивный доступ тегирует для процессора.

Для получения дополнительной информации о синхронизации примитивные инструкции, см. [LDREX](#) и

[STREX на странице 78](#) и [CLREX на странице 79](#).

2.2.8 Программирование подсказок для примитивов синхронизации

ISO/IEC C не может непосредственно генерировать инструкции эксклюзивного доступа. CMSIS обеспечивает встроенные функции для генерации этих инструкций:

Таблица 16. CMSIS функционирует для инструкций эксклюзивного доступа

Инструкция	Функция CMSIS
LDREX	uint32_t __LDREXW (uint32_t *addr)
LDREXH	uint16_t __LDREXH (uint16_t *addr)
LDREXB	uint8_t __LDREXB (uint8_t *addr)
STREX	uint32_t __STREXW (uint32_t value, uint32_t *addr)
STREXH	uint16_t __STREXH (uint16_t value, uint16_t *addr)
STREXB	uint8_t __STREXB (uint8_t value, uint8_t *addr)
CLREX	void __CLREX (void)

Например:

```
uint16_t      значение;
uint16_t *address = 0x20001002;
оцените = __ LDREXH (адрес);           //загружаются, 16-разрядное значение из
памяти адресуются
//0x20001002
```

2.3 Модель исключения

Этот раздел описывает модель исключения.

2.3.1 Состояния исключения

Каждое исключение находится в одном из следующих состояний:

Неактивный Исключение не является активным и не на ожидание.

Ожидание Исключение ожидает, чтобы быть обслуженным процессором. Прерывает запрос от периферийного устройства или от программного обеспечения может изменить состояние соответствующего прерывания к ожиданию.

Активный Исключение, которое обслуживается процессором, но не имеет завершений.

Отметьте: обработчик исключений может прервать выполнение другого исключения обработчика. В этом случае оба исключения находятся в активном состоянии.

Активный и на ожидании исключение обслуживается процессором и исключение на ожидании из того же самого источника.

2.3.2 Типы исключения

Типы исключения:

Сброс Сброс вызывается на, включении или теплом сбросе. Сброс обработок модели исключения как специальная форма исключения. Когда сброс утверждается, работа остановок процессора, потенциально в любой точке в инструкции. Когда сброс является deasserted, перезапускается выполнения от адреса, обеспеченного записью сброса в таблице векторов. Выполнение перезапускает как привилегированное выполнение в режиме Потока.

NMI *NonMaskable* (NMI) может быть сообщен периферийным устройством или инициированный программным обеспечением. Это - самое высокое приоритетное исключение кроме сброса. Оно постоянно включается и имеет фиксированный приоритет-2. NMIs не может быть:

Замаскированным или предотвращенным от активации любым другим исключением вытесненным любым исключением кроме Сброса.

Твердый отказ Твердый отказ является исключением, которое происходит из-за ошибки во время обработки исключения, или потому что исключением не может управлять никакой другой механизм исключения. У твердых отказов есть фиксированный приоритет-1, означая, что у них есть более высокий приоритет чем любое исключение с конфигурируемым приоритетом.

Отказ управления памятью Отказ управления памятью является исключением, которое происходит из-за защиты памяти связанным отказом. MPU или фиксированным ограничением защиты памяти которое определяет этот отказ, и для инструкции и для транзакций памяти данных. Этот отказ используется, чтобы прервать доступы инструкции, чтобы никогда не *Выполнять* (XN) области памяти.

Отказ шины Отказ шины является исключением, которое происходит из-за памяти связанный отказ для инструкции или транзакции памяти данных. Это могло бы быть от ошибки, обнаруженной на шине в системе памяти.

Отказ использования Отказ использования является исключением, которое происходит из-за отказа, связанного с выполнением инструкции. Это включает:

Неопределенная инструкция недопустимый невыровненный доступ
недопустимое состояние на выполнении инструкции ошибка по возврату
исключения.

Следующее может вызвать отказ использования, когда ядро конфигурируется, чтобы сообщить о них:

Невыровненный адрес на доступе памяти слова и полуслова подразделение нулем

SVCall *Вызов супервизора* (SVC) является исключением, которое инициировано инструкцией SVC. В среде ОС приложения могут использовать инструкции SVC, чтобы получить доступ к функциям ядра ОС и драйверам устройств.

PendSV PendSV является управляемым прерыванием запроса на службу на уровне системы. В Среде ОС, используется PendSV для контекстного переключения, когда никакое другое исключение не является активным.

SysTick SysTick является исключением, которое системный таймер генерирует когда оно достигает нуля. Программное обеспечение может также генерировать исключение SysTick. В среде ОС процессор может использовать это исключение в качестве системной галочки. Прерывание (IRQ) Прерывание, или IRQ, является исключением, сообщенным периферийным устройством, или сгенерированным запросом программного обеспечения. Все прерывания являются асинхронными к выполнению инструкции. В системе периферийные устройства используют прерывания, чтобы связаться с процессором.

Таблица 17. Свойства различных типов исключения

Исключение номер (1)	IRQ номер (1)	Исключение ввести	Приоритет	Векторный адрес или смещение (2)	Активация
1	-	Сброс	-3, самое высокое	0x00000004	Асинхронный
2	-14	NMI	-2	0x00000008	Асинхронный
3	-13	Твердый отказ	-1	0x0000000C	-
4	-12	Отказ управления памятью	Конфигурируемый (3)	0x00000010	Синхронный
5	-11	Отказ шины	Конфигурируемый (3)	0x00000014	Синхронный, когда точный Асинхронный когда неточный
6	-10	Отказ использования	Конфигурируемый (3)	0x00000018	Синхронный
7-10	-	-	-	Зарезервированный	-
11	-5	SVCall	Конфигурируемый (3)	0x0000002C	Синхронный
12-13	-	-	-	Зарезервированный	-
14	-2	PendSV	Конфигурируемый (3)	0x00000038	Асинхронный

Таблица 17. Свойства различных типов исключения (продолжение)

Исключение номер (1)	IRQ номер (1)	Исключение ввести	Приоритет	Векторный адрес или смещение (2)	Активация
15	-1	SysTick	Конфигурируемый (3)	0x0000003C	Асинхронный
16 и выше	0 и выше	Прерывание (IRQ)	Конфигурируемый (4)	0x00000040 и выше (5)	Асинхронный

1. Чтобы упростить уровень программного обеспечения, CMSIS только использует числа IRQ и поэтому использует отрицательные величины для исключений другой чем прерывания. IPSR возвращает число Исключения, см., *что состояние программы Прерывания регистрируется на странице 21.*

2. См. *Таблицу векторов на странице 39* для получения дополнительной информации. 3. См. *Системные приоритетные регистры обработчика (SHPRx) на странице 217.* 4. См. *приоритетные регистры Прерывания (NVIC_IPRx) на странице 200.*

5. Увеличение в шагах 4.

Для асинхронного исключения кроме сброса процессор может выполнить другую инструкцию между тем, когда исключение инициировано и когда процессор вводит обработчик исключений.

Привилегированное программное обеспечение может отключить исключения, которые *Таблица 17 на странице 37* показывает как наличие

конфигурируемый приоритет, см.:

Системное управление обработчиком и регистр состояния (SHCSR) на странице 219
регистры четкого включения прерывания (NVIC_ICERx) на странице 196

Для получения дополнительной информации о твердых отказах, отказах управления памятью, отказы шины, и отказы использования, видят *Раздел 2.4: обработка Отказа на странице 43.*

2.3.3 Обработчики исключений

Процессор обрабатывает использование исключений:

Служба прерывания Прерывания IRQ0 к IRQ81 являются исключениями, обработанными ISR.
Подпрограммы (ISR)

Обработчики отказа Твердый отказ, отказ управления памятью, отказ использования, отказ шины
является отказом исключения обработчиками отказа.

