

3.10.23 VNMLA, VNMLS, VNMUL

С плавающей точкой умножаются с отрицанием, сопровождаемым, добавлением или вычитанием.

Синтаксис

VNMLA {cond}.F32 Sd, Sn, Cm

VNMLS {cond}.F32 Sd, Sn, Cm

VNMUL {cond}.F32 {Sd}, Sn, Cm

где:

'cond' является дополнительным кодом условия, см. [Условное выполнение на странице 64](#)

'Sd' является местом назначения значения с плавающей точкой

'Sn, Cm' является операндом значения с плавающей точкой.

Работа инструкция VNMLA:

1. Умножает два значения регистра с плавающей точкой.
2. Добавляет отрицание значения с плавающей точкой в целевом регистре к отрицанию из продукта.
3. Записывает результат обратно к целевому регистру.

Инструкция VNMLS:

1. Умножает два значения регистра с плавающей точкой.
2. Добавляет отрицание значения с плавающей точкой в целевом регистре к продукту.
3. записывает результат обратно к целевому регистру.

Инструкция VNMUL:

1. Умножает вместе два значения регистра с плавающей точкой.
2. Пишет отрицание результата к целевому регистру.

Ограничения

Не являются никакими ограничениями.

Флаги условия

Эти инструкции не изменяют флаги.

3.10.24 VPOP

Население регистра расширения с плавающей точкой

Синтаксис

VPOP {cond} {.size} список

где:

'cond' является дополнительным кодом условия, см. [Условное выполнение на странице 64](#).

'Размер' является спецификатором размера дополнительных данных. Если существующий, равен размеру в битах, 32 или 64, регистров в списке.

'Список' является списком регистров расширения, который будет загружен, как список последовательно пронумерованных двойных слов или однословных регистров, разделенных запятыми и окруженными скобками.

Работа

Эта инструкция загружает многократные последовательные регистры расширения из стека.

Ограничения

Список должен содержать, по крайней мере, один регистр, и не больше чем шестнадцать регистров.

Флаги условия

Эти инструкции не изменяют флаги.

3.10.25 VPUSH

Нажатие регистра расширения с плавающей точкой.

Синтаксис VPUSH {cond} {.size} перечисляет где:

'cond' является дополнительным кодом условия, см. [Условное выполнение на странице 64](#).

'размер' является спецификатором размера дополнительных данных. Если существующий, равен размеру в битах, 32 или 64, регистров в списке.

'Список' является списком регистров расширения, который будет сохранен, как список последовательно пронумерованных двойных слов или однословных регистров, разделенных запятыми и окруженных скобками.

Работа

Эта инструкция хранит многократные последовательные регистры расширения к стеку.

Ограничения

Являются списком, должны содержать, по крайней мере, один регистр, и не больше чем шестнадцать.

Флаги условия

Эти инструкции не изменяют флаги.

3.10.26 VSQRT

Квадратный корень с плавающей точкой.

Синтаксис

VSQRT {cond}.F32 Sd, Cm

где:

'cond' является дополнительным кодом условия, см. [Условное выполнение на странице 64](#)

'Sd' является местом назначения значение с плавающей точкой

'Cm' является операндом значения с плавающей точкой.

Работа

Эта инструкция:

1. Вычисляет квадратный корень значения в регистре с плавающей точкой.
2. Пишет результат в другой регистр с плавающей точкой.

Ограничения

Не являются никакими ограничениями.

Флаги условия

Эти инструкции не изменяют флаги.

3.10.27 VSTM

Многократное Хранилище с плавающей точкой.

Синтаксис

VSTM {режим} {cond} {.size} Rn{!}, список

где:

'Режим' является режимом адресации:

Инкремент IA После. Последовательные адреса запускаются в адресе, определенном в Rn. Это - значение по умолчанию и может быть опущено. Декремент DB Прежде. Последовательный конец адресов как раз перед адресом определяется в Rn.

'cond' является дополнительным кодом условия, см. [Условное выполнение на странице 64](#).

'размер' является спецификатором размера дополнительных данных. Если существующий, равен размеру в битах, 32 или 64, регистров в списке.

'Rn' является индексным регистром. SP может использоваться.

'!' функция, которая заставляет инструкцию записывать измененное значение обратно к Rn. Требуемый, если режим == DB.

'Список' является списком регистров расширения, которые будут сохранены, как список последовательно пронумерованного двойного слова или однословного регистра, разделенного запятыми и окруженный скобками.

Работа

Эта инструкция хранит многократные регистры расширения к последовательным расположениям памяти, используя базовый адрес от регистра ядра ARM.

Ограничения:

Список должен содержать по крайней мере один регистр.

ЕСЛИ содержит регистры двойного слова, то не должно содержать больше чем 16 регистров.

ИСПОЛЬЗОВАНИЕ PC как Rn осуждается.

Флаги условия Эти инструкции не изменяют флаги.

3.10.28 VSTR

Хранилище с плавающей точкой.

Синтаксис

VSTR {cond} {.32} Sd, [Rn {#imm}]

VSTR {cond} {.64} Dd, [Rn {#imm}]

где:

'cond' является дополнительным кодом условия, см. [Условное выполнение на странице 64](#).

'32, 64' спецификаторы размера дополнительных данных.

'Sd' является исходным регистром для хранилища однословного.

'Dd' является исходным регистром для хранилища двойного слова.

'Rn' является индексным регистром. SP может использоваться.

'imm' + или - непосредственное смещение имеет обыкновение формировать адрес. Значения являются сетью магазинов 4 в диапазоне 0-1020.

imm может быть опущен, означая смещение +0.

Работа

Эта инструкция хранит единственный регистр расширения к памяти, используя адрес от Регистра ядра ARM, с дополнительным смещением, определенным в imm.

Ограничения

ограничения являются использованием PC для Rn.

Флаги условия

Эти инструкции не изменяют флаги.

3.10.29 VSUB

С плавающей точкой Вычитают.

Синтаксис

SUB {cond}.F32 {Sd}, Sn, См

где:

'cond' является дополнительным кодом условия, см. [Условное выполнение на странице 64](#).

'Sd' является местом назначения значение с плавающей точкой

'Sn, См' является операндом значения с плавающей точкой.

Работа

Эта инструкция:

1. Вычитает одно значение с плавающей точкой из другого значения с плавающей точкой.
2. Помещает результаты в место назначения регистр с плавающей точкой.

Ограничения

Не являются никакими ограничениями.

Флаги условия

Эти инструкции не изменяют флаги.

3.11 Разные инструкции

[Таблица 36](#) показывает остающиеся инструкции Cortex-M4:

Таблица 36. Разные инструкции

Мнемосхема	Краткое описание	См.
BKPT	Точка останова	<i>BKPT на странице 170</i>
CPSID	Процессор изменения, Прерывания Отключения	<i>CPS на странице 171</i>
CPSIE	Процессор изменения, Прерывания Включения	<i>CPS на странице 171</i>
DMB	Барьер Памяти данных	<i>DMB на странице 172</i>
DSB	Барьер Синхронизации данных	<i>DSB на странице 172</i>
ISB	Барьер Синхронизации инструкции	<i>ISB на странице 173</i>
MRS	Переместитесь от специального регистра, чтобы зарегистрироваться	<i>MRS на странице 173</i>
MSR	Переместитесь от регистра до специального регистра	<i>MSR на странице 174</i>
NOP	Никакая Работа	NOP на странице 175
SEV	Отправьте Событие	<i>CEB на странице 175</i>
SVC	Вызов супервизора	<i>SVC на странице 176</i>
WFE	Ожидайте События	<i>WFE на странице 176</i>
WFI	Ожидайте Прерывания	<i>WFI на странице 177</i>

3.11.1 BKPT

Точка останова.

Синтаксис

BKPT #imm

где:

'imm' является оценкой выражения к целому числу в диапазоне 0-255 (8-разрядное значение).

Работа

Инструкция ВКРТ заставляет процессор вводить состояние Отладки. Отладчики могут использовать исследовать системное состояние, когда инструкция достигается в определенном адресе.

imm игнорируется процессором. Если требующийся, отладчик используется, чтобы сохранить дополнительную информацию о точке останова.

Инструкция ВКРТ может быть помещена в блоке IT, но она выполняется безоговорочно, незатронутым условием определяется инструкцией IT.

