# Описание используемых функций

## HAL\_GPIO\_Init

**Резюме**

**void HAL\_GPIO\_Init**( GPIO\_PortPi \* **const** GPIO\_Pi, **const** uint32\_t Pins, **const** GPIO\_PinMode Mode ) – функция выбора режима работы GPIO\_PinMode Mode пинов **const** uint32\_t Pins порта GPIO\_PortPi \* **const** GPIO\_Pi (PA, PB или PC).

**Описание**

Входной аргумент порта GPIO\_PortPi \* **const** GPIO\_Pi может принимать следующие значения:

LX\_GPIO\_PA – порт PA,

LX\_GPIO\_PB – порт PB,

LX\_GPIO\_PC – порт PC.

Входной аргумент **const** uint32\_t Pins представляет собой набор из 32 бит. Каждому биту Pins соответствует один пин порта по порядку, т.е. пину GPIO\_Pi[0] соответствует бит 0, GPIO\_Pi [1] – бит 1, а пину GPIO\_Pi[31] – бит 31. Для удобства заведено перечисление пинов:

**typedef** **enum**

{

*GPIO\_PIN\_0* = 0x00000001UL,

…

*GPIO\_PIN\_31* = 0x80000000UL,

} GPIO\_PIN;

Для всех пинов порта, соответствующих битам аргумента Pins, установленных в «1», будет назначен режим работы, заданный входным аргументом **const** GPIO\_PinMode Mode.

Входной аргумент **const** GPIO\_PinMode Mode может принимать следующие значения:

*GPIO\_PinMode\_In* - вход,

*GPIO\_PinMode\_InPU* - вход с подтяжкой к высокому уровню (лог «1»),

*GPIO\_PinMode\_Out* - выход,

*GPIO\_PinMode\_OutPU* - выход с подтяжной к высокому уровню (лог «1»),

*GPIO\_PinMode\_Alt* - альтернативная функция.

**Пример**

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

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

// PС0 - выход:

HAL\_GPIO\_Init (LX\_GPIO\_PC, *GPIO\_PIN\_0*, *GPIO\_PinMode\_Out*);

// PA1 - выход с подтяжкой к 1:

HAL\_GPIO\_Init (LX\_GPIO\_PA, *GPIO\_PIN\_1*, *GPIO\_PinMode\_OutPU*);

// PB11 и PB12 вход с подтяжкой к 1:

HAL\_GPIO\_Init (LX\_GPIO\_PB, *GPIO\_PIN\_11* | *GPIO\_PIN\_12*, *GPIO\_PinMode\_InPU*);

// PA2 и PA3 - альтернативная функция (UART1)

HAL\_GPIO\_Init (LX\_GPIO\_PA, *GPIO\_PIN\_2* | *GPIO\_PIN\_3*, *GPIO\_PinMode\_Alt*);

**while** (1) {

}

**return** 0;

}

## HAL\_GPIO\_DeInit

**Резюме**

**void** **HAL\_GPIO\_DeInit** ( GPIO\_PortPi \* **const** GPIO\_Pi, **const** uint32\_t Pins ) – функция деинициализации пинов **const** uint Pins порта GPIO\_PortPi \* **const** GPIO\_Pi (PA, PB или PC).

**Описание**

Функция сбрасывает настройку пинов порта в начальное состояние (вход без подтяжки к «1»). Так же для пинов сбрасываются настройки прерываний и флаги событий срабатывания прерываний по фронтам.

Входной аргумент порта GPIO\_PortPi \* **const** GPIO\_Pi принимает значения, аналогичные функции **HAL\_GPIO\_Init(…)**.

Входной аргумент **const** uint32\_t Pins используется так же, как и в функции **HAL\_GPIO\_Init(…)**.

**Пример**

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

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

// Настройка PA0 как выход:

HAL\_GPIO\_Init (LX\_GPIO\_PA, *GPIO\_PIN\_0*, *GPIO\_PinMode\_Out*);

// Сборос настройки:

HAL\_GPIO\_DeInit (LX\_GPIO\_PA, *GPIO\_PIN\_0*);

**while** (1) {

}

**return** 0;

}

## HAL\_GPIO\_PxD\_Init

**Резюме**

**void** **HAL\_GPIO\_PxD\_Init** ( **const** uint32\_t Pins, **const** GPIO\_PinMode Mode ) – функция инициализации режима работы **const** GPIO\_PinMode Mode пинов **const** uint32\_t Pins для порта PxD.