Системные обработчики NMI, PendSV, SVCall SysTick, и исключения отказа являются всеми системными исключениями, которые обрабатываются системными обработчиками.

2.3.4 Таблица векторов

Таблица векторов содержит значение сброса указателя вершины стека, и начальные адреса, также вызванные векторы исключения, для всех обработчиков исключений. *Рисунок 11 на странице 39* показывает порядок векторов исключения в таблице векторов. Младший значащий бит каждого вектора должен быть 1, указывая, что обработчик исключений является кодом Ползунка.

Рисунок 11. Таблица векторов

Число IRQ числа исключения	Смещение	Вектор
255	239	IRQ239
	0x03FC	
	0x004C	
18	2	IRQ2
17	1	IRQ1
16	0	IRQ0
15	-1	Systick
14	-2	PendSV
13		
12		Зарезервированный Зарезервированный для Отладки
11	-5	SVCall
10		
9		Зарезервированный
8		
7		
6	-10	Отказ использования
5	-11	Отказ шины
4	-12	Отказ управления памятью
3	-13	Твердый отказ
2	-14	NMI
1		Сброс
	0x0004	Начальное значение SP
	0x0000	

На системном сбросе таблица векторов фиксируется в адресе 0x00000000. Привилегированное программное обеспечение может записать в VTOR, чтобы переместить начальный адрес таблицы векторов к различному расположению памяти, в диапазоне 0x00000080 к 0x3FFFFFF80, смотри, [что смещение Таблицы векторов регистрируется \(VTOR\) на странице 212.](#)

2.3.5 Приоритеты исключения

Поскольку [Таблица 17 на странице 37](#) показывает, что у всех исключений есть связанный приоритет, с: Более низким приоритетным значением, указывающим на более высокий приоритет конфигурируемый приоритеты для всех исключений кроме Сброса, Трудно дать сбой, и NMI.

Если программное обеспечение не конфигурирует приоритетов, то у всех исключений с конфигурируемым приоритетом есть приоритет 0. Для получения информации о конфигурировании приоритетов исключения см.

[Системные приоритетные регистры обработчика \(SHPRx\) на странице 217](#)
[приоритетные регистры прерывания \(NVIC_IPRx\) на странице 200](#)

Конфигурируемые приоритетные значения находятся в диапазоне 0-15. Это означает, что Сброс, Трудно даёт сбой, и исключения NMI, с фиксированными отрицательными приоритетными значениями, всегда имеет более высокий приоритет чем любое другое исключение.

Например, присвоение более высокого приоритетного значения к IRQ [0] и более низкого приоритета оценивает IRQ [1] средства, что у IRQ [1] есть более высокий приоритет чем IRQ [0]. Если и IRQ [1] и IRQ [0] утверждаются, IRQ [1] обрабатывается перед IRQ [0].

Если у многократных исключений на ожидании есть тот же самый приоритет, исключение на ожидании с самым низким числом исключений имеет приоритет. Например, если и IRQ [0] и IRQ [1] находятся на рассмотрении и имеют тот же самый приоритет, то IRQ [0] обрабатывается перед IRQ [1].

Когда процессор выполняет обработчик исключений, обработчик исключений вытесняется если более высокое приоритетное исключение происходит. Если исключение происходит с тем же самым приоритетом как обрабатываемое исключение, обработчик не вытесняется, независимо от числа исключения. Однако, состояние нового прерывания изменяется на ожидание.

2.3.6 Приоритетная группировка прерывания

Чтобы увеличить управление приоритетами в системах с прерываниями, NVIC поддерживает приоритетную группировку.

Это делит каждую приоритетную запись регистра прерывания на два поля:

Верхнее поле, которое определяет *групповой приоритет*

более низкое поле, которое определяет *подприоритет* в пределах группы.

Только групповой приоритет определяет вытеснение исключений прерывания. Когда процессор выполняет обработчик исключений прерывания, другое прерывание с тем же самым групповым приоритетом, поскольку обрабатываемое прерывание не вытесняет обработчик,

Если у многократных прерываний на ожидании есть тот же самый групповой приоритет, подприоритетное поле определяет

порядок, в котором они обрабатываются. Если у многократных прерываний на ожидании есть тот же самый групповой приоритет и подприоритет, прерывание с самым низким числом IRQ обрабатывается сначала.

Для получения информации о разделении приоритетных полей прерывания в групповой приоритет и подприоритет,

см. [прерывание Приложения и сбросьте регистр команд \(AIRCR\) на странице 212](#).

2.3.7 Запись исключения и возврат

Описания обработки исключений используют следующие термины:

Вытеснение Когда процессор выполняет обработчик исключений, исключение может вытеснить обработчик исключений, если его приоритет выше чем приоритет обрабатываемого исключения. См. *Раздел 2.3.6: приоритет Прерывания, группирующийся* для получения дополнительной информации о вытеснении прерыванием.

Возврат Когда одно исключение вытесняет другое, исключения вызывают вложенные исключения. См. *запись Исключения на странице 41* информация.
Это происходит, когда обработчик исключений завершается, и:

Нет никакого исключения на ожидании с достаточным приоритетом, который будет обслуживаться завершенным обработчиком исключений который не обрабатывал последнее прибытие исключения.

Процессор выталкивает стек и восстанавливает состояние процессора к состоянию это имел прежде, чем прерывание произошло. См. [возврат Исключения на странице 43](#) для получения дополнительной информации.

Объединение в цепочку хвоста Этот механизм ускоряет обслуживание исключения. На завершении обработчик исключений, если есть исключение на ожидании, которое удовлетворяет требованиям для записи исключения, стек, появляется, пропускается, и управление передает новому обработчику исключений.

Последнее прибытие Этот механизм ускоряет вытеснение. Если более высокое приоритетное исключение происходит во время сохранения состояния для предыдущего исключения процессор переключается, чтобы обработать более высокое приоритетное исключение и инициирует векторную выборку для того исключения. На государственное сохранение не влияет последнее прибытие, потому что сохраненное состояние является тем же самым для обоих исключений. Поэтому сохранение состояния продолжается непрерывный. Процессор может принять последнее прибывающее исключение, пока первая инструкция обработчика исключений исходного исключения не вводит выполнение этапа процессора. По возврату из обработчика исключений поздно прибывающего исключения применяются нормальные объединяющие в цепочку хвост правила.

Запись исключения

Запись исключения происходит, когда есть исключение на ожидании с достаточным приоритетом и также:

Процессор находится в режиме Потока новое исключение имеет более высокий приоритет чем обрабатываемое исключение, когда новое исключение вытесняет исходное исключение.

