# **Описание используемых функций**

## **HAL\_DMA\_WriteDC**

**Резюме**

**void** **HAL\_DMA\_WriteDC**( uint32\_t ch\_number, **void** \*qw\_tcb )

uint32\_t ch\_number – номер канала DMA (с 4 по 13)

**void** \*qw\_tcb – указатель на конфигурацию для регистра TCB, которая должна быть выровнена на границу квадрослова

**Описание**

Функция инициализирует выбранный канал DMA, значениями, лежащими по указателю \*qw\_tcb (должен указывать на квадрослово, хранящее значения для инициализации DMA).

**Пример 1**

**#include <hal\_1967VN044.h>**

**#define** ADDRESS\_MEMORY 0x40000

**\_\_builtin\_quad** tcb;

**int** **main**(**void**)

{

**unsigned int** \*ptr;

ptr = (unsigned int \*)&tcb;

\*(ptr + 0) = ADDRESS\_MEMORY;

\*(ptr + 1) = (1024<<16) | 4;

\*(ptr + 2) = 0;

\*(ptr + 3) = TCB\_INTMEM | TCB\_QUAD;

**HAL\_DMA\_WriteDC** (4, &tcb);

**return** 0;

}

**Пример 2**

**#include <hal\_1967VN044.h>**

**#define** ADDRESS\_MEMORY 0x40000

**unsigned int \_\_attribute**((aligned(4 \* (32/\_\_CHAR\_BIT\_\_)))) tcb[4];

**int** **main**(**void**)

{

tcb[0] = ADDRESS\_MEMORY;

tcb[1] = (1024<<16) | 4;

tcb[2] = 0;

tcb[3] = TCB\_INTMEM | TCB\_QUAD;

**HAL\_DMA\_WriteDC** (5, &tcb);

**return** 0;

}

## **HAL\_DMA\_WriteDCA**

**Резюме**

**void** **HAL\_DMA\_WriteDCA**( uint32\_t ch\_number, uint64\_t \*data )

uint32\_t ch\_number – номер канала DMA (с 4 по 13)

uint64\_t \*data – указатель на конфигурацию для регистра DCA, которая должна быть выровнена на границу двойного слова

**Описание**

Функция инициализирует регистр DCA выбранного канала DMA значениями, лежащими по указателю \*data. Младшие 32 бита задают адрес периферийного устройства, старшие 3 бита определяют расположение устройства в памяти и активируют регистр DCA:

0x2000000000000000ULL – регистр расположен во внутренней памяти

0x4000000000000000ULL – регистр расположен в внешней памяти

**Пример**

**#include <hal\_1967VN044.h>**

**#define** TIMER\_CCR1\_offset 0x4

**int** **main**(**void**)

{

uint64\_t ullData;

// С помомщью регистра DCA указываем загружать новые данные в регистр CCR1 таймера 1

ullData = DMA\_DCA\_INT\_MEM | ( base\_TIMER1 + TIMER\_CCR1\_offset );

HAL\_DMA\_WriteDCA( 5, &ullData );

}

## **HAL\_DMA\_WriteDCS**

## **HAL\_DMA\_WriteDCD**

**Резюме**

**void** **HAL\_DMA\_WriteDCS**( uint32\_t ch\_number, **void** \*qw\_tcb )

**void** **HAL\_DMA\_WriteDCD** ( uint32\_t ch\_number, **void** \*qw\_tcb )

uint32\_t ch\_number – номер канала DMA общего назначения(с 0 по 3)

**void** \*qw\_tcb – указатель на конфигурацию для регистра TCB, которая должна быть выровнена на границу квадрослова

**Описание**

Функция инициализирует передатчик/приемник выбранного канал DMA общего назначения значениями, лежащими по указателю \*qw\_tcb (должен указывать на квадрослово хранящее значения для инициализации DMA).