**Описание**

Поведение входных аргументов **const** uint32\_t Pins и **const** GPIO\_PinMode Mode аналогичны функции **HAL\_GPIO\_Init(…)**.

**Пример**

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

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

// PxD0 - выход:

HAL\_GPIO\_PxD\_Init (*GPIO\_PIN\_0*, *GPIO\_PinMode\_Out*);

// PxD12, PxD13 и PxD25 - вход с подтяжкой к 1:

HAL\_GPIO\_PxD\_Init (*GPIO\_PIN\_12* | *GPIO\_PIN\_13* | *GPIO\_PIN\_25*, *GPIO\_PinMode\_InPU*);

**while** (1) {

}

**return** 0;

}

## HAL\_GPIO\_Px\_DeInit

**Резюме**

**void** **HAL\_GPIO\_PxD\_DeInit** ( **const** uint32\_t Pins ) – функция деинициализации режима работы пинов **const** uint32\_t Pins порта PxD.

**Описание**

Функция сбрасывает настройку пинов порта PxD в начальное состояние – пины общего назначения, сконфигурированные как вход без подтяжки. Сбрасываемые пины зависят от входного аргумента uint Pins, формат которого аналогичен описанию в функции **HAL\_GPIO\_Init(…)**.

**Пример**

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

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

// PxD12, PxD13 и PxD25 - вход с подтяжкой к 1:

HAL\_GPIO\_PxD\_Init (*GPIO\_PIN\_12* | *GPIO\_PIN\_13* | *GPIO\_PIN\_25*, *GPIO\_PinMode\_InPU*);

// Сброс настройки для PxD12 и PxD25:

HAL\_GPIO\_PxD\_DeInit (*GPIO\_PIN\_12* | *GPIO\_PIN\_25*);

**while** (1) {

}

**return** 0;

}

## HAL\_GPIO\_Px\_AltInit

**Резюме**

**void** **HAL\_GPIO\_Px\_AltInit** ( **const** GPIO\_PxAlt\_type PxAlt, **const** GPIO\_PxAlt\_Mode\_type PxAltMode ) – функция включения альтернативных функций для портов PxA и PxD

**Описание**

Функция позволяет конфигурировать пины портов PxA и PxD для организации интерфейса доступа к внешнему устройству. Тип интерфейса зависит от входного аргумента **const** GPIO\_PxAlt\_Mode\_type PxAltMode, который может принимать следующие значения:

|  |  |  |
| --- | --- | --- |
| **№** | **Значение** | **Тип** |
|
|  | *GPIO\_PxAlt\_Mode\_SDRAM* | Шина внешней памяти |
|  | *GPIO\_PxAlt\_Mode\_NAND* | NAND флэш-память |
|  | *GPIO\_PxAlt\_Mode\_MILSTD* | МКПД интерфейс |

Обращаю внимание на то, что **const** GPIO\_PxAlt\_Mode\_type PxAltMode влияет только на старшие биты регистры PxD. Т.к эти интерфейсы NAND и MILSTD физически подключены только в 16 старшим пинам регистра PxD (т.е к младшим 16 битам PxD всегда подключена шина внешней памяти). Интерфейса SDRAM подключен ко всем 32 битам PxD, и в зависимости от конфигурации альтернативных функций для пинов может работать как по 16 битной так и по 32 битной шине.