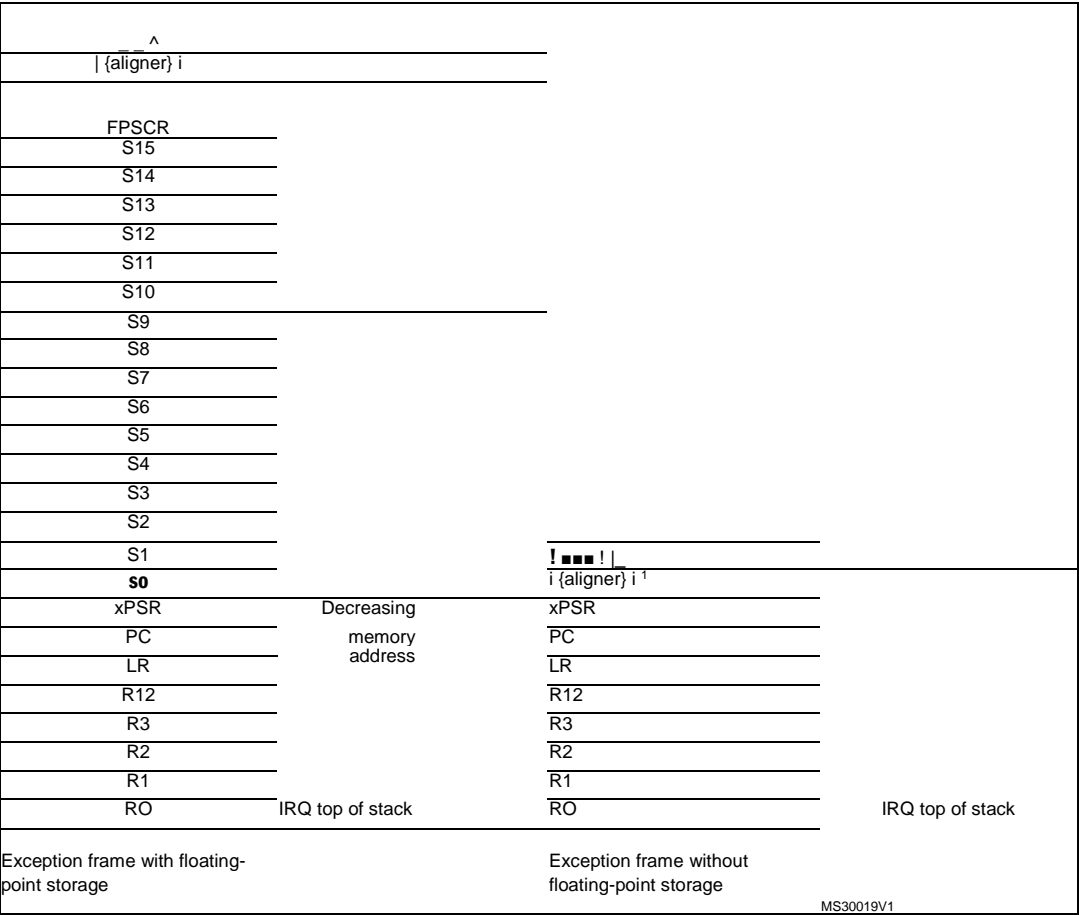
Когда одно исключение вытесняет другое, исключения вкладываются. Достаточный приоритет означает, что у исключения есть больше приоритета чем любые пределы, установленные маской регистра, см. [регистры маски Исключения на странице 22](#). Исключение с меньшим приоритетом чем это находится на рассмотрении, но не обрабатывается процессором.

Когда процессор берет исключение, если исключение не является объединенным в цепочку хвостом или последним - прибывающее исключение, процессор продвигает информацию на текущий стек. Эта работа относится как *укладка*, и структура восьми слов данных относится как *стековый фрейм*.

При использовании подпрограмм с плавающей точкой процессор Cortex-M4 автоматически складывает спроектированное состояние с плавающей точкой на записи исключения. [Рисунок 12 на странице 42](#) показывает Кору - расположение стекового фрейма M4, когда состояние с плавающей точкой сохраняется на стеке как результат прерывания или исключения. Где стековое пространство для состояния с плавающей точкой не выделяется, стековый фрейм является тем

же самым как тем из ARMv7-м. реализаций без FPU. [Рисунок 12 на странице 42](#) показывает этот стековый фрейм также.

Рисунок 12. Расположение стекового фрейма коры-M4



Сразу после укладки, указатель вершины стека указывает на самый низкий адрес в стековом фрейме. Выравниванием стекового фрейма управляют через бит STKALIGN Регистра Управления конфигурацией (CCR).

Стековый фрейм включает обратный адрес. Это - адрес следующей инструкции в

прерванной программе. Это значение восстанавливает PC при возврате исключения так, чтобы прерванная программа возобновилась.

Параллельно к работе укладки, процессор выполняет векторную выборку, которая читает начальный адрес обработчика исключений от таблицы векторов. Когда укладка полна, процессор начинает выполнять обработчик исключений. Одновременно, процессор пишет, что EXC_RETURN оценивает LR. Это указывает, какой указатель вершины стека соответствует стековому фрейму и какой режим работы был процессором, был в том, прежде, чем запись произошла.

Если никакое более высокое приоритетное исключение не происходит во время записи исключения, процессор начинает выполняться обработчик исключений и автоматически изменяет состояние соответствующего прерывания на ожидании к активному.

Если другое более высокое приоритетное исключение происходит во время записи исключения, процессор запускается выполнение обработчика исключений для этого исключения не изменяет состояние на ожидании более раннего исключения. Это - последний случай прибытия.

Возврат исключения

Возврат исключения происходит, когда процессор находится в режиме Обработчика и выполняет одно из следований инструкций, чтобы загрузить значение EXC_RETURN в PC: LDM или инструкция POP, которая загружает PC инструкция LDR с PC как место назначения инструкция BX, используя любой регистр.

EXC_RETURN является значением, загруженным в LR на записи исключения. Механизм исключения полагается на это значение, чтобы обнаружить, когда процессор завершил обработчик исключений. Самые низкие пять битов этого значения предоставляют информацию о стеке возврата и режиме процессора. [Таблица 18](#) показывает значения EXC_RETURN с описанием поведения возврата исключения.

У всех значений EXC_RETURN есть биты [31:5] набор одному. Когда это значение загружается в PC это указывает процессору, что исключение полно, и процессор иницирует соответствующую последовательность возврата исключения.

Таблица 18. Поведение возврата исключения

EXC_RETURN [31:0]	Описание
0xFFFFFFFF1	Возвратитесь к режиму Обработчика, возврат исключения использует состояние нес плавающей точкой от MSP, и выполнение использует MSP после возврата.
0xFFFFFFFF9	Возвратитесь, чтобы Распараллелить режим, возврат исключения использует состояние нес плавающей точкой от MSP, и выполнение использует MSP после возврата.
0xFFFFFFFFD	Возвратитесь, чтобы Распараллелить режим, возврат исключения использует состояние нес плавающей точкой от PSP, и выполнение использует PSP после возврата.
0xFFFFFFE1	Возвратитесь к режиму Обработчика, возврат исключения использует состояние с плавающей точкой от MSP, и выполнение использует MSP после возврата.
0xFFFFFE9	Возвратитесь, чтобы Распараллелить режим, возврат исключения использует состояние с плавающей точкой от MSP, и выполнение использует MSP после возврата.
0xFFFFFED	Возвратитесь, чтобы Распараллелить режим, возврат исключения использует состояние с плавающей точкой от PSP, и выполнение использует PSP после возврата.

2.4 Обработка отказа

Отказы являются подмножеством исключений, смотри [модель Исключения на странице 36](#).

Следующий генерируемый отказ:

Ошибка **ШИНЫ НА:**

- Выборка инструкции или загрузка таблицы векторов
- Доступ к данным

Внутренне обнаруженная ошибка, такая как неопределенная инструкция попытка выполнить инструкцию от области памяти, отмеченной как *Неисполнимая программа* (XN).

Нарушение полномочий или попытка получить доступ к неуправляемой области, вызывающей MPU отказ.

2.4.1 Типы отказа

[Таблица 19](#) показывает типы отказа, обработчик, используемый для отказа, соответствующего отказа регистр состояния, и бит регистра, который указывает, что отказ произошел. См., [что](#)

[Конфигурируемое состояние отказа регистрируется \(CFSR; UFSR+BFSR+MMFSR\) на странице](#)

[221](#) для получения дополнительной информации о регистрах состояния отказа.