**Пример 1**

**#include <hal\_1967VN044.h>**

**#define** ADDRESS\_MEMORY\_TX 0x40000000

**#define** ADDRESS\_MEMORY\_RX 0x00040000

**\_\_builtin\_quad** tcb\_rx, tcb\_tx;

**int** **main**(**void**)

{

**unsigned int** \*ptr;

ptr = (unsigned int \*)&tcb\_rx;

\*(ptr + 0) = ADDRESS\_MEMORY\_RX;

\*(ptr + 1) = (1024<<16) | 4;

\*(ptr + 2) = 0;

\*(ptr + 3) = TCB\_INTMEM | TCB\_QUAD;

ptr = (unsigned int \*)&tcb\_tx;

\*(ptr + 0) = ADDRESS\_MEMORY\_TX;

\*(ptr + 1) = (1024<<16) | 4;

\*(ptr + 2) = 0;

\*(ptr + 3) = TCB\_EXTMEM | TCB\_QUAD;

**HAL\_DMA\_WriteDCD** (0, &tcb\_rx);

**HAL\_DMA\_WriteDCS** (0, &tcb\_tx);

**return** 0;

}

**Пример 2**

**#include <hal\_1967VN044.h>**

**#define** ADDRESS\_MEMORY\_TX 0x40000000

**#define** ADDRESS\_MEMORY\_RX 0x00040000

**unsigned int \_\_attribute**((aligned(4 \* (32/\_\_CHAR\_BIT\_\_)))) tcb\_rx[4], tcb\_tx[4];

**int** **main**(**void**)

{

tcb\_rx[0] = ADDRESS\_MEMORY\_RX;

tcb\_rx[1] = (1024<<16) | 4;

tcb\_rx[2] = 0;

tcb\_rx[3] = TCB\_INTMEM | TCB\_QUAD;

tcb\_tx[0] = ADDRESS\_MEMORY\_TX;

tcb\_tx[1] = (1024<<16) | 4;

tcb\_tx[2] = 0;

tcb\_tx[3] = TCB\_EXTMEM | TCB\_QUAD;

**HAL\_DMA\_WriteDCD** (3, &tcb\_rx);

**HAL\_DMA\_WriteDCS** (3, &tcb\_tx);

**return** 0;

}

## **HAL\_DMA\_ReadDC**

**Резюме**

**void** **HAL\_DMA\_ReadDC**( uint32\_t ch\_number, \_\_builtin\_quad \*qw\_tcb )

uint32\_t ch\_number – номер канала DMA (с 4 по 13)

\_\_builtin\_quad \*qw\_tcb – указатель на место сохранения регистра TCB

**Описание**

Функция сохраняет текущее состояния регистра TCB выбранного канала DMA по указателю (который должен указывать на квадрослово).

**Пример 1**

**#include <hal\_1967VN044.h>**

**#define** ADDRESS\_MEMORY 0x40000

**\_\_builtin\_quad** tcb;

**int** **main**(**void**)

{

**HAL\_DMA\_ReadDC** (9, &tcb);

**return** 0;

}

## **HAL\_DMA\_ReadDCA**

**Резюме**

**void** **HAL\_DMA\_ReadDCA**( uint32\_t ch\_number, uint64\_t \*data )

uint32\_t ch\_number – номер канала DMA (с 4 по 13)

uint64\_t \*data – указатель на место сохранения регистра DCA

**Описание**

Функция сохраняет текущее состояния регистра DCA выбранного канала DMA по указателю.

## **HAL\_DMA\_ReadDCS**

## **HAL\_DMA\_ReadDCD**

**Резюме**

**void** **HAL\_DMA\_ReadDCS**( uint32\_t ch\_number, \_\_builtin\_quad \*qw\_tcb )

**void** **HAL\_DMA\_ReadDCD**( uint32\_t ch\_number, \_\_builtin\_quad \*qw\_tcb )

uint32\_t ch\_number – номер канала DMA общего назначения (с 0 по 3)

\_\_builtin\_quad \*qw\_tcb – указатель на место сохранения регистра TCB

**Описание**

Функция сохраняет текущее состояния регистра TCB выбранного канала DMA общего назначения по указателю (который должен указывать на квадрослово).

**Пример 1**

**#include <hal\_1967VN044.h>**

**#define** ADDRESS\_MEMORY 0x40000

**\_\_builtin\_quad** tcb;

**int** **main**(**void**)

{

**HAL\_DMA\_ReadDCS** (1, &tcb);

**HAL\_DMA\_ReadDCD** (2, &tcb);

**return** 0;

}

## **HAL\_DMA\_Stop**

**Резюме**

**void** **HAL\_DMA\_Stop**( uint32\_t ch\_number )

uint32\_t ch\_number – номер канала DMA (с 0 по 13)

**Описание**

Функция останавливает (инициализирует нулями регистр TCB) выбранный канал DMA.

**Пример 1**

**#include <hal\_1967VN044.h>**

**int** **main**(**void**)

{

**HAL\_DMA\_Stop**(11);

**return** 0;

}

## **HAL\_DMA\_GetChannelStatus**

**Резюме**

uint32\_t **HAL\_DMA\_GetChannelStatus**( uint32\_t channel )

uint32\_t channel – номер канала DMA (с 0 по 13)

**Описание**

Функция возвращает статус заданного канала DMA.