Альтернативные функции для пинов регистров PxD и PxA настраиваются с помощью аргумента **const** GPIO\_PxAlt\_type PxAlt. Он принимает следущие значения:

|  |  |  |
| --- | --- | --- |
| **№** | **Значение** | **Тип** |
|
| 1 | GPIO\_PxAlt\_Data7\_0 | Включение альтернативной функции для битов 7-0 регистра PxD |
| 2 | GPIO\_PxAlt\_Data15\_8 | Включение альтернативной функции для битов 15-8 регистра PxD |
| 3 | GPIO\_PxAlt\_Data23\_16 | Включение альтернативной функции для битов 23-16 регистра PxD |
|  | GPIO\_PxAlt\_Data31\_24 | Включение альтернативной функции для битов 31-24 регистра PxD |
|  | GPIO\_PxAlt\_Addr7\_0 | Включение альтернативной функции для битов 7-0 регистра PxA |
|  | GPIO\_PxAlt\_Addr15\_8 | Включение альтернативной функции для битов 15-8 регистра PxA |
|  | GPIO\_PxAlt\_Addr21\_16 | Включение альтернативной функции для битов 21-16 регистра PxA |
|  | GPIO\_PxAlt\_All | Включение альтернативной функции для всех битов регистров PxA и PxD |

**Пример**

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

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

// Инициализация альтернативных функций для NAND

*HAL\_GPIO\_Px\_AltInit (GPIO\_PxAlt\_Data23\_16 | GPIO\_PxAlt\_Data31\_24 | GPIO\_PxAlt\_Addr7\_0 | GPIO\_PxAlt\_Addr15\_8 | GPIO\_PxAlt\_Addr21\_16, GPIO\_PxAlt\_Mode\_NAND);*

// Пример инициализации альтернативных функций для MILSTD

*HAL\_GPIO\_Px\_AltInit (GPIO\_PxAlt\_Data23\_16 | GPIO\_PxAlt\_Data31\_24 | GPIO\_PxAlt\_Addr7\_0 | GPIO\_PxAlt\_Addr15\_8 | GPIO\_PxAlt\_Addr21\_16, GPIO\_PxAlt\_Mode\_MILSTD);*

// Инициализация альтернативных функций для SDRAM

*HAL\_GPIO\_Px\_AltInit (GPIO\_PxAlt\_All, GPIO\_PxAlt\_Mode\_SDRAM);*

**while** (1) {

}

**return** 0;

}

## HAL\_GPIO\_Px\_AltDeInit

**Резюме**

**void** **HAL\_GPIO\_Px\_AltDeInit**( **const** GPIO\_PxAlt\_type PxAlt) – функция выключения альтернативных функций для портов PxA и PxD

**Описание**

Функция отключает альтернативные функции для всех пинов регистров PxA и PxD.

**Пример**

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

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

// Включение режима GPIO для всех пинов регистров PxA и PxD

*HAL\_GPIO\_Px\_AltDeInit(GPIO\_PxAlt\_Data23\_16);*

**while** (1) {

}

**return** 0;

}

## HAL\_GPIO\_WritePin

**Резюме**

**void** **HAL\_GPIO\_WritePin** ( GPIO\_PortPi \* **const** GPIO\_Pi, **const** GPIO\_PIN Pin, **const** GPIO\_PinState NewState ) – функция записи значения **const** GPIO\_PinState NewState в пин **const** GPIO\_PIN Pin порта GPIO\_PortPi \* **const** GPIO\_Pi (PA, PB или PC).

**Описание**

Функция записывает значение в регистр PiDR. Порт записи определяется входным аргументом GPIO\_PortPi \* **const** GPIO\_Pi, возможные значения которого описаны для функции **HAL\_GPIO\_Init(…)**.

Пин, в который записывается значение, определяется входным аргуметом **const** GPIO\_PIN Pin. Его возможные значения содержатся в нумерованном списке GPIO\_PIN, структура которого показана в описании функции **HAL\_GPIO\_Init(…)**.

Записываемое значение определяется входным аргументом **const** GPIO\_PinState NewState. Он может принимать следующие значения:

**typedef** **enum** {

*GPIO\_PinState\_Reset* = 0 – запись в пин 0 (низкий уровень),

*GPIO\_PinState\_Set* = 1 – запись в пин 1 (высокий уровень).

} GPIO\_PinState;

**Пример**

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

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

// Конфигурация пина:

HAL\_GPIO\_Init (LX\_GPIO\_PA, *GPIO\_PIN\_0*, *GPIO\_PinMode\_Out*);

// Установка выходного значения 1:

HAL\_GPIO\_WritePin (LX\_GPIO\_PA, *GPIO\_PIN\_0*, *GPIO\_PinState\_Set*);

**while** (1) {

}

**return** 0;

}

## HAL\_GPIO\_ReadPin

**Резюме**

GPIO\_PinState **HAL\_GPIO\_ReadPin** ( GPIO\_PortPi \* **const** GPIO\_Pi, **const** GPIO\_PIN Pin ) – функция получения значения пина **const** GPIO\_PIN Pin из порта GPIO\_PortPi \* **const** GPIO\_Pi (PA, PB или PC).