Таблица 19. Отказы

Отказ	Обработчик	Разрядное имя	Регистр состояния отказа
Ошибка шины на векторном чтении	Твердый отказ	VECTTBL	<i>Твердый регистр (HFSR) на странице 225</i>
, наращиваемом к твердому отказу		ВЫЗЫВАЕТСЯ	
MPU или память по умолчанию отображают несоответствие:	MemManage	-	<i>Управление памятью дает сбой регистр адреса (MMFAR) на странице 226</i>
- на доступе инструкции		IACCVIOL ⁽¹⁾	
- на доступе к данным		DACCVIOL	
- во время укладки исключения		MSTKERR	
- во время неукладки исключения		MUNSKERR	
- во время ленивого состояния с плавающей точкой		MLSPERR	
Ошибка шины:	Отказ шины	-	<i>Отказ шины адресует регистр (BFAR) на странице 226</i>
- Во время укладки исключения		STKERR	
- Во время неукладки исключения		UNSTKERR	
- Во время упреждающей выборки инструкции		IBUSERR	
- Во время ленивого состояния с плавающей точкой		LSPERR	
Точная ошибка шины данных		PRECISERR	
Неточная ошибка шины данных		IMPRECISERR	
Попытайтесь получить доступ к сопроцессору	Отказ использования	NOCP	<i>Конфигурируемый регистр состояния отказа (CFSR; UFSR+BFSR+MMFSR) на странице 221</i>
инструкции Undefined		UNDEFINSTR	
ввести недопустимое состояние набора команд (2),		INVSTATE	
Недопустимые EXC_RETURN оценивают		INVPC	
Недопустимую невыровненную загрузку, или хранилище		UNALIGNED	
Делятся На 0		DIVBYZERO	

1. Происходит на доступе к области XN, даже если MPU отключается. 2. Попытка использовать набор команд кроме набора команд Ползунка, или возвраты к не load/store-многоадресная команда с продолжением ICI.

2.4.2 Подъем отказа и трудно дает сбой

Все исключения отказов за исключением твердого отказа имеют конфигурируемый приоритет исключения, смотри [Систему приоритетные регистры обработчика \(SHPRx\) на странице 217](#).

Программное обеспечение может отключить выполнение обработчиков для этих отказов, смотри [Системное управление обработчиком и утвердить регистр \(SHCSR\) на странице 219](#).

Обычно, приоритет исключения, вместе со значениями регистров маски исключения, определяет, вводит ли процессор обработчик отказа, и может ли обработчик отказа вытеснить другой обработчик отказа. как описано в [Разделе 2.3: модель Исключения на странице 36](#).

В некоторых ситуациях отказ с конфигурируемым приоритетом обрабатывается как твердый отказ. Это вызывает *приоритетный подъем*, и отказ описывается как *наращивающийся, чтобы трудно было дать сбой*. Подъем, чтобы трудно было дать сбой происходит когда:

Обработчик отказа вызывает тот же самый вид отказа как тот, который это обслуживает. Этот подъем трудно давать сбой происходит, потому что обработчик отказа не может вытеснить себя, потому что у него должен быть тот же самый приоритет как текущий приоритетный уровень.

Обработчик отказа вызывает отказ с тем же самым или более низким приоритетом как отказ, который это обслуживает. Это - то, потому что обработчик для нового отказа не может вытеснить в настоящий момент выполняющийся обработчик отказа.

Обработчик исключений вызывает отказ, для которого приоритет является тем же самым как или ниже чем в настоящий момент выполняющееся исключение.

Отказ происходит, и обработчик для того отказа не включается.

Если отказ шины происходит во время нажатия стека, вводя обработчик отказа шины, отказ шины не возрастает к твердому отказу. Это означает, что, если поврежденный стек вызывает отказ, обработчик отказа выполняется даже при том, что стек стремится к отказавшему обработчику. Обработчик отказа работает, но содержание стека повреждается. Только Сброс и NMI могут вытеснить фиксированный приоритет, трудно дают сбой. Твердый отказ может вытеснить любого исключение кроме Сброса, NMI, или другого твердого отказа.

2.4.3 Регистры состояния отказа и отказ адресуют регистры

Регистры состояния отказа указывают на причину отказа. Для отказов шины и памяти отказы управления, регистр адреса отказа указывает на адрес, к которому получает доступ работа, которая вызвала отказ, как показано в [Таблице 20](#).

Таблица 20. Состояние отказа и отказ адресуют регистры

Обработчик	Имя регистра состояния	Имя регистра адреса	Описание регистра
Твердый отказ	HFSR	-	Твердый регистр состояния отказа (HFSR) на странице 225
Отказ управления памятью	MMFSR	MMFAR	Управление памятью дает сбой регистр адреса (MMFAR) на странице 226
Отказ шины	BFSR	BFAR	Отказ шины адресует регистр (BFAR) на странице 226
Отказ использования	UFSR	-	Конфигурируемый регистр состояния отказа (CFSR; UFSR+BFSR+MMFSR) на странице 221

2.4.4 Тупик

Процессор вводит состояние тупика, если твердый отказ происходит, выполняя NMI или трудно обработчики отказа. Когда процессор находится в тупике, утверждают, что это не выполняет инструкций. Процессор остается в состоянии тупика до также:

Это сбрасывается NMI происходит останавливание отладчиком.

Если состояние тупика происходит от обработчика NMI, последующий NMI не заставляет процессор оставлять состояние тупика.

2.5 Управление электропитанием

Режимы ожидания процессора STM32 И CORTEX-M4 уменьшают расход энергии:

Режим ожидания останавливает часы процессора. Вся другая система и периферийные часы могут все еще работать.

Режим глубокого сна останавливает большую часть системы STM32 и периферийных часов. В продукте уровень, это соответствует или Остановке или Резервному режиму. Для получения дополнительной информации, пожалуйста, обратитесь к "Разделу" режимов питания в справочнике STM32.

Бит SLEEPDEEP SCR выбирает, какой режим ожидания используется, см. [Системное управление регистр \(SCR\) на странице 214](#). Для получения дополнительной информации о поведении режимов ожидания см. справочник продукта STM32.

Этот раздел описывает механизмы для того, чтобы ввести режим ожидания, и условия для пробуждения от режима ожидания.

2.5.1 Ввод режима ожидания

Этот раздел описывает программное обеспечение механизмов, может использовать, чтобы поместить процессор в режим сна.

Система может генерировать побочные события пробуждения, например работа отладки просыпается процессору. Поэтому программное обеспечение должно быть в состоянии отложить процессор в режим ожидания после такого события. У программы мог бы быть неактивный цикл, чтобы отложить процессор к режиму ожидания.

Ожидайте прерывания

Ожидание инструкции прерывания, WFI, вызывает непосредственную запись в режим ожидания (если условие пробуждения является истиной, см. [Пробуждение от WFI или сна на выходе на странице 47](#)). Когда процессор выполняет инструкцию WFI, он прекращает выполнять инструкции и вводит режим ожидания. См. [WFI на странице 177](#) для получения дополнительной информации.

Ожидайте события

Ожидание инструкции события, WFE, вызывает запись в режим ожидания в зависимости от значения одноразрядный регистра события. Когда процессор выполняет инструкцию WFE, он проверяет значение регистра события:

0: процессор прекращает выполнять инструкции и вводит режим ожидания

1: процессор очищает регистр к 0 и продолжает выполнять инструкции без ввод режима ожидания.

См. [WFE на странице 176](#) для получения дополнительной информации. Если регистр события 1, указывает, что процессор не должен ввести режим ожидания в выполнение инструкции WFE. Как правило, это , потому что сигнал внешнего события утверждается, или процессор в системе выполнил инструкцию SEV, см. [SEV на странице 175](#). Программное обеспечение не может получить доступ к этому регистру непосредственно.

Сон на выходе

Если бит SLEEPONEXIT SCR устанавливается в 1, когда процессора завершает выполнение из обработчика исключений это возвращается, чтобы Распараллелить режим и сразу вводит режим ожидания. Используйте этот механизм в приложениях, которые только требуют, чтобы процессор работал, когда исключение происходит.