* 0 – выключен;
* 1 – активен (работает);
* 2 – завершил работу;
* 3 – резерв;
* 4 – ошибка. Инициализация активного канала;
* 5 – ошибка. Запрещенная конфигурация канала;
* 6 – резерв;
* 7 – ошибка. Некорректный адрес.

**Пример 1**

**#include <hal\_1967VN044.h>**

**int** **main**(**void**)

{

**int** chStat;

chStat = **HAL\_DMA\_GetChannelStatus**(7);

**return** 0;

}

## **HAL\_DMA\_GetChannelStatusClear**

**Резюме**

uint32\_t **HAL\_DMA\_GetChannelStatusClear**( uint32\_t channel )

uint32\_t channel – номер канала DMA (с 0 по 13)

**Описание**

Функция возвращает статус заданного канала DMA, после чего очищает регистр статуса DMA.

* 0 – выключен;
* 1 – активен (работает);
* 2 – завершил работу;
* 3 – резерв;
* 4 – ошибка. Инициализация активного канала;
* 5 – ошибка. Запрещенная конфигурация канала;
* 6 – резерв;
* 7 – ошибка. Некорректный адрес.

**Пример 1**

**#include <hal\_1967VN044.h>**

**int** **main**(**void**)

{

**int** chStat;

chStat = **HAL\_DMA\_GetChannelStatusClear**(6);

**return** 0;

}

## **HAL\_DMA\_WaitForChannel**

**Резюме**

uint32\_t **HAL\_DMA\_WaitForChannel**( uint32\_t channel )

uint32\_t channel – номер канала DMA (с 0 по 13)

**Описание**

Функция исполняется пока идет прием/передача по выбранному каналу DMA. Возвращает следующие значения:

0 – канал завершил работу успешно

1 – обнаружена ошибка

2 – канал выключен

**Пример 1**

**#include <hal\_1967VN044.h>**

**#define** ADDRESS\_MEMORY 0x40000

**unsigned int \_\_attribute**((aligned(4 \* (32/\_\_CHAR\_BIT\_\_)))) tcb[4];

**int** **main**(**void**)

{

**int** errFlag;

tcb[0] = ADDRESS\_MEMORY;

tcb[1] = (1024<<16) | 4;

tcb[2] = 0;

tcb[3] = TCB\_INTMEM | TCB\_QUAD;

**HAL\_DMA\_WriteDC** (7, &tcb);

errFlag = **HAL\_DMA\_WaitForChannel** (7);

**return** 0;

}

## **HAL\_DMA\_GetDcCountX**

**Резюме**

uint32\_t **HAL\_DMA\_GetDcCountX**( uint32\_t ch\_number )

uint32\_t ch\_number – номер канала DMA (с 4 по 13)

**Описание**

Функция принимает номер канала DMA и возвращает, значение счетчика передаваемых/принимаемых данных этого канала

**Пример 1**

**#include <hal\_1967VN044.h>**

**#define** ADDRESS\_MEMORY 0x40000

**unsigned int \_\_attribute**((aligned(4 \* (32/\_\_CHAR\_BIT\_\_)))) tcb[4];

**int** **main**(**void**)

{

**unsigned int** DC;

tcb[0] = ADDRESS\_MEMORY;

tcb[1] = (1024<<16) | 4;

tcb[2] = 0;

tcb[3] = TCB\_INTMEM | TCB\_QUAD;

**HAL\_DMA\_WriteDC** (7, &tcb);

DC = **HAL\_DMA\_GetDcCountX** (6);

**return** 0;

}

## **HAL\_DMA\_CreateChannelDest**

**Резюме**

**void** **HAL\_DMA\_CreateChannelDest**( uint32\_t channel, **void** \*tcb\_current, **void** \*tcb\_next )

uint32\_t channel – номер канала DMA. Задает следующий канал в цепочки DMA, для каналов с 4 по 11. Для каналов с 0 по 3, этот параметр не имеет значения.