Флаги условия

Эта инструкция не изменяет флаги.

Примеры

ВКРТ 0xAB ; Точка останова с непосредственным набором значений к 0xAB (отладчик может ; извлеките непосредственное значение, определяя местоположение этого использующий PC) ,

3.11.2 CPS

Состояние процессора изменения.

Синтаксис

CPSeffect iflags

где:

'Эффект' является одним из:

IE: Очищает регистр особого назначения.

ID: Устанавливает регистр особого назначения.

'iflags' является последовательностью одного или более флагов:

i: Набор или четкий PRIMASK.

f: Набор или четкий FAULTMASK.

Работа

CPS изменяет PRIMASK и специальные значения регистра FAULTMASK. См. [маску Исключения регистры на странице 22](#) для получения дополнительной информации об этих регистрах.

Ограничения

Ограничения:

Используйте CPS только от привилегированного программного обеспечения, он не имеет никакого эффекта если используется в непривилегированном программном обеспечении

CPS не может быть условным выражением не должен использоваться в блоке IT.

Флаги условия

Эта инструкция не изменяет флаги условия.

Примеры

CPSID ;Прерывания отключения и конфигурируемые обработчики отказа (устанавливает PRIMASK) ,
CPSID f;Прерывания отключения и все обработчики отказа (устанавливает FAULTMASK) ,
CPSIE я;
Прерывания включения и конфигурируемые обработчики отказа (очищают PRIMASK) ,
CPSIE f;Прерывания включения и обработчики отказа (очищают FAULTMASK) ,

3.11.3 DMB

Барьер памяти данных.

Синтаксис

`DMB {cond}`

где: 'cond' является дополнительным кодом условия, см. [Условное выполнение на странице 64](#).

Работа

DMB действует как барьер памяти данных гарантирует что все явные доступы памяти появятся, в порядке программы, прежде, чем инструкция DMB будет завершена перед любыми явными доступами памяти, которые появляются, в порядке программы, после инструкции DMB. DMB не влияет на упорядочивание или выполнение инструкций, которые не получают доступ к памяти.

Флаги условия

Эта инструкция не изменяет флаги.

Примеры

`DMB ; Барьер Памяти Данных`

3.11.4 DSB

Барьер синхронизации данных.

Синтаксис

`DSB {cond}`

где:

'cond' является дополнительным кодом условия, см. [Условное выполнение на странице 64](#).

Работа

DSB действует как специальный барьер памяти синхронизации данных. Инструкции, которые прибывают после DSB, в порядке программы, не выполняются, пока инструкция DSB не завершается. Инструкция DSB завершается, когда все явные доступы памяти перед нею завершаются.

Флаги условия

Эта инструкция не изменяют флаги

. Примеры

`DSB; Барьер Синхронизации Данных`

3.11.5 ISB

Барьер синхронизации инструкции.

Синтаксис

`ISB {cond}`

где:

'cond' является дополнительным кодом условия, см. [Условное выполнение на странице 64](#).

Работа

ISB действует как барьер синхронизации инструкции сбрасывает конвейер процессора, таким образом, все инструкции после ISB выбираются от кэша или памяти снова, после инструкции ISB, было завершено.

Флаги условия

Эта инструкция не изменяет флаги.

Примеры

ISB ; Барьер Синхронизации Инструкции

3.11.6 MRS

Переместите содержание специального регистра к регистру общего назначения.

Синтаксис

MRS {cond} Резерфорд, spec_reg

где:

'cond' является дополнительным кодом условия, см. [Условное выполнение на странице 64](#).

'Резерфорд' является целевым регистром

'spec_reg' может быть любым из: APSR, IPSR, EPSR, IEPSR, IAPSR, EAPSR, PSR, MSP, PSP, PRIMASK, BASEPRI, BASEPRI_MAX, FAULTMASK, или УПРАВЛЕНИЕ.

Работа

Используйте MRS в комбинации с MSR, поскольку часть последовательности "чтение изменяет запись" для того, чтобы обновить а PSR, например чтобы очистить флаг Q. См. [MSR на странице 174](#). В коде подкачки процесса состояние модели программирования выгружаемого процесса должно быть сохранено, включая соответствующее содержание PSR. Точно так же состояние загружаемого процесса должно также быть восстановлено. Эти операции используют MRS в сохраняющей состоянии последовательности инструкции и MSR в восстанавливающей состоянии последовательности инструкции. BASEPRI_MAX является псевдонимом BASEPRI когда используется с инструкцией MRS.

Ограничения

Резерфорд не должен быть SP и не должен быть PC.

Флаги условия

Эта инструкция не изменяет флаги.

Примеры

MRS R0, PRIMASK; Считайте значение PRIMASK и запишите это в R0

3.11.7 MSR

Переместите содержание регистра общего назначения в указанный специальный регистр.

Синтаксис

MSR {cond} spec_reg, Rn

где:

'cond' является дополнительным кодом условия, см. [Условное выполнение на странице 64](#).

'Rn' является исходным регистром.

'spec_reg' может быть любым из: APSR, IPSR, EPSR, IEPSR, IAPSR, EAPSR, PSR, MSP, PSP, PRIMASK, BASEPRI, BASEPRI_MAX, FAULTMASK, или УПРАВЛЕНИЕ.

Работа

Работа доступа регистра в MSR зависит от уровня полномочий. Непривилегированное программное обеспечение может только получить доступ к APSR, смотри [Таблицу 5: определения бита APSR на странице 20](#). Привилегированное программное обеспечение может получить доступ ко всем специальным регистрам. В непривилегированных записях программного обеспечения к битам освобожденного или режима выполнения в PSR игнорируются. Когда Вы пишете в BASEPRI_MAX, инструкция пишет в BASEPRI только если также:

Rn является ненулевым, и текущее значение BASEPRI 0

Rn является ненулевым и меньше чем текущее значение BASEPRI. См. [MRS на странице 173](#).

Ограничения

Rn не должен быть SP и не должен быть PC.

Условие отмечает инструкцию обновления флагов, явно основанных на значении в *Rn*.

Примеры

MSR УПРАВЛЕНИЕ, R1; значение Р1 и запись это к Регистру команд

3.11.8 ТОЛЬКО ДЛЯ УКАЗАННЫХ ЦЕЛЕЙ

Никакая Работа.

Синтаксис

ТОЛЬКО ДЛЯ УКАЗАННЫХ ЦЕЛЕЙ {cond}

где:

'cond' является дополнительным кодом условия, см. [Условное выполнение на странице 64](#).

Работа

ТОЛЬКО ДЛЯ УКАЗАННЫХ ЦЕЛЕЙ ничего не делает. ТОЛЬКО ДЛЯ УКАЗАННЫХ ЦЕЛЕЙ не обязательно отнимающее много времени . Процессор мог бы удалите это из конвейера прежде, чем это достигнет стадии выполнения.

Используйте ТОЛЬКО ДЛЯ УКАЗАННЫХ ЦЕЛЕЙ для того, чтобы дополнить, например поместить следующие инструкции в 64-разрядную границу.

Флаги условия

Эта инструкция не изменяет флаги.

Примеры

ТОЛЬКО ДЛЯ УКАЗАННЫХ ЦЕЛЕЙ ; Никакая работа

3.11.9 СЕВ

Отправьте Событие.

Синтаксис

СЕВ {cond}

где:

'cond' является дополнительным кодом условия, см. [Условное выполнение на странице 64](#).

Работа

СЕВ является инструкцией подсказки, которая заставляет событие быть сообщенным ко всем процессорам в пределах многопроцессорной системы также устанавливает локальный регистр события в 1, см. [Управление электропитанием на странице 46](#).

Флаги условия

Эта инструкция не изменяет флаги.

Примеры

СЕВ; Отправьте Событие

3.11.10 SVC

Вызов супервизора.

Синтаксис

SVC {cond} #imm

где:

'cond' является дополнительным кодом условия, см. [Условное выполнение на странице 64](#)
'imm' является оценкой выражения к целому числу в диапазоне 0-255 (8-разрядное значение).

Работа

Инструкция SVC вызывает исключение SVC. imm игнорируется процессором. Если необходимое, может быть получено обработчиком исключений, чтобы определить, какую службу требуют.

Флаги условия

Эта инструкция не изменяет флаги.

