# Описание используемых типов

## UPDOWN\_Init\_type

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Поле** | **Описание** | **Тип** | **Значения** | |
|  | Mode | Режим работы модуля | UPDOWN\_Mode | *UPDOWN\_Mode\_Down* | децимация (DOWN-режим) |
| *UPDOWN\_Mode\_Up* | интерполяция (UP-режим) |
|  | LinkX | Link-порт для приема/выдачи данных | UPDOWN\_Link | *UPDOWN\_Link\_Test* | APB-шина (только для DOWN) |
| *UPDOWN\_Link\_0* | Link-порт № 0 |
| *UPDOWN\_Link\_1* | Link-порт № 1 |
|  | FiltersStage | Количество ступеней фильтров | UPDOWN\_FiltersStage | *UPDOWN\_FiltersStage\_3* | 3 ступени |
| *UPDOWN\_FiltersStage\_5* | 5 ступеней |
| *UPDOWN\_FiltersStage\_7* | 7 ступеней |
|  | Round | Режим округления данных | UPDOWN\_Round | *UPDOWN\_Round\_Off* | выключено |
| *UPDOWN\_Round\_MultOnly* | вкл. только после перемножения |
| *UPDOWN\_Round\_OutOnly* | вкл. только для выходного результата |
| *UPDOWN\_Round\_OutOnlyNoOverflow* | вкл. только для выходного результата с ограничением переполнения |
| *UPDOWN\_Round\_All* | вкл. после перемножения и для выходного результата |
| *UPDOWN\_Round\_AllNoOverflow* | вкл. после перемножения и для выходного результата с ограничением переполнения |
|  | Shift | сдвиг выходных данных | uint8\_t | От 0 до 255 | |
|  | FifoMode | Режим хранения данных в FIFO | UPDOWN\_FifoMode | *UPDOWN\_FifoMode\_LowWord\_IQ* | Первое слово хранится в младших битах регистра данных DR, мнимая часть хранится в битах [31:16], действительная часть в битах [15:0] слова. |
| *UPDOWN\_FifoMode\_LowWord\_QI* | Первое слово хранится в младших битах регистра данных DR, мнимая часть хранится в битах [15:0], действительная часть в битах [31:16] слова. |
| *UPDOWN\_FifoMode\_HighWord\_IQ* | Первое слово хранится в старших битах регистра данных DR, мнимая часть хранится в битах [31:16], действительная часть в битах [15:0] слова. |
| *UPDOWN\_FifoMode\_HighWord\_QI* | Первое слово хранится в старших битах регистра данных DR, мнимая часть хранится в битах [15:0], действительная часть в битах [31:16] слова. |
|  | Coefficient |  | uint16\_t | От 4 до 513 | |
|  | CntMode | Режим работы счетчика отсчетов | UPDOWN\_CntMode | *UPDOWN\_CntMode\_Off* | Выключен |
| *UPDOWN\_CntMode\_Run* | Включен |
| *UPDOWN\_CntMode\_IT* | Включен и разрешено прерывание |
|  | Cnt | Значение счетчика отсчетов | uint32\_t | От 0 до 232 – 1 | |
|  | Step | Шаг приращения счетчика для выбора констант | uint32\_t | От 0 до 216 – 1 | |

# Описание используемых функций

## HAL\_UPDOWN\_DefaultInitStruct

**Резюме**

**void** **HAL\_UPDOWN\_DefaultInitStruct**( UPDOWN\_Init\_type \*Init )

UPDOWN\_Init\_type \*Init – указатель на структуру с настройками

**Описание**

Функция заполняет значениями по умолчанию структуру инициализации UPDOWN\_Init\_type \*Init модуля UP/DOWN.

При вызове функции в поля записываются следующие значения:

Mode = *UPDOWN\_Mode\_Down*;

LinkX = *UPDOWN\_Link\_Test*;

FiltersStage = *UPDOWN\_FiltersStage\_3*;

Round = *UPDOWN\_Round\_Off*;

Shift = 0xFF;

FifoMode = *UPDOWN\_FifoMode\_LowWord\_IQ*;

Coefficient = 4;

CntMode = *UPDOWN\_CntMode\_Off*;

Step = 0;

Cnt = 0.