**void** \*tcb\_current – указатель на текущую конфигурацию для регистров TCB, которая должна быть выровнена на границу квадрослова

**void** \*tcb\_next – указатель на следующую конфигурации для регистров TCB, которая должна быть выровнена на границу квадрослова

**Описание**

Функция принимает номер канала DMA, который будет следующим в цепочке. Указатель на текущую и следующую конфигурацию регистров TCB. Функция вызывается ДО записи первой конфигурации в цепочки в регистры TCB. Рассмотрим несколько примеров:

**Пример 1**

// Бесконечная передача по DMA одного и того же участка памяти в периферию по //каналу 4

**#include <hal\_1967VN044.h>**

**#define** ADDRESS\_MEMORY 0x40000

**unsigned int \_\_attribute**((aligned(4 \* (32/\_\_CHAR\_BIT\_\_)))) tcb[4];

**int** **main**(**void**)

{

tcb[0] = ADDRESS\_MEMORY;

tcb[1] = (1024<<16) | 4;

tcb[2] = 0;

tcb[3] = TCB\_INTMEM | TCB\_QUAD;

**HAL\_DMA\_CreateChannelDest**(4, &tcb, &tcb);

**HAL\_DMA\_WriteDC** (4, &tcb);

**return** 0;

}

**Пример 2**

// Два элемента в цепочка DMA зацикленные друг на друга (то есть сначала работает //tcb1 по 4 каналу, затем tcb2 по 5 каналу)

**#include <hal\_1967VN044.h>**

**#define** ADDRESS\_MEMORY 0x40000

**unsigned int \_\_attribute**((aligned(4 \* (32/\_\_CHAR\_BIT\_\_)))) tcb1[4], tcb2[4];

**int** **main**(**void**)

{

tcb1[0] = ADDRESS\_MEMORY;

tcb1[1] = (1024<<16) | 4;

tcb1[2] = 0;

tcb1[3] = TCB\_INTMEM | TCB\_QUAD;

tcb2[0] = ADDRESS\_MEMORY+0x40000;

tcb2[1] = (1024<<16) | 4;

tcb2[2] = 0;

tcb2[3] = TCB\_INTMEM | TCB\_QUAD;

**HAL\_DMA\_CreateChannelDest**(5, &tcb1, &tcb2);//после завершения передачи //данных по каналу с конфигурацией tcb1, начнётся передача данных по каналу 5, с //конфигурацией tcb2

**HAL\_DMA\_CreateChannelDest**(4, &tcb2, &tcb1); //после завершения передачи //данных по каналу с конфигурацией tcb2, начнётся передача данных по каналу 4, с //конфигурацией tcb1

**HAL\_DMA\_WriteDC** (4, &tcb1); //Включение DMA 4 (таким образом он //становится первым каналом в цепочке), после завершения работы канала 4, начнет //работать канал 5 с конфигурацией tcb2, затем после завершения работы канала 5, //начнет работать канал 4 и т.д.

**return** 0;

}

**Пример 3**

// Три элемента в цепочке, для каналов 0-3, для данной ситуации возьмем канал //0.

**#define** MLT 16

**int \_\_attribute**((aligned(4 \* (32/\_\_CHAR\_BIT\_\_)))) tcbTx1[4], tcbTx2[4], tcbTx3[4], tcbRx[4];

**int \_\_attribute**((aligned(4 \* (32/\_\_CHAR\_BIT\_\_)))) color[MLT][MLT] , color1[MLT][MLT] , color2[MLT][MLT] , color3[MLT][MLT];

**int** **main**(**void**)

{

tcbRx[0] = &color;

tcbRx[1] = 4\*MLT<<16 | 0;

tcbRx[2] = 4\*MLT<<16;

tcbRx[3] = TCB\_INTMEM | TCB\_QUAD | TCB\_TWODIM | 0;

tcbTx1[0] = &color1;

tcbTx1[1] = 4\*MLT<<16 | 0;

tcbTx1[2] = 4\*MLT<<16;

tcbTx1[3] = TCB\_INTMEM | TCB\_QUAD | TCB\_TWODIM | 0;

tcbTx2[0] = &color2;

tcbTx2[1] = 4\*MLT<<16 | 0;

tcbTx2[2] = 4\*MLT<<16;

tcbTx2[3] = TCB\_INTMEM | TCB\_QUAD | TCB\_TWODIM | 0;

tcbTx3[0] = &color3;

tcbTx3[1] = 4\*MLT<<16 | 0;

tcbTx3[2] = 4\*MLT<<16;

tcbTx3[3] = TCB\_INTMEM | TCB\_QUAD | TCB\_TWODIM | 0;

HAL\_DMA\_CreateChannelDest(0, &tcbRx, &tcbRx); //Цепочка для приемника – //приемник бесконечно работает по конфигурации tcbRx