Примеры

SVC 0x32 ; Вызов Супервизора (обработчик SVC может извлечь непосредственное значение
; определяя местоположение этого через сложенный PC)

3.11.11 WFE

Ожидайте События. WFE является инструкцией подсказки.

Синтаксис

WFE {cond}

где: 'cond' является дополнительным кодом условия, см. [Условное выполнение на странице 64](#).

Работа

Если регистр события 0, WFE приостанавливает выполнение, пока одно из следующих событий не имеет место:

Исключение, если не маскирующийся маской исключения регистрируется или текущий приоритетный уровень исключения вводит состояние

На ожидании, если SEVONPEND в Системном Регистре команд устанавливается запрос

Записи Отладки, если Отладка включается событие, сообщенное периферийным устройством или другим процессором в многопроцессорной системе
использование инструкции СЕВА.

Если регистр события 1, WFE очищает его к 0 и сразу возвращается.

Для получения дополнительной информации см. [Управление электропитанием на странице 46](#).

Флаги условия

Эта инструкция не изменяет флаги.

Примеры

WFE ; Ожидайте события

3.11.12 WFI

Ожидайте Прерывания.

Синтаксис

WFI {cond}

где:

'cond' является дополнительным кодом условия, см. [Условное выполнение на странице 64](#).

Работа

WFI является инструкцией подсказки, которая приостанавливает выполнение, пока одно из следующих событий не имеет место: Исключение запрос Записи Отладки, независимо от того, включается ли Отладка.

Флаги условия

Эта инструкция не изменяет флаги.

Примеры

WFI; Ожидайте прерывания

4 Базовые периферийные устройства

4.1 О периферийных устройствах ядра Кору-M4 STM32

Карта адреса *Частной периферийной шины (PPB)*:

Таблица 37. STM32 периферийное устройство ядра регистрирует области PM0214

Адрес	Базовое периферийное устройство	Описание
0xE000E010-0xE000E01F	Системный таймер	Таблица 55 на странице 235
0xE000E100-0xE000E4EF	Вложенный контроллер векторного прерывания	Таблица 49 на странице 204
0xE000ED00-0xE000ED3F	Системный блок управления	Таблица 53 на странице 228
0xE000ED88-0xE000ED8B	Управление доступом сопроцессора математического сопроцессора	Таблица 56 на странице 236
0xE000ED90-0xE000EDB8	Модуль защиты памяти	Таблица 44 на странице 191
0xE000EF00-0xE000EF03	Вложенный контроллер векторного прерывания	Таблица 49 на странице 204
0xE000EF30-0xE000EF44	Математический сопроцессор	Таблица 56 на странице 236

В описаниях регистра,

Тип регистра описывается следующим образом:

- RW: Читайте и запишите
- RO: только для чтения.
- WO: только для записи.

Необходимое полномочие дает уровень полномочий, требуемый получить доступ к регистру, следующим образом:

- Привилегированный:

Только привилегированное программное обеспечение может получить доступ к регистру

- Непривилегированный:

И непривилегированное и привилегированное программное обеспечение может получить доступ к регистру.

4.2 Модуль защиты памяти (MPU)

Этот раздел описывает модуль защиты Памяти (MPU), который реализуется в некоторых Микроконтроллерах STM32. Обратитесь к соответствующей таблице данных устройства, чтобы видеть, присутствует ли MPU в типе STM32, Вы используете. MPU делит карту памяти на многие области, и определяет расположение, размер, права доступа, и атрибуты памяти каждой области. Он поддерживает:

Независимые настройки атрибута для каждой области перекрывающиеся области экспорт памяти приписывает системе.

Атрибуты памяти влияют на поведение доступов памяти к области. Кора - M4 MPU определяет:

Восемь отдельных областей памяти, 0-7 фоновая область памяти.

Когда области памяти накладываются, на доступ памяти влияют атрибуты области с самым большим количеством. Например, атрибуты для области 7 имеют приоритет по атрибутам любой области, которая перекрывает область 7.

У фоновой области памяти есть те же самые атрибуты доступа памяти как память по умолчанию карта, но доступно от привилегированного программного обеспечения только.

Кора-M4 карта памяти MPU объединяется. Это означает доступы инструкции и данные у доступов есть те же самые настройки области.

Если программа получает доступ к расположению памяти, которое запрещается MPU, процессором генерирует отказ управления памятью. Это вызывает исключение отказа, и могло бы вызвать завершение процесса в среде ОС.

В среде ОС ядро может обновить область MPU, устанавливающую динамически основанный на процессе, который будет выполнен. Как правило, встроенный ОС использует MPU для защиты памяти.

Конфигурация областей MPU основана на типах памяти, см. [Раздел 2.2.1: Память области, типы и атрибуты на странице 28](#).

[Таблица 38](#) показывает возможные атрибуты области MPU.

Таблица 38. Память приписывает сводку

Тип памяти	Shareability	Другие атрибуты	Описание
Строго упорядоченный	-	-	Все доступы к Строго упорядоченной памяти происходят в порядке программы. Все Строго упорядоченные области, как предполагается, совместно используются
Устройство	Совместно используемый	-	Отображенные на память периферийные устройства те несколько долей процессоров.
	Несовместно используемый	-	Отображенные на память периферийные устройства, которые использует только единственный процессор.
Нормальный	Совместно используемый	Некэшируемая Запись - через Кэшируемую Обратную запись	Нормальная память, которая совместно используется несколькими процессорами
	Несовместно используемый	Некэшируемая Запись - через Кэшируемую Обратную запись	Нормальная память, которую использует только единственный процессор

4.2.1 Атрибуты права доступа MPU

Этот раздел описывает атрибуты права доступа MPU. Биты права доступа, TEX, C, B, S, AP, и XN, регистра MPU_RASR, управляет доступом к соответствующей области памяти. Если доступ делается к области памяти без необходимых полномочий, то MPU генерирует отказ разрешения. [Таблица 39](#) показывает кодировки для TEX, C, B, и биты права доступа S.

Таблица 39. TEX, C, B, и кодирование S

TEX	C	B	S	Тип памяти	Shareability	Другие атрибуты
b000	0	0	x ⁽¹⁾	Строго упорядоченный	Общий	-
		1	x ⁽¹⁾	Устройство	Общий	-
	1	0	0	Нормальный	Не общий	Внешняя и внутренняя запись - через. Нет запись выделяет.
			1		Общий	
		1	0	Нормальный	Не общий	Внешняя и внутренняя обратная запись. Никакая запись выделить.
			1		Общий	
b001	0	0	0	Нормальный	Не общий	Внешний и внутренний некэшируемый.
			1		Общий	
		1	x ⁽¹⁾	Зарезервированное кодирование		-
	1	0	x ⁽¹⁾	Реализация определила атрибуты.		-
		1	0	Нормальный	Не общий	Внешняя и внутренняя обратная запись. Записать и читайте, выделяют
			1		Общий	
b010	0	0	x ⁽¹⁾	Устройство	Не общий	Несовместно используемое Устройство
		1	x ⁽¹⁾	Зарезервированное кодирование		-
	1	x ⁽¹⁾	x ⁽¹⁾	Зарезервированное кодирование		-
b1BB	A	A	0	Нормальный	Не общий	Кэшируемая память (2), BB = внешний политика, AA = внутренняя политика.
			1		Общий	

1. MPU игнорирует значение fo этот бит. 2. См. [Таблицу 40](#) для кодирования битов BB и AA.

[Таблица 40](#) показывает, что политика кэша для кодировок атрибута памяти со значением TEX находится в диапазоне 4-7.

Таблица 40. Политика кэша для кодирования атрибута памяти

Кодирование, AA или BB	Соответствующая политика кэша
00	Некэшируемый
01	Запишите обратно, запишите и читайте, выделяют
10	Запишите через, никакая запись выделяют
11	Запишите обратно, никакая запись выделяют

[Таблица 41](#) показывает кодировки AP, которые определяют права доступа для привилегированного и непривилегированного программного обеспечения.