2.5.2

Пробуждение от режима ожидания

Условия для процессора к пробуждению зависят от механизма, которые заставляют входить режим ожидания.

Пробуждение от WFI или сна на выходе

Обычно, процессор просыпается только, когда он обнаруживает исключение с достаточным приоритетом к запись исключения причины. Некоторым встроенным системам, возможно, придется выполнить системные задачи восстановления после пробуждения процессора, и прежде, чем это выполнит обработчик прерываний. Достигнуть этого набора бит PRIMASK к 1 и бит FAULTMASK к 0. Если прерывание прибывает, который включается и имеет более высокий приоритет чем текущий приоритет исключения, процессор просыпается, но не выполняет обработчика прерываний, пока процессор не обнуляет PRIMASK. Для получения дополнительной информации о PRIMASK и FAULTMASK см. [регистры маски Исключения на странице 22](#).

Пробуждение от WFE

Процессор просыпается если:

Он обнаруживает исключение с достаточным приоритетом вызывает запись исключения обнаруживает сигнал внешнего события, см. [Раздел 2.5.3: ввод Внешнего события / расширенный прерывание и ввод событий](#)

В многопроцессорной системе другой процессор в системе выполняет SEV инструкцию.

Кроме того, если бит SEVONPEND в SCR устанавливается в 1, какие-либо новые триггеры прерывания на ожидания события и просыпается процессор, даже если прерывание отключается или имеет

недостаточный приоритет вызова записи исключения. Для получения дополнительной информации о SCR см. [Системный регистр команд \(SCR\) на странице 214](#).

2.5.3 Ввод внешнего события / расширенное прерывание и ввод событий

Процессор обеспечивает входной сигнал внешнего события. Для серии STM32F4xxx этот сигнал может быть сгенерирован до 16 внешних вводов строки, PVD, аварийным сигналом RTC или событием пробуждения USB, сконфигурированным через внешний контроллер прерывания/события (EXTI).

Для серии STM32F3xxx этот сигнал может быть сгенерирован расширенным прерыванием/событием контроллер (EXTI), который управляет до 29 внешних и внутренних асинхронных прерываний и событий. Этот сигнал может пробуждать процессор от WFE, или устанавливать внутренний регистр события WFE

в

один, чтобы указать, что процессор не должен ввести режим ожидания в более позднюю инструкцию WFE, см., [Ожидают события на странице 47](#).

2.5.4 Управление электропитанием, программирующие подсказки

ISO/IEC C не может непосредственно генерировать инструкции WFI И WFE. CMSIS обеспечивает следующие функции для этих инструкций:

```
void WFE(void) // Wait for Event
void WFI(void) // Wait for Interrupt
```

3 Набор команд Кору-М4 STM32

Эта глава является ссылочным материалом для описания набора команд Кору-М4 в Руководстве пользователя. Следующие разделы дают общую информацию:

[Раздел 3.1: сводка Набора команд на Разделе страницы 49](#)

[3.2: встроенные функции CMSIS на Разделе страницы 57](#)

[3.3: Об описаниях инструкции на странице 59](#) Каждый из следующих разделов описывает функциональную группу инструкций Cortex-M4. Вместе они описывают все инструкции, поддерживаемые процессором Cortex-M4:

[Раздел 3.4: инструкции доступа Памяти по Разделу страницы 68](#)

[3.5: Общие инструкции обработки данных по Разделу страницы 80](#)

[3.6: Умножители и делители инструкции по Разделу страницы 108](#)

[3.7: инструкции Saturating по Разделу страницы 124](#)

[3.8: Упаковка и распаковка инструкций по Разделу страницы 132](#)

[3.9: инструкции Bitfield по Разделу страницы 136](#)

[3.10: инструкции с плавающей точкой по Разделу страницы 148](#)

[3.11: инструкции Miscellaneous на странице 170](#)

3.1 Сводка набора команд

Процессор реализует версию набора команд ползунка. [Таблица 21](#) перечисляет

поддерживаемые инструкции.

В *Таблице 21*:

Угловые скобки, <>, включают альтернативные формы операнда

фигурные скобки, {}, включают дополнительные операнды столбец операндов не исчерпывающий

Op2 является гибким вторым операндом, который может быть или регистром или константой

большинство инструкций может использовать дополнительный суффикс кода условия

Для получения дополнительной информации по инструкциям и операндам, см. описания инструкции.

Таблица 21. Инструкции коры-M4

Мнемосхема	Операнды	Краткое описание	Флаги	Страница
ADC, ADCS	{Rd,} Rn, Op2	Добавьте с переносом	N,Z,C,V	3.5.1 на странице 82
ADD, ADDS	{Rd,} Rn, Op2	Добавить	N,Z,C,V	3.5.1 на странице 82
ADD, ADDW	{Rd,} Rn, #imm12	Добавить	N,Z,C,V	3.5.1 на странице 82
ADR	Rd, label	Загрузите относительный PC адрес	—	3.4.1 на странице 69

Таблица 21. Инструкции коры-M4 (продолжение)

Мнемосхема	Операнды	Краткое описание	Флаги	Страница
AND, ANDS	{Rd,} Rn, Op2	Логичный И	N,Z,C	3.5.2 на странице 84
ASR, ASRS	Rd, Rm, <Rs n>	Право арифметического сдвига	N,Z,C	3.5.3 на странице 85
B	label	Ответвление	—	3.9.5 на странице 141
BFC	Rd, #lsb, #width	Четкое битовое поле	—	3.9.1 на странице 137
BFI	Rd, Rn, #lsb, #width	Битовое поле вставляет	—	3.9.1 на странице 137
BIC, BICS	{Rd,} Rn, Op2	Чистый бит	N,Z,C	3.5.2 на странице 84
BKPT	#imm	Точка останова	—	3.11.1 на странице 170
BL	label	Ответвление со ссылкой	—	3.9.5 на странице 141
BLX	Rm	Ответвление, косвенное со ссылкой	—	3.9.5 на странице 141
BX	Rm	Косвенное ответвление	—	3.9.5 на странице 141
CBNZ	Rn, label	Сравнитесь и перейдите если не ноль	—	3.9.6 на странице 143
CBZ	Rn, label	Сравнитесь и перейдите если ноль	—	3.9.6 на странице 143
CLREX	—	Очистите монопольный	—	3.4.9 на странице 79
CLZ	Rd, Rm	Подсчитайте начальные нули	—	3.5.4 на странице 86
CMN	Rn, Op2	Сравнитесь отрицательный	N,Z,C,V	3.5.5 на странице 87
CMP	Rn, Op2	Сравниться	N,Z,C,V	3.5.5 на странице 87
CPSID	iflags	Состояние процессора изменения, отключите прерывания	—	3.11.2 на странице 171
CPSIE	iflags	Состояние процессора изменения, включите прерываниям	—	3.11.2 на странице 171
DMB	—	Барьер памяти данных	—	3.11.4 на странице 172
DSB	—	Барьер синхронизации данных	—	3.11.4 на странице 172
EOR, EORS	{Rd,} Rn, Op2	Монопольный ИЛИ	N,Z,C	3.5.2 на странице 84
ISB	—	Барьер синхронизации инструкции	—	3.11.5 на странице 173
IT	—	Есл -тогда блок условия	—	3.9.7 на странице 144
LDM	Rn(!), reglist	Загрузите многократные регистры, инкремент после	—	3.4.6 на странице 75
LDMDB, LDMEA	Rn(!), reglist	Загрузите многократные регистры, декремент прежде	—	3.4.6 на странице 75
LDMFD, LDMIA	Rn(!), reglist	Загрузите многократные регистры, инкремент после	—	3.4.6 на странице 75
LDR	Rt, [Rn, #offset]	Регистр загрузки со словом	—	3.4 на странице 68
LDRB, LDRBT	Rt, [Rn, #offset]	Регистр загрузки с байтом	—	3.4 на странице 68
LDRD	Rt, Rt2, [Rn, #offset]	Регистр загрузки с двумя байтами	—	3.4.2 на странице 70
LDREX	Rt, [Rn, #offset]	Монопольный регистр загрузки	—	3.4.8 на странице 78