**Пример**

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

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

{

UPDOWN\_InitTypeDef UD;

HAL\_UPDOWN\_DefaultInitStruct (&UD);

**while** (1)

{

}

}

## HAL\_UPDOWN\_Init

**Резюме**

**void** **HAL\_UPDOWN\_Init**( UPDOWN\_type \* **const** LX\_UPDOWNx, UPDOWN\_Init\_type \* **const** Init )

UPDOWN\_type \* **const** LX\_UPDOWNx – указатель на структуру регистров UPDOWN

UPDOWN\_Init\_type \* **const** Init – указатель на структуру с настройками

**Описание**

Функция инициализирует регистры CR, RCNT и STEP модуля UPDOWN. Выбор модуля осуществляется с помощью входного аргумента UPDOWN\_type \* **const** LX\_UPDOWNx, значения инициализации определяются входным аргументом UPDOWN\_Init\_type \* **const** Init.

Входной аргумент выбора модуля LX\_UPDOWNx может принимать следующие значения:

LX\_UPDOWN0 – модуль 0,

LX\_UPDOWN1 – модуль 1,

LX\_UPDOWN2 – модуль 2,

LX\_UPDOWN3 – модуль 3.

При инициализации значение поля Kdelay регистра CR определяется полем Coefficient структуры Init. Если Coefficient ≤ 4, то в Kdelay записывается 1, если Coefficient ≥ 514, то в Kdelay записывается 511. В остальных случаях Kdelay = (Coefficient >> 1) – 1.

При инициализации значение поля SHFR регистра CR определяется полем Shift структуры Init. Если Shift < 128, то он копируется в поле SHFR. Если Shift ≥ 128, то SHFR = FLEN·log2((Kdelay + 1) · 2).

Значения регистров RCNT и STEP модуля устанавливаются одновременно.

**Пример**

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

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

{

UPDOWN\_Init\_type UD;

HAL\_UPDOWN\_DefaultInitStruct (&UD);

UD.LinkX = *UPDOWN\_Link\_Test*;

HAL\_UPDOWN\_Init (LX\_UPDOWN0, &UD);

**while** (1)

{

}

}

## HAL\_UPDOWN\_DeInit

**Резюме**

**void** **HAL\_UPDOWN\_DeInit**( UPDOWN\_type \* **const** LX\_UPDOWNx )

UPDOWN\_type \* **const** LX\_UPDOWNx – указатель на структуру регистров UPDOWN

**Описание**

Функция деинициализации модуля UPDOWN. Выбор модуля осуществляется входным аргументом UPDOWN\_type \* **const** LX\_UPDOWNx.

Функция сбрасывает в 0 регистры CR, STEP и RCNT модуля UPDOWN. Допустимые значения входного аргумента LX\_UPDOWNx, определяющего выбор модуля, описаны в функции **HAL\_UPDOWN\_Init(…)**.

**Пример**

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

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

{

UPDOWN\_Init\_type UD;

HAL\_UPDOWN\_DefaultInitStruct (&UD);

HAL\_UPDOWN\_Init (LX\_UPDOWN0, &UD); // модуль включен и настроен по умолчанию

HAL\_UPDOWN\_DeInit (LX\_UPDOWN0); // модуль выключен и настройки сброшены

**while** (1)

{

}

}

## HAL\_UPDOWN\_Enable

**Резюме**

**void** **HAL\_UPDOWN\_Enable**( UPDOWN\_type \* **const** LX\_UPDOWNx )

UPDOWN\_type \* **const** LX\_UPDOWNx – указатель на структуру регистров UPDOWN

**Описание**

Функция устанавливает 1 в поле EN регистра CR модуля UPDOWN. Выбор модуля осуществляется входным аргументом UPDOWN\_type \* **const** LX\_UPDOWNx, допустимые значения которого описаны в функции **HAL\_UPDOWN\_Init(…)**.