Таблица 41. Кодирование AP

AP [2:0]	Привилегированные полномочия	Непривилегированные полномочия	Описание
000	Никакой доступ	Никакой доступ	Все доступы генерируют отказ разрешения
001	RW	Никакой доступ	Доступ от привилегированного программного обеспечения только
010	RW	RO	Записи непривилегированным программным обеспечением генерируют отказ разрешения
011	RW	RW	Полный доступ
100	Непредсказуемый	Непредсказуемый	Зарезервированный
101	RO	Никакой доступ	Чтения привилегированным программным обеспечением только
110	RO	RO	Только для чтения, привилегированным или непривилегированным программным обеспечением
111	RO	RO	Только для чтения, привилегированным или непривилегированным программным

4.2.2 Несоответствие MPU

Когда доступ нарушает полномочия MPU, процессор генерирует память отказ управления, см.

[Раздел 2.1.4: Исключения и прерывания на странице 25](#). MMFSR указывает на причину отказа.

См. [Раздел 4.4.15: управление Памятью дает сбой регистр адреса \(MMFAR\) на странице 226](#)

для получения дополнительной информации.

4.2.3 Обновление области MPU

Чтобы обновить атрибуты для области MPU, обновите MPU_RNR, MPU_RBAR и

Регистры MPU_RASR. Можно программировать каждый регистр отдельно, или использовать запись многократного слова, чтобы программировать все эти регистры. Можно использовать MPU_RBAR и псевдонимы MPU_RASR, чтобы программировать до четырех областей, одновременно используя инструкцию STM.

Обновление области MPU, используя отдельные слова Простой код, чтобы сконфигурировать

одну область:

; R1 = число области; R2 = размер/включение;

R3 = атрибуты; R4 = адрес LDR R0, =MPU_RNR STR R1, [R0, #0x0] STR R4, [R0, #0x4] STRH R2, [R0, #0x8] STRH R3, [R0, #0xA]

; 0xE000ED98, регистр числа области MPU; Число Области; Базовый адрес

Области; Размер Области и Включение; Атрибут Области

Отключите область прежде, чем записать новые настройки области в MPU, если Вы имеете ранее включенную изменяемую область. Например:

; R1 = число области

; R2 = размер/включение

; R3 = атрибут

; R4 = адрес

LDR R0, =MPU_RNR STR R1, [R0, #0x0] BIC R2, R2, #1 STRH R2, [R0, #0x8] STR R4, [R0, #0x4] STRH R3, [R0, #0xA] ORR R2, #1 STRH R2, [R0, #0x8]

; 0xE000ED98, регистр числа области MPU; Число Области; Отключение; Размер Области и

Включение; Базовый адрес Области; Атрибут Области; Включение; Размер Области и

Включение

Программное обеспечение должно использовать инструкции барьера памяти:

Прежде, чем MPU устанавливаются, если могли бы быть выдающиеся передачи памяти, такой как буферирование записи, на которые могло бы влиять изменение в настройках MPU

После того, как MPU устанавливаются, если это включает передачи памяти, которые должны использовать новые настройки MPU. Однако, инструкции барьера памяти не требуются, если процесс установки MPU запускается, вводя обработчик исключений, или сопровождается возвратом исключения, потому что запись исключения и исключение возвращают поведение барьера памяти причины механизма.

Программное обеспечение не нуждается ни в каких инструкциях барьера памяти во время установки MPU, потому что получает доступ к MPU через PPB, который является Строго упорядоченной областью памяти.

Например, если Вы хотите, чтобы все поведение доступа памяти сразу вступило в силу после последовательности программирования, используйте инструкцию DSB и инструкцию ISB: DSB требуется после изменения настроек MPU, такой как в конце контекстного переключения. ISB требуется, если код, который программирует область MPU или области, вводится, используя ответвление или вызов. Если последовательность программирования вводится, используя возврат из исключения, или беря исключение, то Вы не требуете ISB.

Обновление области MPU, используя записи многословные

Можно программировать непосредственно использующие записи многословные, в зависимости от того, как информация разделена. Рассмотрите следующее перепрограммирование:

```
; R1 = число области
; R2 = адрес
; R3 = размер, атрибуты в одном
LDR R0, =MPU_RNR      ; 0xE000ED98, регистр числа области MPU STR R1, [R0,
#0x0]; Число Области STR R2, [R0, #0x4]    ; Базовый адрес Области STR R3, [R0,
#0x8]    ; Атрибут Области, Размер и Включение
```

Используйте инструкцию STM, чтобы оптимизировать это:

```
; R1 = число области; R2 = адрес; R3 = размер, атрибуты в одном
LDR R0, =MPU_RNR      ; 0xE000ED98, регистр числа области MPU STM R0, {R1-R3} ; Число
Области, адрес, атрибут, размер и включение
```

Можно сделать это в двух словах для предварительно упакованной информации. Это означает, что RBAR содержит необходимое число области и имеет ДОПУСТИМЫЙ набор битов к 1, см. [индексный регистр области MPU \(MPU_RBAR\) на странице 188](#). Используйте его, когда данные статически упаковываются, например в загрузчике:

```
; R1 = адрес и число области в одном; R2 = размер и атрибуты в одном
LDR R0, =MPU_RBAR      ; 0xE000ED9C, Индексный регистр Области MPU STR R1, [R0, #0x0]
; базовый адрес Области и
; число области объединило с ДОПУСТИМЫМ (бит 4) набор к 1 STR R2, [R0, #0x4]
; Атрибут Области, Размер и Включение
```

Используйте инструкцию STM, чтобы оптимизировать :

```
LDR R0, =MPU_RBAR
STM R0, {R1-R2}
```

Подобласти

; R1 = адрес и число области в одном; R2 = размер и атрибуты в одном; 0xE000ED9C, Индексный регистр Области MPU; базовый адрес Области, число области и ДОПУСТИМЫЙ бит; и Атрибут Области, Размер и Включение

Области 256 байтов или больше делятся на восемь подобластей равного размера. Установите соответствующий бит в поле SRD RASR, чтобы отключить подобласть, см. [Раздел 4.2.9: атрибут области MPU и регистр размера \(MPU_RASR\) на странице 189](#). Младший значащий бит SRD управляет первой подобластью, и старший значащий бит управляет последней подобластью. Отключение подобласти означает другую область, перекрывающую отключенные соответствия диапазона вместо этого. Если никакая другая включенная область не перекрывает отключенную подобласть, MPU выпускает отказ. Области 32, 64, и 128 байтов не поддерживают подобласти, С областями этих размеров, следует установить поле SRD в 0x00, иначе поведение MPU Непредсказуемо.

Пример использования SRD: Две области с тем же самым перекрытием базового адреса. Область каждый - 128 Кбит, и область два, 512 Кбит. Чтобы гарантировать атрибуты от области, каждый применяется к first128KB области, устанавливая поле SRD для области два к b00000011, чтобы отключить первые две подобласти, как шоу числа.

4.2.4 MPU разрабатывают полезные советы

Чтобы избежать неожиданного поведения, отключите прерывания прежде, чем обновить атрибуты области, к которой могли бы получить доступ обработчики прерываний.

Гарантируйте использованию программного обеспечения выровненные доступы корректного размера к доступу регистра MPU:

За исключением RASR, должно использовать выровненные доступы слова **ДЛЯ** RASR это может использовать байт или выровненные доступы полуслова или слова.

Процессор не поддерживает не выровненные доступы к регистрам MPU. Устанавливая MPU, и если MPU был ранее запрограммирован, отключение неиспользованные области, чтобы препятствовать тому, чтобы любые предыдущие настройки области влияли на новую установку MPU.

Рекомендуемая конфигурация MPU

У системы микроконтроллера STM32 есть только единственный процессор, таким образом, следует программировать MPU следующим образом:

Таблица 42. Область памяти приписывает для STM32

Область памяти	TEX	C	B	S	Тип памяти и атрибуты
Флэш-память	b000	1	0	0	Нормальная память, Необщая, пишет - через
Внутренний SRAM	b000	1	0	1	Нормальная память, Общая, пишет - через
Внешний SRAM	b000	1	1	1	Нормальная память, Общая, обратная запись, пишет - выделяют
Периферийные устройства	b000	0	1	1	Память устройства, Общая

В реализациях STM32 shareability и атрибуты политики кэша не влияют на поведение системы. Однако, использование этих настроек для областей MPU может сделать код программы более переносимым. Данные значения для типичных ситуаций.

Отметьте: Атрибуты MPU не влияют на доступы данных DMA к адресу памяти/периферийных устройств пробы. поэтому, чтобы защитить области памяти от непреднамеренных доступов DMA, MPU должен управлять доступом SW/ЦП к регистрам DMA.