Таблица 21. Инструкции коры-M4 (продолжение)

Мнемосхема	Операнды	Краткое описание	Флаги	Страница
LDREXB	Rt, [Rn]	Регистр загрузки, монопольный с байтом	—	3.4.8 на странице 78
LDREXH	Rt, [Rn]	Регистр загрузки, монопольный с полусловом	—	3.4.8 на странице 78
LDRH, LDRHT	Rt, [Rn, #offset]	Регистр загрузки с полусловом	—	3.4 на странице 68
LDRSB, LDRSBT	Rt, [Rn, #offset]	Регистр загрузки с байтом со знаком	—	3.4 на странице 68
LDRSH, LDRSHT	Rt, [Rn, #offset]	Регистр загрузки с полусловом со знаком	—	3.4 на странице 68
LDRT	Rt, [Rn, #offset]	Регистр загрузки со словом	—	3.4 на странице 68
LSL, LSLS	Rd, Rm, <Rs n>	Логический сдвиг уехал	N,Z,C	3.5.3 на странице 85
LSR, LSRS	Rd, Rm, <Rs n>	Право логического сдвига	N,Z,C	3.5.3 на странице 85
MLA	Rd, Rn, Rm, Ra	умножители накапливаются, результат на 32-биты	—	3.6.1 на странице 109
MLS	Rd, Rn, Rm, Ra	Умножьте и вычитите, 32-разрядный результат	—	3.6.1 на странице 109
MOV, MOVS	Rd, Op2	Переместиться	N,Z,C	3.5.6 на странице 88
MOVT	Rd, #imm16	Переместите вершину	—	3.5.7 на странице 90
MOVW, MOV	Rd, #imm16	Переместите 16-разрядную константу	N,Z,C	3.5.6 на странице 88
MRS	Rd, spec_reg	Переместитесь от специального регистра до общего регистра	—	3.11.6 на странице 173
MSR	spec_reg, Rm	Переместитесь от общего регистра до специального	N,Z,C,V	3.11.7 на странице 174
MUL, MULS	{Rd,} Rn, Rm	Умножьтесь, 32-разрядный результат	N,Z	3.6.1 на странице 109
MVN, MVNS	Rd, Op2	Переместитесь НЕТ	N,Z,C	3.5.6 на странице 88
NOP	—	Никакая работа	—	3.11.8 на странице 175
ORN, ORNS	{Rd,} Rn, Op2	Логичный ИЛИ НЕТ	N,Z,C	3.5.2 на странице 84
ORR, ORRS	{Rd,} Rn, Op2	Логичный ИЛИ	N,Z,C	3.5.2 на странице 84
PKHTB, PKHBT	{Rd,} Rn, Rm, Op2	Упакуйте Полуслово		3.8.1 на странице 133
POP	reglist	Вытолкайте регистры от стека	—	3.4.7 на странице 77
PUSH	reglist	Продвиньте регистры на стек	—	3.4.7 на странице 77
QADD	{Rd,} Rn, Rm	Двойное насыщение и добавляет		3.7.3 на странице 127
QADD16	{Rd,} Rn, Rm	Насыщение добавляет 16		3.7.3 на странице 127
QADD8	{Rd,} Rn, Rm	Насыщение добавляет 8		3.7.3 на странице 127
QASX	{Rd,} Rn, Rm	Насыщение добавляет и вычитает с обменом		3.7.4 на странице 128

Таблица 21. Инструкции коры-M4 (продолжение)

Мнемосхема	Операнды	Краткое описание	Флаги	Страница
QDADD	{Rd,} Rn, Rm	Насыщение добавляет		3.7.5 на странице 129
QDSUB	{Rd,} Rn, Rm	Двойное насыщение и вычитает		3.7.5 на странице 129
QSAX	{Rd,} Rn, Rm	Насыщение вычитает и добавляет с обменом		3.7.4 на странице 128
QSUB	{Rd,} Rn, Rm	Насыщение вычитает		3.7.3 на странице 127
QSUB16	{Rd,} Rn, Rm	Насыщение вычитает 16		3.7.4 на странице 128
QSUB8	{Rd,} Rn, Rm	Насыщение вычитает 8		3.7.4 на странице 128
RBIT	Rd, Rn	Обратные биты	—	3.7.4 на странице 128
REV	Rd, Rn	Обратный порядок байтов одним словом	—	3.5.8 на странице 91
REV16	Rd, Rn	Обратный порядок байтов в каждом полуслове	—	3.5.8 на странице 91
REVSH	Rd, Rn	Обратный порядок байтов в нижнем полуслове и знаке расширяется	—	3.5.8 на странице 91
ROR, RORS	Rd, Rm, <Rs >#n>	Поверните право	N,Z,C	3.5.3 на странице 85
RRX, RRXS	Rd, Rm	Вращайтесь прямо с, расширяются	N,Z,C	3.5.3 на странице 85
RSB, RSBS	{Rd,} Rn, Op2	Реверс вычитает	N,Z,C,V	3.5.1 on page 82
SADD16	{Rd,} Rn, Rm	Подписанный добавляют 16		3.5.9 на странице 92
SADD8	{Rd,} Rn, Rm	Подписанный добавляют 8		3.5.9 на странице 92
SASX	{Rd,} Rn, Rm	Подписанный добавляют и вычитают с обменом		3.5.14 на странице 97
SBC, SBCS	{Rd,} Rn, Op2	Вычитите с переносом	N,Z,C,V	3.5.1 на странице 82
SBFX	Rd, Rn, #sb, #width	Извлечение битового поля со знаком	—	3.9.2 на странице 138
SDIV	{Rd,} Rn, Rm	Подписанный делятся	—	3.6.3 на странице 111
SEV	—	Отправьте событие	—	3.11.9 на странице 175
SHADD16	{Rd,} Rn, Rm	Деление на два со знаком добавляет 16	—	3.5.10 на странице 93
SHADD8	{Rd,} Rn, Rm	Деление на два со знаком добавляет 8	—	3.5.10 на странице 93
SHASX	{Rd,} Rn, Rm	Деление на два со знаком добавляет и вычитает с обменом	—	3.5.11 на странице 94
SHSAX	{Rd,} Rn, Rm	Деление на два со знаком вычитает и добавляет с обменом	—	3.5.11 на странице 94
SHSUB16	{Rd,} Rn, Rm	Деление на два со знаком вычитает 16	—	3.5.12 на странице 95
SHSUB8	{Rd,} Rn, Rm	Деление на два со знаком вычитает 8	—	3.5.12 на странице 95
SMLABB, SMLABT, SMLATB, SMLATT	Rd, Rn, Rm, Ra	Подписанный умножаются, накапливаются длинный (полуслова)	Q	3.6.3 на странице 111
SMLAD, SMLADX	Rd, Rn, Rm, Ra	Подписанный умножаются, накапливаются двойной	Q	3.6.4 на странице 113

Таблица 21. Инструкции коры-M4 (продолжение)