**Пример**

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

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

{

UPDOWN\_Init\_type UD;

HAL\_UPDOWN\_DefaultInitStruct (&UD);

HAL\_UPDOWN\_Init (LX\_UPDOWN0, &UD);

HAL\_UPDOWN\_Disable(LX\_UPDOWN0); // модуль выключен

HAL\_UPDOWN\_SetRcntStep (LX\_UPDOWN0, 128, 10);

HAL\_UPDOWN\_Enable(LX\_UPDOWN0); // модуль включен

**while** (1)

{

}

}

## HAL\_UPDOWN\_Disable

**Резюме**

**void** **HAL\_UPDOWN\_Disable**( UPDOWN\_type \* **const** LX\_UPDOWNx )

UPDOWN\_type \* **const** LX\_UPDOWNx – указатель на структуру регистров UPDOWN

**Описание**

Функция записывает 0 в поле EN регистра CR модуля UPDOWN. Выбор модуля осуществляется входным аргументом UPDOWN\_type \* **const** LX\_UPDOWNx, допустимые значения которого описаны в функции **HAL\_UPDOWN\_Init(…)**.

**Пример**

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

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

{

UPDOWN\_Init\_type UD;

HAL\_UPDOWN\_DefaultInitStruct (&UD);

HAL\_UPDOWN\_Init (LX\_UPDOWN0, &UD);

HAL\_UPDOWN\_Disable(LX\_UPDOWN0); // модуль выключен

HAL\_UPDOWN\_SetRcntStep (LX\_UPDOWN0, 128, 10);

HAL\_UPDOWN\_Enable(LX\_UPDOWN0); // модуль включен

**while** (1)

{

}

}

## HAL\_SetRcntStep

**Резюме**

**void** **HAL\_SetRcntStep**( UPDOWN\_type \* **const** LX\_UPDOWNx, **const** uint32\_t RCNT, **const** uint16\_t Step )

UPDOWN\_type \* **const** LX\_UPDOWNx – указатель на структуру регистров UPDOWN

**const** uint32\_t RCNT – значение счетчика отсчетов

**const** uint16\_t Step – значение шага

**Описание**

Функция одновременной установки регистров RCNT и STEP модуля UPDOWN\_type \* **const** LX\_UPDOWNx.

Выбор модуля осуществляется входным аргументом UPDOWN\_type \* **const** LX\_UPDOWNx, допустимые значения которого описаны в функции **HAL\_UPDOWN\_Init(…)**.

Значение регистра RCNT будет соответствовать значению входного аргумента **const** uint32\_t RCNT.

Значение регистра STEP будет соответствовать значению входного аргумента **const** uint16\_t Step.

**Пример**

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

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

{

UPDOWN\_Init\_type UD;

HAL\_UPDOWN\_DefaultInitStruct (&UD);

HAL\_UPDOWN\_Init (LX\_UPDOWN0, &UD);

HAL\_UPDOWN\_Disable(LX\_UPDOWN0);

HAL\_UPDOWN\_SetRcntStep (LX\_UPDOWN0, 128, 10); // RCNT = 128, STEP = 10

HAL\_UPDOWN\_Enable(LX\_UPDOWN0);

**while** (1) {

}

}

## HAL\_UPDOWN\_GetFlag

**Резюме**

UPDOWN\_FlagState\_type **HAL\_UPDOWN\_GetFlag**( UPDOWN\_type \* **const** LX\_UPDOWNx, **const** UPDOWN\_Flag\_type Flag )

UPDOWN\_type \* **const** LX\_UPDOWNx – указатель на структуру регистров UPDOWN

**const** UPDOWN\_Flag\_type Flag – флаг

**Описание**

Функция возвращает актуальное значение флага **const** UPDOWN\_Flag\_type Flag модуля UPDOWN\_type \* **const** LX\_UPDOWNx из регистра SR.

Выбор модуля осуществляется входным аргументом UPDOWN\_type \* **const** LX\_UPDOWNx, допустимые значения которого описаны в функции **HAL\_UPDOWN\_Init(…)**.

Выбор флага осуществляется входным аргументом Flag, который может принимать следующие значения:

**typedef** **enum** {

*UPDOWN\_Flag\_Treq* – флаг T\_req (бит 0 регистра SR),

*UPDOWN\_Flag\_Rreq* – флаг R\_req (бит 1 регистра SR),

*UPDOWN\_Flag\_Ferr* – флаг Ferr (бит 2 регистра SR),

*UPDOWN\_Flag\_Overf* – флаг overf (бит 3 регистра SR),

} UPDOWN\_Flag\_type;

Возвращаемый результат может принимать следующие значения:

**typedef** **enum** {

*UPDOWN\_FlagState\_Off* - флаг сброшен (0),

*UPDOWN\_FlagState\_On* - флаг установлен (1).