4.2.5 MPU вводят регистр (MPU_TYPER)

Адрес смещения: значение Сброса 0x00: 0x0000 0800

Необходимые полномочия: Привилегированный

Регистр MPU_TYPER указывает, присутствует ли MPU, и если так, сколько областей это поддерживает.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								IREGION[7:0]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DREGION[7:0]								Reserved							SEPA RATE
r	r	r	r	r	r	r	r								r

Биты 31:24 **Зарезервированный**.

Биты 23:16 **IREGION [7:0]**: области инструкции Number of MPU. Эти биты указывают на число поддерживаемых областей инструкции MPU. Всегда содержит 0x00. Карта памяти MPU объединяется и описывается полем DREGION.

Биты 15:8 **DREGION [7:0]**: Число областей данных MPU.

Эти биты указывают на число поддерживаемых областей данных MPU. 0x08: Восемь областей MPU 0x00: MPU, не существующий

Биты 7:1 **Зарезервированный**.

Бит 0 **ОТДЕЛЬНЫЙ**: Отдельный флаг.

Этот бит указывает на поддержку объединенной или отдельной инструкции и карт памяти данных: 0 = Объединенный 1 = Отдельный

4.2.6 Регистр команд MPU (MPU_CTRL)

Адрес смещения: значение Сброса 0x04: 0x0000 0000

Необходимые полномочия: Привилегированный регистр MPU_CTRL:

Включает MPU т фоновой области памяти карты памяти по умолчанию

Использование включений MPU, когда в твердом отказе, Немаскируемое прерывание (NMI), и FAULTMASK наращивал обработчики.

Когда ВКЛЮЧЕНИЕ и PRIVDEFENA оба устанавливаются в 1:

Для привилегированных доступов карта памяти по умолчанию как описывается в [Разделе 2.2: Модель памяти на странице 27](#). Любой доступ привилегированным программным обеспечением, которое не адресует включенную область памяти, ведет себя как определено картой памяти по умолчанию.

Любой доступ непривилегированным программным обеспечением, которое не адресует включенную область памяти вызывает отказ управления памятью.

XN и Строго упорядоченные правила всегда применяются к Системному Пространству Управления независимо от значение бита ВКЛЮЧЕНИЯ. Когда бит ВКЛЮЧЕНИЯ устанавливается в 1, по крайней мере одна область карты памяти должна быть включена для системы, чтобы функционировать, если бит PRIVDEFENA не устанавливается в 1. Если бит PRIVDEFENA устанавливается в 1, и никакие области не включаются, то только привилегированное программное обеспечение может работать.

Когда бит ВКЛЮЧЕНИЯ устанавливается в 0, система использует карту памяти по умолчанию. Это имеет

те же самые атрибуты памяти, как будто MPU не реализуется, смотри [Таблицу 13: поведение доступа Памяти на странице 29](#). Карта памяти по умолчанию применяется к доступам и от привилегированного и от непривилегированного программного обеспечения.

Когда MPU включается, доступы к Системному Пространству Управления и таблице векторов всегда разрешаются. Другие области доступны на областях и устанавливается ли PRIVDEFENA в 1.

Если HFNMIENA не устанавливается в 1, MPU не включается, когда процессор выполняется обработчик для исключения с приоритетом-1 или-2. Эти приоритеты только возможны, обрабатывая твердый отказ или исключение NMI, или когда FAULTMASK включается. Установка бита HFNMIENA к 1 включению MPU, работая с этими двумя приоритетами.

Reserved															

Биты 31:3 **Зарезервированный, вызванный аппаратными средствами к 0.**

Бит 2 **PRIVDEFENA**: Включите privileged доступу программного обеспечения к карте памяти по умолчанию.

0: Если MPU включает, использование отключений карты памяти по умолчанию. Любой доступ памяти к расположению, не покрытому любой включенной областью, вызывает отказ.

1: Если MPU включается, использование включений карты памяти по умолчанию как фоновая область памяти для привилегированных доступов программного обеспечения.

Отметьте: Когда включено, фоновая область памяти действует, как будто это - область номер-1. Любая область определяется и включается, имеет приоритет над этой картой по умолчанию. Если MPU отключается, процессор игнорирует этот бит.

Бит 1 **HFNMIENA**: Включает работе MPU во время твердого отказа, NMI, и обработчиков FAULTMASK. Когда MPU включается:

0: MPU отключается во время твердого отказа, NMI, и обработчиков FAULTMASK, независимо от значения бита ВКЛЮЧЕНИЯ

1: MPU включается во время твердого отказа, NMI, и обработчиков FAULTMASK.

Отметьте: Когда MPU отключается, если этот бит устанавливается в 1, поведение непредсказуемо. Бит 0 **ВКЛЮЧЕНИЙ**: Включает MPU

0: MPU отключил

1: MPU включается

4.2.7 Регистр числа области MPU (MPU_RNR)

Адрес смещения: значение Сброса 0x08: 0x0000 0000

Необходимые полномочия: Привилегированный регистр MPU_RNR выбирает, на какую область памяти ссылается MPU_RBAR и Регистры MPU_RASR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								REGION[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Биты 31:8 **Зарезервированный, вызванный аппаратными средствами к 0.**

Биты 7:0 **ОБЛАСТЬ [7:0]**: область MPU

Эти биты указывают на область MPU, на которую ссылаются регистры MPU_RASR и MPU_RBAR. MPU поддерживает 8 областей памяти, таким образом, разрешенные значения этого поля 0-7. Обычно, Вы пишете необходимое число области в этот регистр прежде, чем получить доступ к MPU_RBAR или

MPU_RASR. Однако можно изменить число области при записи в регистр MPU_RBAR с ДОПУСТИМЫМ набором битов к 1, смотри [индексный регистр области MPU \(MPU_RBAR\)](#). Эта запись обновляет значение поля REGION.

4.2.8 Индексный регистр области MPU (MPU_RBAR)

Адрес смещения: значение Сброса 0x0C: 0x0000 0000

Необходимые полномочия: Привилегированный

Регистр MPU_RBAR определяет базовый адрес области MPU, выбранной Регистром MPU_RNR, и может обновить значение регистра MPU_RNR. Запишите в регистр MPU_RBAR с ДОПУСТИМЫМ набором битов к 1, чтобы изменить текущую область число и обновление регистр MPU_RNR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDR[31:N]...															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
....ADDR[31:N]											VALID	REGION[3:0]			
											rw	rw	rw	rw	rw

Биты 31:N **ADDR [31:N]**: поле базового адреса Области Значение N зависит от размера области. Размер области, как определено полем SIZE в MPU_RASR, определяет значение N:

$N = \text{Log}_2$ (Размер области в байтах), Если размер области конфигурируется к 4 Гбайт, в регистре MPU_RASR, нет никакого допустимого поля ADDR. В этом случае, область занимает полную карту памяти, и базовый адрес является 0x00000000. Базовый адрес выровненный к размеру области. Например, область на 64 Кбайта должна быть выровненная на кратном числе 64 Кбайт, например, в 0x00010000 или 0x00020000.

Биты N-1:5, **Зарезервированный, вызванный аппаратными средствами к 0.**

Бит 4 **ДОПУСТИМЫЙ**: допустимое число области MPU

Запишите:

0: Регистр MPU_RNR, не измененный, и процессор:

- Обновляет базовый адрес для области, определенной в MPU_RNR
- Игнорирует значение поля REGION

1: процессор:

- обновляет значение MPU_RNR к значению поля REGION
- обновляет базовый адрес для области, определенной в поле REGION.

Читайте:

Всегда читайте как ноль. Биты 3:0 **ОБЛАСТЬ [3:0]**: поле области MPU

Для поведения на записях, см. описание поля VALID. На чтениях, возвращает текущее число области, как определено регистром MPU_RNR.

4.2.9 Атрибут области MPU и регистр размера (MPU_RASR)

Адрес смещения: значение Сброса 0x10: 0x0000 0000

Необходимые полномочия: Привилегированный

Регистр MPU_RASR определяет размер области и атрибуты памяти области MPU определенный MPU_RNR, и включениями, что область и любые подобласти.