HAL\_DMA\_CreateChannelDest(1, &tcbTx1, &tcbTx2); //Цепочка tcbTx1 -> tcbTx2

HAL\_DMA\_CreateChannelDest(2, &tcbTx2, &tcbTx3); //Цепочка tcbTx2 -> tcbTx3

HAL\_DMA\_CreateChannelDest(3, &tcbTx3, &tcbTx1); //Цепочка tcbTx3 -> tcbTx1

//Обратите внимание, что при создании цепочки для каналов 0-3, входной //параметр ch\_number не имеет значения, т.к каналы 0-3 всегда переконфигурируют //новым элементом цепочки ТОЛЬКО самих себя. То есть канал, с которым будут работать //конфигурации, задается функциями HAL\_DMA\_WriteDCD и HAL\_DMA\_WriteDCS

HAL\_DMA\_WriteDCD(0,&tcbRx);// Включение приемника канал 0

HAL\_DMA\_WriteDCS(0,&tcbTx1);// Включение передатчика – канал 0, конфигурация //tcbTx1(т.е. tcbTx1 становится первым элементом в цепочке)

}

## **HAL\_DMA\_MemCopy32**

**Резюме**

DMA\_Return\_type **HAL\_DMA\_MemCopy32**( uint32\_t ch\_number, **const** **void** \*src, **const** **void** \*dst, uint32\_t size )

uint32\_t ch\_number – номер канала DMA общего назначения (с 0 по 3)

**const** **void** \*src – адрес передатчика

**const** **void** \*dst – адрес приемника

uint32\_t size – количество передаваемых слов

**Описание**

Функция передает массив по выбранному каналу DMA общего назначения. Передача производится словами, функции прерывания, цепочки, высокого приоритета, внешних запросов и двумерной передачи – отключены. Указывать можно как внешний, так и внутренней адрес передаваемого/принимаемого массива, исходя из этого функция автоматически определит режим передачи.

Функция возвращает код ошибки:

**Функция возвращает код ошибки:**

Таблица 1.3 Ошибки

|  |  |
| --- | --- |
| **Код ошибки** | **Описание** |
| 0 (dmaCopyValid) | Ошибок нет. Инициализация успешна. |
| 1 (dmaCopyAddrErr) | Некорректный адрес передаваемого/принимаемого массива |
| 2 (dmaCopyLengthErr) | Длинна передаваемых данных превышает максимальное значение |
| 3 (dmaCopyChNumErr) | Неверный канал DMA (разрешены канал с 0 по 3) |
| 4 (dmaCopyInitActCh) | Инициализация активного канала |
| 5 (dmaCopyCfgErr) | Запрещенная конфигурация канала |

**Ограничения**

size – не должно превышать значение 65535

**Пример**

//Передача по 2-му каналу DMA, массива data\_tx32, находящего во внутренней //памяти, по адресу массива data\_rx32, находящегося во внешней памяти

**#include <hal\_1967VN044.h>**

**#define** N 1024

**int** data\_tx32[N];

**\_\_attribute\_\_**((section(".sdr\_data")))

**int** data\_rx32[N];

**int** **main**(**void**)

{

**int** errFlag;

errFlag = HAL\_DMA\_MemCopy32 (2, &data\_tx32, &data\_rx32, N);

**return** 0;

}

## **HAL\_DMA\_RqstSet**

**Резюме**

**void** **HAL\_DMA\_RqstSet**( uint32\_t ch\_number, DMA\_Requester\_type   
dmaRqster )

uint32\_t ch\_number – номер канала DMA (с 0 по 11)