} UPDOWN\_FlagState\_type;

**Пример**

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

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

{

UPDOWN\_Init\_type UD;

UPDOWN\_Flag\_type F;

HAL\_UPDOWN\_DefaultInitStruct (&UD);

HAL\_UPDOWN\_Init (LX\_UPDOWN0, &UD);

F = HAL\_UPDOWN\_GetFlag (LX\_UPDOWN0, *UPDOWN\_Flag\_Treq*);

**while** (1)

{

}

}

## HAL\_UPDOWN\_ConnectDMA

**Резюме**

**void** **HAL\_UPDOWN\_ConnectDMA**( UPDOWN\_type \* **const** LX\_UPDOWNx, uint32\_t ch\_number, **void** \*tcb )

UPDOWN\_type \* **const** LX\_UPDOWNx – указатель на структуру регистров UPDOWN

uint32\_t ch\_number – номер DMA канала (0-3)

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

**Описание**

Функция подключает DMA к модулю LX\_UPDOWNx. Функционирование DMA (прием или выдача данных) определяется исходя из бита DAM регистра CR модуля.

Выбор модуля осуществляется входным аргументом UPDOWN\_еype \* **const** LX\_UPDOWNx, допустимые значения которого описаны в функции **HAL\_UPDOWN\_Init(…)**.

Выбор канала DMA осуществляет с помощью входного аргумента ch\_number, который должен иметь значения [0; 3].

Входной аргумент tcb задает настройки регистра TCB для DMA.

**Пример**

Пример отражает старт работы с АЦП 5101НВ015 на плате ТСКЯ.469575.014 Версия 2.1:

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

**#include** <builtins.h>

**#include** <sysreg.h>

**#define** ADC\_OEN GPIO\_PIN\_12

**#define** ADC\_CSB GPIO\_PIN\_9

**#define** ADC\_CALCRUN GPIO\_PIN\_11

uint32\_t **\_\_attribute**((aligned(4 \* (32/\_\_CHAR\_BIT\_\_)))) BufRx[1024];

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

{

UPDOWN\_Init\_type UD;

uint32\_t **\_\_attribute**((aligned(4 \* (32/\_\_CHAR\_BIT\_\_)))) TCB[4];

**volatile** uint32\_t tmp;

// Настройка UPDOWN:

HAL\_UPDOWN\_DefaultInitStruct (&UD);

UD.Mode = *UPDOWN\_Mode\_Down*;

UD.LinkX = *UPDOWN\_Link\_0*;

UD.Cnt = 1234;

UD.Step = 10;

UD.Coefficient = 8;

UD.Shift = 10;

HAL\_UPDOWN\_Init (LX\_UPDOWN0, &UD);

// Настройка DMA:

TCB[0] = (uint32\_t) BufRx;

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

TCB[2] = 0;

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

HAL\_UPDOWN\_ConnectDMA (LX\_UPDOWN0, 0, TCB);

// Включение АЦП:

HAL\_GPIO\_Init (LX\_GPIO\_PA, ADC\_OEN | ADC\_CSB, GPIO\_PinMode\_Out);

HAL\_GPIO\_WritePins (LX\_GPIO\_PA, ADC\_OEN | ADC\_CSB, ADC\_OEN | ADC\_CSB);

// Калибровка АЦП:

HAL\_GPIO\_Init (LX\_GPIO\_PA, ADC\_CALCRUN, GPIO\_PinMode\_Out);

HAL\_GPIO\_WritePin (LX\_GPIO\_PA, ADC\_CALCRUN, GPIO\_PinState\_Set);

tmp = 0;

**while** (tmp < 1000) tmp++;

HAL\_GPIO\_WritePin (LX\_GPIO\_PA, ADC\_CALCRUN, GPIO\_PinState\_Reset);

// Настройка Link0:

tmp = LRX\_EN | LRX\_DSIZE\_8BIT | 0x0800 | 0x8000 | 0x2000 | 0x4000;

\_\_builtin\_sysreg\_write (*\_\_LRCTL0*, tmp);

tmp = 0;

// Ожидание окончания заполнения буфера:

HAL\_DMA\_WaitForChannel (4 + 0);

**while** (1) {

}

}