MPU_RASR является доступными доступами слова или полуслова использования:

Старшее значащее полуслово содержит атрибуты области младшее значащее полуслово содержит размер области и область и подобласть биты включения.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			XN	Reserved	AP[2:0]			Reserved		TEX[2:0]			S	C	B
			rw		rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRD[7:0]								Reserved		SIZE				ENABLE	
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Биты 31:29 **Зарезервированный, вызванный аппаратными средствами к 0.**

Бит 28 **XN**: доступ Инструкции отключает бит:

0: Выборки инструкции включали

1: выборки Инструкции отключаются.

Бит 27 **Зарезервированный, вызванный аппаратными средствами к 0.**

Биты 26:24 **AP [2:0]**: Право доступа

Для получения информации о праве доступа, см. [Раздел 4: Базовые периферийные устройства](#) Поскольку описание кодирования битов AP относится к [Таблице 41 на странице 181](#).

Биты 23:22 **Зарезервированный, вызванный аппаратными средствами к 0.**

Биты 21:19 **TEX [2:0]**: атрибут памяти

Поскольку описание кодирования битов TEX относится к [Таблице 39 на странице 180](#)

Бит 18 **S**: **Общий атрибут памяти**

Поскольку описание кодирования битов S отсылает к [Таблице 39 на странице 180](#)

битов 17 **C**: Бит **атрибута памяти**

16 **B**: **атрибут памяти**

Биты 15:8 **SRD**: биты отключения Подобласти. Для каждого бита в этом поле:

0: соответствующая подобласть включается

1: соответствующая подобласть отключается

См. [Подобласти на странице 183](#) для получения дополнительной информации. Размеры области 128 байтов и меньше не поддерживает подобласти. При записи атрибутов для такой области, запишите поле SRD как 0x00.

Биты 7:6 **Зарезервированный, вызванный аппаратными средствами к 0.**

Биты 5:1 **РАЗМЕР**: Размер области защиты MPU.

Минимальное разрешенное значение 3 (b00010), см. [значения полей РАЗМЕРА](#) для получения дополнительной информации.

Бит 0 **ВКЛЮЧЕНИЙ**: бит включения Области.

Значения полей РАЗМЕРА

Поле SIZE определяет размер области памяти MPU, определенной MPU_RNR register следующим образом:

(Размер области в байтах), = 2 (SIZE+1) самый маленький разрешенный размер области 32B, соответствующая значению РАЗМЕРА 4. [Таблица 43](#) дает значения РАЗМЕРА в качестве примера, с соответствующим размером области и значением N в MPU_RBAR.

Таблица 43. Значения полей РАЗМЕРА в качестве примера

Значение РАЗМЕРА	Размер области	Значение N (1)	Отметить
b00100(4)	32B	5	Минимальный разрешенный размер
b01001 (9)	1KB	10	-
b10011 (19)	1MB	20	-
b11101 (29)	1GB	30	-
b11111 (31)	4GB	b01100	Максимальный возможный размер

1. В MPU_RBAR регистр см. [Раздел 4.2.8 на странице 188](#)

4.2.10 MPU регистрирующие карту

Таблица 44. MPU регистрирующие значения сброса и карты

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
0x00	MPU_TYPER	Reserved								IREGION[7:0]								DREGION[7:0]								Reserved								SEPARATE			
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x04	MPU_CTRL	Reserved																														PRIVDEFENA[1]	HFNMENA[1]				
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x08	MPU_RNR	Reserved																				REGION[7:0]															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x0C	MPU_RBAR	ADDR[31:N]...																				VALID 1				REGION[3:0]											
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x10	MPU_RASR	Reserved		NX	Reserved 1		AP[2:0]		Reserved		TEX[2:0]		S	C	B	SRD[7:0]								Reserved		SIZE				ES[8]							
	Reset Value	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x14	MPU_RBAR_A1 ⁽¹⁾	ADDR[31:N]...																				VALID 1				REGION[3:0]											
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x18	MPU_RASR_A1 ⁽²⁾	Reserved		NX	Reserved 1		AP[2:0]		Reserved		TEX[2:0]		S	C	B	SRD[7:0]								Reserved		SIZE				ES[8]							
	Reset Value	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x1C	MPU_RBAR_A2 ⁽¹⁾	ADDR[31:N]...																				VALID 1				REGION[3:0]				ES[8]							
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x20	MPU_RASR_A2 ⁽²⁾	Reserved		NX	Reserved 1		AP[2:0]		Reserved		TEX[2:0]		S	C	B	SRD[7:0]								Reserved		SIZE				ES[8]							
	Reset Value	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

4.3 Вложенный контроллер векторного прерывания (NVIC)

Этот раздел описывает Вложенный Контроллер Векторного прерывания (NVIC) и регистры использования. NVIC поддерживает: До 81 прерывания (номер прерывания зависит от типа устройства STM32; обратитесь к таблицы данных)

Программируемый приоритетный уровень 0-15 для каждого прерывания. Более высокий уровень соответствует более низкому приоритету, таким образом, уровень 0 является самым высоким приоритетом прерывания

Уровень и импульсное обнаружение сигналов прерывания динамическое пере установление приоритетов прерываний группировка приоритета оценивает в групповые приоритетные и под

приоритетные поля объединение в цепочку хвоста прерывания внешнее *Немаскируемое прерывание* (NMI) Процессор автоматически складывает свое состояние на записи исключения и не складывает это состояние на выходе исключения, без издержек инструкции. Это обеспечивает низкую обработку исключений задержки. Аппаратная реализация регистров NVIC:

Таблица 45. Сводка регистра NVIC

Адрес	Имя	Ввести	Необходимое полномочие	Значение сброса	Описание
0xE000E100-0xE000E10B	NVIC_ISER0-NVIC_ISER2	RW	Привилегированный	0x00000000	Таблица 4.3.2: регистры включения набора Прерывания (NVIC_ISERx) на странице 195
0xE000E180-0xE000E18B	NVIC_ICER0-NVIC_ICER2	RW	Привилегированный	0x00000000	Таблица 4.3.3: регистры четкого включения Прерывания (NVIC_ICERx) на странице 196
0xE000E200-0xE000E20B	NVIC_ISPR0-NVIC_ISPR2	RW	Привилегированный	0x00000000	Таблица 4.3.4: Прервите ожидающие набор регистры (NVIC_ISPRx) на странице 197
0xE000E280-0xE000E29C	NVIC_ICPR0-NVIC_ICPR2	RW	Привилегированный	0x00000000	Таблица 4.3.5: Прервите четки на ожидании регистры (NVIC_ICPRx) на странице 198
0xE000E300-0xE000E31C	NVIC_IABR0-NVIC_IABR2	RW	Привилегированный	0x00000000	Таблица 4.3.6: Прервите активные разрядные регистры (NVIC_IABRx) на странице 199
0xE000E400-0xE000E503	NVIC_IPR0-NVIC_IPR20	RW	Привилегированный	0x00000000	Таблица 4.3.7: приоритетные регистры Прерывания (NVIC_IPRx) на странице 200
0xE000EF00	STIR	WO	Конфигурируемый	0x00000000	Таблица 4.3.8: триггер программного обеспечения прерывает регистр (NVIC_STIR) на странице 201

4.3.1 Доступ к Коре-M4 регистры NVIC, используя CMSIS

CMSIS функционирует мобильность программного обеспечения включения между различными М. коры процессоров профиля.

Чтобы получить доступ к регистрам NVIC при использовании CMSIS, используйте следующие функции:

Таблица 46. Доступ CMSIS функции НВИЧА

Функция CMSIS (1)	Описание
void NVIC_EnableIRQ(IRQn_Type IRQn)	Включает прерыванию или исключению.
void NVIC_DisableIRQ(IRQn_Type IRQn)	Отключает прерывание или исключение.
void NVIC_SetPendingIRQ(IRQn_Type IRQn)	Устанавливает состояние на ожидании прерывания или исключения к 1.
void NVIC_ClearPendingIRQ(IRQn_Type IRQn)	Очищает состояние на ожидании прерывания или исключения к 0.
uint32_t NVIC_GetPendingIRQ(IRQn_Type IRQn)	Читает состояние на ожидании прерывания или исключения. Эта функция возвращается не - нулевое значение, если состояние на ожидании устанавливается в 1.
void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)	Устанавливает приоритет прерывания или исключения с конфигурируемым приоритетным уровнем к 1.
uint32_t NVIC_GetPriority(IRQn_Type IRQn)	Читает приоритет прерывания или исключения с конфигурируемым приоритетным уровнем. Эта функция возвращает текущий приоритетный уровень

1. Входной параметр IRQn является числом IRQ,

4.3.2 Регистры включения набора прерывания (NVIC_ISERx)

Адрес смещения: 0x00 - 0x0B значение Сброса: 0x0000 0000

Необходимые полномочия: Привилегированный

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SETENA[31:16]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETENA[15:0]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Биты 31:0 **SETENA**: биты включения набора Прерывания.