DMA\_Requester\_type dmaRqster – источник запросов

|  |  |
| --- | --- |
| **Источник** | **Значение** |
| Стандартный режим:  0 – nDMAR0  1 – nDMAR1  2 – nDMAR2  3 – nDMAR3  4 – LINK 0  5 – LINK 1  6, 7 –  8 – LINK 0  9 – LINK 1  10, 11 – | *dmaSTD* |
| UART0 | *dmaUART0* |
| UART1 | *dmaUART1* |
| SPI0 | *dmaSPI0* |
| LCD для канала-передатчика  VIDEO CAMERA для канала-приемника | *dmaVIDEO* |
| SSI0 | *dmaSSI0* |
| SSI1 | *dmaSSI1* |
| NAND Flash | *dmaNAND* |
| UP DOWN конвертор 1 (ADDA1) | *dmaUPDOWN1* |
| UP DOWN конвертор 0 (ADDA0) | *dmaUPDOWN0* |
| UP DOWN конвертор 2 (ADDA2) | *dmaUPDOWN2* |
| UP DOWN конвертор 3 (ADDA3) | *dmaUPDOWN3* |
| SPI1 | *dmaSPI1* |
| SPI2 | *dmaSPI2* |
| Н264\_RQ1 | *dmaH264RQ1* |
| Н264\_RQ0 | *dmaH264RQ0* |
| Внешний запрос nDMAR[4] (каналы 4-7) или nDMAR[8] (каналы 8-11) | *dmaDMAR\_4\_8* |
| Внешний запрос nDMAR[5] (каналы 4-7) или nDMAR[9] (каналы 8-11) | *dmaDMAR\_5\_9* |
| Внешний запрос nDMAR[6] (каналы 4-7) или nDMAR[10] (каналы 8-11) | *dmaDMAR\_6\_10* |
| Внешний запрос nDMAR[7] (каналы 4-7) или nDMAR[11] (каналы 8-11) | *dmaDMAR\_7\_11* |
| Запрос REQ от таймера 0 с функцией Захвата/ШИМ | *dmaGTIMER0\_REQ* |
| Запрос REQ1 от таймера 0 с функцией Захвата/ШИМ | *dmaGTIMER0\_REQ1* |
| Запрос REQ2 от таймера 0 с функцией Захвата/ШИМ | *dmaGTIMER0\_REQ2* |
| Запрос REQ3 от таймера 0 с функцией Захвата/ШИМ | *dmaGTIMER0\_REQ3* |
| Запрос REQ4 от таймера 0 с функцией Захвата/ШИМ | *dmaGTIMER0\_REQ4* |
| Запрос REQ от таймера 1 с функцией Захвата/ШИМ | *dmaGTIMER1\_REQ* |
| Запрос REQ1 от таймера 1 с функцией Захвата/ШИМ | *dmaGTIMER1\_REQ1* |
| Запрос REQ2 от таймера 1 с функцией Захвата/ШИМ | *dmaGTIMER1\_REQ2* |
| Запрос REQ3 от таймера 1 с функцией Захвата/ШИМ | *dmaGTIMER1\_REQ3* |
| Запрос REQ4 от таймера 1 с функцией Захвата/ШИМ | *dmaGTIMER1\_REQ4* |
| Запрос от таймера 0 контроллера прерываний | *dmaTIMER0* |
| Запрос от таймера 1 контроллера прерываний | *dmaTIMER1* |

**Описание**

Функция устанавливает для выбранного канала-передатчика DMA (с 0 по 11), источников запросов (т.е выбирается периферия с который будет работать выбранный DMA канал, в соответствии с параметром dmaRqster).

**Пример**

**#include <hal\_1967VN044.h>**

**int** **main**(**void**)

{

HAL\_DMA\_RqstSet(4,*dmaSSI0*); //Канал DMA 4 теперь работает с SSI0

}

## **HAL\_DMA\_RqstCl**

**Резюме**

**void** **HAL\_DMA\_RqstClr**( uint32\_t ch\_number )

uint32\_t ch\_number – номер канала DMA (с 0 по 11)

**Описание**

Функция устанавливает стандартный режим работы, для выбранного канала DMA.

**Пример**

**#include <hal\_1967VN044.h>**

**int** **main**(**void**)

{

HAL\_DMA\_RqstCl(5); //Канал DMA 5 теперь работает в стандартном режиме

}

## **HAL\_DMA\_SourceDestRqstReset**

**Резюме**

**void** **HAL\_DMA\_SourceDestRqstReset**( **void** )

**Описание**

Функция устанавливает стандартный режим работы, все каналы DMA.

**Пример**

**#include <hal\_1967VN044.h>**

**int** **main**(**void**)

{

HAL\_DMA\_SourceDestRqstReset(); //Все каналы DMA теперь работают в стандартном режиме

}

## **HAL\_DMA\_PrimaryPriority**

**Резюме**

**void** **HAL\_DMA\_PrimaryPriority**( **void** )

**Описание**

Функция разрешает высокоприоритетному каналу DMA иметь преимущество перед процессором при доступе к внешней памяти. Есть так же обратная функция (устанавливает ядру приоритет перед DMA) HAL\_Core\_PrimaryPriority.

**Пример**

**#include <hal\_1967VN044.h>**

**int** **main**(**void**)

{

HAL\_DMA\_PrimaryPriority ();

}