Мнемосхема	Операнды	Краткое описание	Флаги	Страница
SMLAL	RdLo, RdHi, Rn, Rm	Подписанный умножаются с, накапливаются (32 x 32 + 64), результат на 64-биты	—	3.6.2 на странице 110
SMLALBB, SMLALBT, SMLALTB, SMLALTT	RdLo, RdHi, Rn, Rm	Подписанный умножаются, накапливаются долго, полуслова	—	3.6.5 на странице 114
SMLALD, SMLALDX	RdLo, RdHi, Rn, Rm	Подписанный умножаются, накапливаются долго двойной	—	3.6.5 на странице 114
SMLAWB, SMLAWT	Rd, Rn, Rm, Ra	Подписанный умножаются, накапливаются, слово полусловом	Q	3.6.3 на странице 111
SMLS	Rd, Rn, Rm, Ra	Подписанный умножаются, вычитают двойной	Q	3.6.6 на странице 116
SMLS	RdLo, RdHi, Rn, Rm	Подписанный умножаются, вычитают долго двойной	—	3.6.6 на странице 116
SMMLA	Rd, Rn, Rm, Ra	Старшее значащее слово со знаком умножается,	—	3.6.7 на странице 118
SMMLS, SMMLR	Rd, Rn, Rm, Ra	Старшее значащее слово со знаком умножается, вычитают	—	3.6.7 на странице 118
SMMUL, SMMULR	{Rd,} Rn, Rm	Старшее значащее слово со знаком умножается	—	3.6.8 на странице 119
SMUAD	{Rd,} Rn, Rm	Со знаком двойной умножаются, добавляют	Q	3.6.9 на странице 120
SMULBB, SMULBT SMULTB, SMULTT	{Rd,} Rn, Rm	Подписанный умножаются (полуслова)	—	3.6.10 на странице 121
SMULL	RdLo, RdHi, Rn, Rm	Подписанный умножаются (32 x 32), результат на 64-биты	—	3.6.2 на странице 110
SSAT	Rd, #n, Rm {,shift #s}	Подписанный насыщают	Q	3.7.1 на странице 125
SSAT16	Rd, #n, Rm	Подписанный насыщают 16	Q	3.7.2 на странице 126
SSAX	{Rd,} Rn, Rm	Подписанный вычитают и добавляют с обменом	GE	3.5.14 на странице 97
SSUB16	{Rd,} Rn, Rm	Подписанный вычитают 16	—	3.5.13 на странице 96
SSUB8	{Rd,} Rn, Rm	Подписанный вычитают 8	—	3.5.13 на странице 96
STM	Rn{!}, reglist	Сохраните многократные регистры, инкремент после	—	3.4.6 на странице 75
STMDB, STMEA	Rn{!}, reglist	Сохраните многократные регистры, декремент прежде	—	3.4.6 на странице 75
STMFD, STMIA	Rn{!}, reglist	Сохраните многократные регистры, инкремент после	—	3.4.6 на странице 75
STR	Rt, [Rn, #offset]	Слово регистра хранилища	—	3.4 на странице 68
STRB, STRBT	Rt, [Rn, #offset]	Байт регистра хранилища	—	3.4 на странице 68

Таблица 21. Инструкции коры-M4 (продолжение)

Мнемосхема	Операнды	Краткое описание	Флаги	Страница
STRD	Rt, Rt2, [Rn, #offset]	Регистр хранилища два слова	—	3.4.2 на странице 70
STREX	Rd, Rt, [Rn, #offset]	Монопольный регистр хранилища	—	3.4.8 на странице 78
STREXB	Rd, Rt, [Rn]	Регистр хранилища монопольный байт	—	3.4.8 на странице 78
STREXH	Rd, Rt, [Rn]	Регистр хранилища монопольное полуслово	—	3.4.8 на странице 78
STRH, STRHT	Rt, [Rn, #offset]	Полуслово регистра хранилища	—	3.4 на странице 68
STRT	Rt, [Rn, #offset]	Слово регистра хранилища	—	3.4 на странице 68
SUB, SUBS	{Rd,} Rn, Op2	Вычесть	N,Z,C,V	3.5.1 на странице 82
SUB, SUBW	{Rd,} Rn, #imm12	Вычесть	N,Z,C,V	3.5.1 на странице 82
SVC	#imm	Вызов супервизора	—	3.11.10 на странице 176
SXTAB	{Rd,} Rn, Rm,{,ROR #}	Расширитель на 8 битов на 32 и добавьте	—	3.8.3 на странице 135
SXTAB16	{Rd,} Rn, Rm,{,ROR #}	Двойной расширяются на 8 битов на 16 и добавляют	—	3.8.3 на странице 135
SXTAH	{Rd,} Rn, Rm,{,ROR #}	Расширитель на 16 битов на 32 и добавьте	—	3.8.3 на странице 135
SXTB16	{Rd,} Rm {,ROR #n}	Подписанный расширяют байт 16	—	3.8.2 на странице 134
SXTB	{Rd,} Rm {,ROR #n}	Знак расширяет байт	—	3.9.3 на странице 139
SXTH	{Rd,} Rm {,ROR #n}	Знак расширяет полуслово	—	3.9.3 на странице 139
TBB	[Rn, Rm]	Табличный байт ответвления	—	3.9.8 на странице 146
TBH	[Rn, Rm, LSL #1]	Табличное полуслово ответвления	—	3.9.8 на странице 146
TEQ	Rn, Op2	Тестовая эквивалентность	N,Z,C	3.5.9 на странице 92
TST	Rn, Op2	Тест	N,Z,C	3.5.9 на странице 92
UADD16	{Rd,} Rn, Rm	Без знака добавляют 16	GE	3.5.16 на странице 99
UADD8	{Rd,} Rn, Rm	Без знака добавляют 8	GE	3.5.16 на странице 99
USAX	{Rd,} Rn, Rm	Без знака вычитают и добавляют с обменом	GE	3.5.17 на странице 100
UHADD16	{Rd,} Rn, Rm	Деление на два без знака добавляет 16	—	3.5.18 на странице 101
UHADD8	{Rd,} Rn, Rm	Деление на два без знака добавляет 8	—	3.5.18 на странице 101
UHASX	{Rd,} Rn, Rm	Деление на два без знака добавляет и вычитает с обменом	—	3.5.19 на странице 102
UHSAX	{Rd,} Rn, Rm	Деление на два без знака вычитает и добавляет с обменом	—	3.5.19 на странице 102
UHSUB16	{Rd,} Rn, Rm	Деление на два без знака вычитает 16	—	3.5.20 на странице 103
UHSUB8	{Rd,} Rn, Rm	Деление на два без знака вычитает 8	—	3.5.20 на странице 103
UBFX	Rd, Rn, #lsb, #width	Извлечение битового поля без знака	—	3.9.2 на странице 138

Таблица 21. Инструкции коры-M4 (продолжение)