Запишите:

0: Никакой эффект

1: прерывание Включения

Читайте:

0: Прервите отключенный

1: Прерывание включается.

Если прерывание на ожидании включается, NVIC активирует прерывание, основанное на его приоритете. Если прерывание не включается, утверждая, что его сигнал прерывания изменяет состояние прерывания на ожидание, но NVIC никогда не активирует прерывание, независимо от его приоритета.

4.3.3 Регистры четкого включения прерывания (NVIC_ICERx)

Адрес смещения: 0x00 - 0x0B значение Сброса: 0x0000 0000

Необходимые полномочия: Привилегированный

ICER0-ICER2 регистрирует прерывания отключения, и шоу, какие прерывания включаются.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLRENA[31:16]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLRENA[15:0]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Биты 31:0 **CLRENA**: биты четкого включения Прерывания.

Запишите:

0: Никакой эффект

1: прерывание Отключения

Читайте:

0: Прервите отключенный

1: Прерывание включается.

4.3.4 Прервите ожидающие набор регистры (NVIC_ISPRx)

Адрес смещения: 0x00 - 0x0B значение Сброса: 0x0000 0000

Необходимые полномочия: Привилегированный

ISPR0-ISPR2 регистрирует прерывания силы в состоянии на ожидании, и шоу прерывания находятся на рассмотрении.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SETPEND[31:16]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETPEND[15:0]															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Биты 31:0 **SETPEND**: Прервите ожидающие набор биты

Запишите:

0: Никакой эффект

1: прерывание Изменений утверждает к ожиданию

Читайте:

0: Прерывание не ожидает

1: Прерывание находится на рассмотрении

Запись 1 к биту ISPR, соответствующему прерыванию, которое находится на рассмотрении:

- **НЕ** имеет никакого эффекта.

Запись 1 к биту ISPR, соответствующему заблокированному прерыванию:

- устанавливает состояние того прерывания к ожиданию.

4.3.5 Прервите четки на ожидании регистры (NVIC_ICPRx)

Адрес смещения: 0x00 - 0x0B значение Сброса: 0x0000 0000

Необходимые полномочия: Привилегированный

Регистры ICPR0-ICPR2 удаляют состояние на ожидании из прерываний, и шоу прерывания находятся на рассмотрении.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLRPEND[31:16]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLRPEND[15:0]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Биты 31:0 **CLRPEND**: Прервите на ожидании биты

Запишите:

0: Никакой эффект

1: Удаляет состояние на ожидании прерывания

Читайте:

0: Прерывание не ожидает

1: Прерывание находится на рассмотрении

Запись 1 к биту ICPR не влияет на активное состояние соответствующего прерывания.

4.3.6 Прервите активные разрядные регистры (NVIC_IABRx)

Адрес смещения: 0x00 - 0x0B

значение Сброса: 0x0000 0000

Необходимые полномочия: Привилегированный

регистры IABR0-IABR2 указывают, какие прерывания являются активными. Разрядные присвоения:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACTIVE[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACTIVE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Биты 31:0 **АКТИВНЫЙ**: Прервите активные флаги

0: Прервите не активный

1: активное Прерывание

Маленькие чтения как 1, если состояние соответствующего прерывания является активным или активным и на ожидании.

4.3.7 Приоритетные регистры прерывания (NVIC_IPRx)

Адрес смещения: 0x00 - 0x0B

значение Сброса: 0x0000 0000

Необходимые полномочия: Привилегированный

Регистры NVIC_IPR0-IPR80 обеспечивают 8-разрядное приоритетное поле для каждого прерывания. Они регистры доступны для байта. Каждый регистр содержит четыре приоритетных поля, карту к четырем элементам в приоритетном IP массива прерывания CMSIS [0] к IP [67], как показано в [рисунке 19](#).

Рисунок 19. Отображение регистра NVIC_IPRx

31:24	23:16	15:8	7:0
Reserved	Reserved	Reserved	IP[80]

IP[4m+3]	IP[4m+2]	IP[4m+1]	IP[4m]
----------	----------	----------	--------

IP[3]	IP[2]	IP[1]	IP[0]
-------	-------	-------	-------

Таблица 47. Присвоения бита IPR

Биты	Имя
[31:24]	Приоритет, байтовое смещение 3
[23:16]	Приоритета, байтовое смещение 2
[15:8]	Приоритета, байтовое смещение 1
[7:0]	Приоритет, байтовое смещение 0

Каждое приоритетное поле содержит приоритетное значение, 0-255. Чем ниже значение, тем больше приоритет соответствующего прерывания. Процессор реализует только биты [7:4] каждого поля, биты [3:0] чтение как ноль, и проигнорируйте записи.

См. [регистры включения набора Прерывания \(NVIC_ISERx\) на странице 195](#) для получения дополнительной информации о приоритетном массиве прерывания, который обеспечивает представление программного обеспечения приоритетов прерывания.

Найдите число IPR и байтовое смещение для прерывания *N* следующим образом:

Соответствующее число IPR, *M*., дается $M = N \text{ DIV } 4$ байтовое смещение необходимого поля

Priority в этом регистре является $\text{MOD } N \text{ } 4$, где:

- байтовое смещение 0 относится, чтобы зарегистрировать биты [7:0]
- байтовое смещение 1 относится, чтобы зарегистрировать биты [15:8]
- байтовое смещение 2 относится, чтобы зарегистрировать биты [23:16]
- байтовое смещение 3 относится, чтобы зарегистрировать биты [31:24].

4.3.8 Триггер программного обеспечения прерывает регистр (NVIC_STIR)

Адрес смещения: значение Сброса 0xE00: 0x0000 0000

Необходимое полномочие: Когда бит USERSETMPEND в SCR устанавливается в 1, непривилегированное программное обеспечение может получить доступ к ДВИЖЕНИЮ, смотри [Раздел 4.4.6: Системный регистр команд \(SCR\)](#). Только привилегированное программное обеспечение может включить непривилегированный доступ к ДВИЖЕНИЮ.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								INTID[8:0]							
								w	w	w	w	w	w	w	w

Биты 31:9 Зарезервированный, должен быть сохранен очищенным.

Биты 8:0 программное обеспечение **INTID** генерирует ID прерывания

Запишите в ДВИЖЕНИЕ, чтобы генерировать программное обеспечение Сгенерированное Прерывание (SGI). Значение, которое будет записано, является ID Прерывания необходимого SGI, в диапазоне 0-239. Например, значение 0x03 определяет прерывание IRQ3.

4.3.9 Чувствительные к уровню и импульсные прерывания

Прерывания STM32 и чувствительны к уровню и чувствительны к импульсу. Импульсные прерывания также описываются как инициированные фронтом сигнала прерывания.

Чувствительное к уровню прерывание считается утверждаемое до периферийного устройства deasserts сигнал прерывания.

Обычно это происходит, потому что ISR получает доступ к периферийному устройству, заставляя его очистить запрос на прерывание. Импульсное прерывание является сигналом прерывания, выбранным синхронно на возрастающем краю часов процессора. Гарантирует NVIC обнаружение прерывания, Периферийное устройство должно утверждать сигнал прерывания по крайней мере для одного такта, во время которого NVIC обнаруживает импульс и фиксирует прерывание.

Когда процессор вводит ISR, он автоматически удаляет состояние на ожидании из прерывания, см.

[Аппаратное и программное управление прерываний](#). Для чувствительного к уровню прерывания, если сигнал не является deasserted перед возвратами процессора из ISR, прерывание становится ожиданием снова, и процессор должен выполнить свой ISR снова. Это означает, что периферийное устройство может содержать сигнал прерывания, утверждаемый, пока это больше не нуждается в обслуживании.

Аппаратное и программное управление прерываний

Кора-M4 фиксирует все прерывания. Прерывание ввода-вывода становится ожиданием для одной из следующих причин:

NVIC обнаруживает, что сигнал прерывания ВЫСОК, и прерывание не активно NVIC обнаруживает возрастающий край на сигнале прерывания программное обеспечение пишет в соответствующее ожидание набора прерывания, зарегистрируйте бит, смотри [Раздел 4.3.4: Прервите ожидающие набор регистры \(NVIC_ISPRx\)](#), или к ДВИЖЕНИЮ, чтобы сделать ожидание SGI, см. [Раздел 4.3.8: триггер программного обеспечения прерывает регистр \(NVIC_STIR\)](#).

Прерывание на ожидании остается ожидать до одного из следующего:

Процессор вводит ISR для прерывания. Это изменяет состояние прерывания от ожидания до активного. Затем:

- Для чувствительного к уровню прерывания, когда процессор возвращается из ISR, NVIC выбирает сигнал прерывания. Если сигнал утверждается, состояние прерывания изменяется на ожидание, которое могло бы заставить процессор сразу повторно входить в ISR. Иначе, состояние прерывания изменяется на неактивное.
- Для импульсного прерывания NVIC продолжает контролировать сигнал прерывания, и если он

пульсирует состояние прерывания изменяется на ожидание и активный. В этом случае, когда возвраты процессора из ISR, который состояние прерывания изменяет на ожидание, которое могло бы заставить процессор сразу повторно входить в ISR. Если сигнал прерывания не пульсирует, в то время как процессор находится в ISR, когда возвраты процессора из ISR состояние прерывания изменяются на неактивный.

Программное обеспечение пишет в соответствующий бит регистра ожидания прерывания.

Для чувствительного к уровню прерывания, если сигнал прерывания все еще утверждается, не изменяет состояние прерывания. Иначе, состояние прерывания изменяется на неактивный. Для импульсного прерывания состояние прерывания изменяется на:

- Неактивный, если состояние находилось на рассмотрении
- Активный, если состояние было активным и на ожидании.

4.3.10 Полезные советы проекта NVIC

Гарантируйте использованию программного обеспечения правильно выровненные доступы

регистра. Процессор не поддерживает не выровненные доступы к регистрам NVIC. См.

отдельные описания регистра для поддерживаемых размеров доступа. Прерывание может ввести состояние на ожидании, даже это отключается. Отключение прерывания только предотвращает процессор от взятия того прерывания. Прежде, чем запрограммировать VTOR, чтобы переместить таблицу векторов, гарантируйте записи таблицы векторов новая таблица векторов является установкой для обработчиков отказа, NMI и всего включенного исключения как прерывания. Для получения дополнительной информации см. [Раздел 4.4.4: регистр смещения Таблицы векторов \(VTOR\) на странице 212.](#)

NVIC, программирующий подсказки

Программное обеспечение использует CPSIE I и CPSID I инструкций, чтобы включить и отключить прерывания.

CMSIS обеспечивает следующие встроенные функции для этих инструкций:

```
освободите __disable_irq (пусто) //Прерывания Отключения пусто __enable_irq  
(пусто) //Прерывания Включения
```

Кроме того, CMSIS обеспечивает много функций для управления NVIC, включая:

CMSIS прерывают функцию управления	Описание
void NVIC_SetPriorityGrouping(uint32_t priority_grouping)	Установите приоритетную группировку
void NVIC_EnableIRQ(IRQn_t IRQn)	Включение IRQn
void NVIC_DisableIRQ(IRQn_t IRQn)	Отключение IRQn
uint32_t NVIC_GetPendingIRQ(IRQn_t IRQn)	Возвратите true (Число IRQ), если IRQn находится на рассмотрении
void NVIC_SetPendingIRQ(IRQn_t IRQn)	Набор ожидание IRQn
void NVIC_ClearPendingIRQ(IRQn_t IRQn)	Очистите IRQn состояние на ожидании
uint32_t NVIC_GetActive(IRQn_t IRQn)	Возвратите число IRQ активного прерывания
void NVIC_SetPriority(IRQn_t IRQn, uint32_t priority)	Приоритет набора для IRQn
uint32_t NVIC_GetPriority(IRQn_t IRQn)	Считайте приоритет IRQn
void NVIC_SystemReset(void)	Сбросьте систему

Входной параметр IRQn является числом IRQ, смотри [Таблицу 17: Свойства различного исключения вводят на странице 37](#). Для получения дополнительной информации об этих функциях см. документацию CMSIS.

4.3.11 Карта регистра NVIC

Эта таблица показывает карту регистра NVIC и значения сброса. Базовый адрес основного

Блока регистра NVIC является 0xE00E100. Регистр NVIC_STIR располагается в отдельном блоке в 0xE00EF00.

Таблица 49. Карта регистра NVIC и значения сброса

[illegible]

Таблица 49. Карта регистра NVIC и значения сброса (продолжение)

[illegible]

4.4 Системный блок управления (SCB)

Системный блок управления (SCB) предоставляет системную информацию о реализации, и систему управление включает конфигурацию, управление, и создание отчетов системных исключений.

Таблица 50. Сводка системных регистров блока управления

Адрес	Имя	Введи	Необходимо полномочие	Значение сброса	Описание
0xE000E008	ACTLR	RW	Привилегированный	0x00000000	Таблица 4.4.1: Вспомогательный регистр команд (ACTLR) на странице 207
0xE000ED00	CPUID	RO	Привилегированный	0x410FC241	Таблица 4.4.2: индексный регистр CPUID (CPUID) на странице 208
0xE000ED04	ICSR	RW ⁽¹⁾	Привилегированный	0x00000000	Таблица 4.4.3: управление Прерыванием и регистр состояния (ICSR) на странице 210
0xE000ED08	VTOR	RW	Привилегированный	0x00000000	Таблица 4.4.4: регистр смещения Таблицы векторов (VTOR) на странице 212
0xE000ED0C	AIRCR	RW ⁽¹⁾	Привилегированный	0xFA050000	Таблица 4.4.5: прерывание Приложения и регистр команд сброса (AIRCR) на странице 212
0xE000ED10	SCR	RW	Привилегированный	0x00000000	Таблица 4.4.6: Системный регистр команд (SCR) на странице 214
0xE000ED14	CCR	RW	Привилегированный	0x00000200	Таблица 4.4.7: Конфигурация и регистр команд (CCR) на странице 215
0xE000ED18	SHPR1	RW	Привилегированный	0x00000000	Таблица 4.4.8: Системные приоритетные регистры обработчика (SHPRx) на странице 217
0xE000ED1C	SHPR2	RW	Привилегированный	0x00000000	
0xE000ED20	SHPR3	RW	Привилегированный	0x00000000	
0xE000ED24	SHCRS	RW	Привилегированный	0x00000000	Таблица 4.4.9: Системное управление обработчиком и регистр состояния (SHCSR) на странице 219
0xE000ED28	CFSR	RW	Привилегированный	0x00000000	Таблица 4.4.10: Конфигурируемый регистр состояния отказа (CFSR; UFSR+BFSR+MMFSR) на странице 221
0xE000ED28	MMSR ⁽²⁾	RW	Привилегированный	0x00	Таблица 4.4.10 Регистра Состояния Отказа MemManage на странице 221
0xE000ED29	BFSR ⁽²⁾	RW	Привилегированный	0x00	Таблица 4.4.10 Регистра Состояния BusFault на странице 221
0xE000ED2A	UFSR ⁽²⁾	RW	Привилегированный	0x0000	Таблица 4.4.10 Регистра Состояния UsageFault на странице 221
0xE000ED2C	HFSR	RW	Привилегированный	0x00000000	Таблица 4.4.14: Твердый регистр состояния отказа (HFSR) на странице 225
0xE000ED34	MMAR	RW	Привилегированный	Unknown	Таблица 4.4.15: управление Памятью дает сбой регистр адреса (MMFAR) на странице 226
0xE000ED38	BFAR	RW	Привилегированный	Unknown	Таблица 4.4.16: отказ Шины адресует регистр (BFAR) на странице 226
0xE000ED3C	AFSR	RW	Привилегированный	0x00000000	Таблица 4.4.17: Вспомогательный регистр состояния отказа (AFSR) на странице 227

1. См. описание регистра для получения дополнительной информации. 2. Подрегистр CFBSR