Мнемосхема	Операнды	Краткое описание	Флаги	Страница
UDIV	{Rd,} Rn, Rm	Деление без знака	—	3.6.3 на странице 111
UMAAL	RdLo, RdHi, Rn, Rm	Без знака умножаются, накапливаются, накапливаются долго (32 x 32 + 32 +32), 64-разрядный результат	—	3.6.2 на странице 110
UMLAL	RdLo, RdHi, Rn, Rm	Без знака умножаются с, накапливаются (32 x 32 + 64), результат на 64-биты	—	3.6.2 на странице 110
UMULL	RdLo, RdHi, Rn, Rm	Без знака умножаются (32 x 32), 64-разрядный результат	—	3.6.2 на странице 110
UQADD16	{Rd,} Rn, Rm	Насыщение без знака добавляет 16	—	3.7.7 на странице 131
UQADD8	{Rd,} Rn, Rm	Насыщение без знака добавляет 8	—	3.7.7 на странице 131
UQASX	{Rd,} Rn, Rm	Насыщение без знака добавляет и вычитает с обменом	—	3.7.6 на странице 130
UQSAX	{Rd,} Rn, Rm	Насыщение без знака вычитает и добавляет с обменом	—	3.7.6 на странице 130
UQSUB16	{Rd,} Rn, Rm	Насыщение без знака вычитает 16	—	3.7.7 на странице 131
UQSUB8	{Rd,} Rn, Rm	Насыщение без знака вычитает 8	—	3.7.7 на странице 131
USAD8	{Rd,} Rn, Rm	Сумма без знака абсолютных разностей	—	3.5.22 на странице 105
USADA8	{Rd,} Rn, Rm, Ra	Сумма без знака абсолютных разностей и накапливается	—	3.5.23 на странице 106
USAT	Rd, #n, Rm {,shift #s}	Без знака насыщают	Q	3.7.1 на странице 125
USAT16	Rd, #n, Rm	Без знака насыщают 16	Q	3.7.2 на странице 126
UASX	{Rd,} Rn, Rm	Без знака добавляю и вычитают с обменом	GE	3.5.17 на странице 100
USUB16	{Rd,} Rn, Rm	Без знака вычитают 16	GE	3.5.24 на странице 107
USUB8	{Rd,} Rn, Rm	Без знака вычитают 8	GE	3.5.24 на странице 107
UXTAB	{Rd,} Rn, Rm,{,ROR #}	Вращайтесь, расширитесь на 8 битов на 32 и добавьте	—	3.8.3 на странице 135
UXTAB16	{Rd,} Rn, Rm,{,ROR #}	Вращайтесь, двойной расширятся на 8 битов на 16 и добавляю	—	3.8.3 на странице 135
UXTAH	{Rd,} Rn, Rm,{,ROR #}	Вращайтесь, без знака расширяют и добавляю полуслово	—	3.8.3 на странице 135
UXTB	{Rd,} Rm {,ROR #n}	Ноль расширяет байт	—	3.8.2 на странице 134
UXTB16	{Rd,} Rm {,ROR #n}	Без знака расширяют байт 16	—	3.8.2 на странице 134
UXTH	{Rd,} Rm {,ROR #n}	Ноль расширяет полуслово	—	3.8.2 на странице 134
VABS.F32	Sd, Sm	Абсолют с плавающей точкой	—	3.10.1 на странице 149
VADD.F32	{Sd,} Sn, Sm	С плавающей точкой добавляю	—	3.10.2 на странице 150

Таблица 21. Инструкции коры-M4 (продолжение)

Мнемосхема	Операнды	Краткое описание	Флаги	Страница
VCMPF32	Sd, <Sm #0.0>	Сравните два регистра с плавающей точкой, или один регистр с плавающей точкой и ноль	FPSCR	3.10.3 на странице 150
VCMPE.F32	Sd, <Sm #0.0>	Сравните два регистра с плавающей точкой, или один регистр с плавающей точкой и ноль с Недопустимой проверкой Работы	FPSCR	3.10.3 на странице 150
VCVT.S32.F32	Sd, Sm	Преобразуйте между с плавающей точкой и целочисленным	—	3.10.4 на странице 151
VCVT.S16.F32	Sd, Sd, #fbits	Преобразуйте между с плавающей точкой и фиксированной точкой	—	3.10.4 на странице 151
VCVTR.S32.F32	Sd, Sm	Преобразуйте между с плавающей точкой и целочисленным с округлением	—	3.10.4 на странице 151
VCVT<B H>.F32.F16	Sd, Sm	Преобразовывает значение полуточности в одинарную точность	—	3.10.5 на странице 152
VCVTT<B T>.F32.F16	Sd, Sm	Преобразовывает регистр одинарной точности в полуточность	—	3.10.6 на странице 153
VDIV.F32	{Sd,} Sn, Sm	Деление с плавающей точкой	—	3.10.7 на странице 154
VFMA.F32	{Sd,} Sn, Sm	С плавающей точкой сплавленный умножаются, накапливаются	—	3.10.8 на странице 154
VFNMA.F32	{Sd,} Sn, Sm	С плавающей точкой сплавленный инвертируют, умножаются, накапливаются	—	3.10.9 на странице 155
VFMS.F32	{Sd,} Sn, Sm	С плавающей точкой сплавленный умножаются, вычитают	—	3.10.8 на странице 154
VFNMS.F32	{Sd,} Sn, Sm	С плавающей точкой сплавленный инвертируют, умножаются, вычитают	—	3.10.9 на странице 155
VLDM.F<32 64>	Rn{!}, list	Загрузите многократные регистры расширения	—	3.10.10 на странице 156
VLDR.F<32 64>	<Dd Sd>, [Rn]	Загрузите регистр расширения из памяти	—	3.10.11 на странице 157
VLMA.F32	{Sd,} Sn, Sm	С плавающей точкой умножаются, накапливаются	—	3.10.12 на странице 158
VLMS.F32	{Sd,} Sn, Sm	С плавающей точкой умножаются, вычитают	—	3.10.12 на странице 158
VMOV.F32	Sd, #imm	Непосредственное перемещение с плавающей точкой	—	3.10.13 на странице 158
VMOV	Sd, Sm	Регистр перемещения с плавающей точкой	—	3.10.14 на странице 159
VMOV	Sn, Rt	Ядро ARM копии регистрируется к одинарной точности	—	3.10.18 на странице 162
VMOV	Sm, Sm1, Rt, Rt2	Скопируйте 2 регистра ядра ARM в 2 одинарной точности	—	3.10.17 на странице 161

Таблица 21. Инструкции коры-M4 (продолжение)

Мнемосхема	Операнды	Краткое описание	Флаги	Страница
VMOV	Dd[x], Rt	Ядро ARM копии регистрируется к скаляру	—	3.10.15 на странице 159
VMOV	Rt, Dn[x]	Скопируйте скаляр в регистр ядра ARM	—	3.10.16 на странице 160
VMRS	Rt, FPSCR	Переместите FPSCR в регистр ядра ARM или APSR	N,Z,C,V	3.10.19 на странице 162
VMSR	FPSCR, Rt	Переместитесь в FPSCR от регистра Ядра ARM	FPSCR	3.10.20 на странице 163
VMUL.F32	{Sd,} Sn, Sm	С плавающей точкой умножаются	—	3.10.21 на странице 163
VNEG.F32	Sd, Sm	С плавающей точкой инвертируют	—	3.10.22 на странице 164
VNMLA.F32	Sd, Sn, Sm	С плавающей точкой умножаются и добавляют	—	3.10.23 на странице 165
VNMLS.F32	Sd, Sn, Sm	С плавающей точкой умножают и вычитают	—	3.10.23 на странице 165
VNMUL	{Sd,} Sn, Sm	С плавающей точкой умножаются	—	3.10.23 на странице 165
VPOP	list	Вытолкайте регистры расширения	—	3.10.24 на странице 166
VPUSH	list	Продвиньте регистры расширения	—	3.10.25 на странице 166
VSQRT.F32	Sd, Sm	Вычисляет квадратный корень с плавающей точкой	—	3.10.26 на странице 167
VSTM	Rn(!), list	Многократное хранилище регистра с плавающей точкой	—	3.10.27 на странице 167
WFE	—	Ожидайте события	—	3.11.11 на странице 176
WFI	—	Ожидайте прерывания	—	3.11.12 на странице 177

3.2 Встроенные функции CMSIS

Код ISO/IEC C не может непосредственно получить доступ к некоторым инструкциям Cortex-M4. Этот раздел описывает встроенные функции, которые могут генерировать эти инструкции, при условии CMSIS и это может бы быть обеспечено компилятором C. Если компилятор C не поддерживает соответствующую встроенную функцию, Вам, возможно, придется использовать встроенный ассемблер, чтобы получить доступ к некоторым инструкциям. CMSIS обеспечивает встроенные функции, перечисленные в [Таблице 22](#), чтобы генерировать инструкции ANSI не может непосредственно получить доступ.