**Описание**

Функция возвращает GPIO\_PinState – текущее состояние пина из регистра PiPXD. Описание возможных значений дано в описании функции **HAL\_GPIO\_WritePin(…)**.

Порт для чтения задается с помощью входного аргумента GPIO\_PortPi \* **const** GPIO\_Pi, допустимые значения которого даны в описании к функции **HAL\_GPIO\_Init(…)**.

Пин для чтения определяется входным аргументом **const** GPIO\_PIN Pin, допустимые значения которого указаны в описании функции **HAL\_GPIO\_WritePin(…)**.

**Пример**

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

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

GPIO\_PinState A;

HAL\_GPIO\_Init (LX\_GPIO\_PA, *GPIO\_PIN\_0*, *GPIO\_PinMode\_In*);

A = HAL\_GPIO\_ReadPin (LX\_GPIO\_PA, *GPIO\_PIN\_0*);

**while** (1) {

}

**return** 0;

}

## HAL\_GPIO\_WritePins

**Резюме**

**void** **HAL\_GPIO\_WritePins** ( GPIO\_PortPi \* **const** GPIO\_Pi, **const** uint32\_t Pins, **const** uint32\_t Data ) – функция установки значений **const** uint32\_t Data для набора пинов **const** uint32\_t Pins порта GPIO\_PortPi \* **const** GPIO\_Pi (PA, PB или PC).

**Описание**

Значения входного аргумента GPIO\_PortPi \* **const** GPIO\_Pi, задающего порт, указаны в описании к функции **HAL\_GPIO\_Init(…)**.

Значения входного аргумента **const** uint32\_t Pins, определяющего пины для записи, аналогичны значениям в функции **HAL\_GPIO\_Init(…)**.

Входной аргумент **const** uint32\_t Data определяет записываемые значения в пины. При этом каждому пину соответствует один бит из Data. Нулевому пину соответствует значение, указанное в нулевом бите Data, 1-му пину – значение в 1-м бите, …, 31-му пину – значение в 31-м бите.

**Пример**

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

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

HAL\_GPIO\_Init (LX\_GPIO\_PA, *GPIO\_PIN\_0* | *GPIO\_PIN\_1* | *GPIO\_PIN\_2*, *GPIO\_PinMode\_Out*);

// Записать 1 в пины 0 и 2:

HAL\_GPIO\_WritePins (LX\_GPIO\_PA, *GPIO\_PIN\_0* | *GPIO\_PIN\_1* | *GPIO\_PIN\_2*, 0x0005);

// Записать 1 во все пины:

HAL\_GPIO\_WritePins (LX\_GPIO\_PA, *GPIO\_PIN\_0* | *GPIO\_PIN\_1* | *GPIO\_PIN\_2*, 0xFFFF);

// Записать 0 во все пины:

HAL\_GPIO\_WritePins (LX\_GPIO\_PA, *GPIO\_PIN\_0* | *GPIO\_PIN\_1* | *GPIO\_PIN\_2*, 0);

**while** (1) {

}

**return** 0;

}

## HAL\_GPIO\_TogglePins

**Резюме**

**void** **HAL\_GPIO\_TogglePins**( GPIO\_PortPi \* **const** GPIO\_Pi, **const** uint32\_t Pins ) – функция переключает текущее значение пинов **const** uint32\_t Pins порта GPIO\_PortPi \* **const** GPIO\_Pi (PA, PB или PC) на противоположное значение.

**Описание**

Значения входного аргумента GPIO\_PortPi \* **const** GPIO\_Pi, задающего порт, указаны в описании к функции **HAL\_GPIO\_Init(…)**.

Значения входного аргумента **const** uint32\_t Pins, определяющего пины для записи, аналогичны значениям в функции **HAL\_GPIO\_Init(…)**.

**Пример**

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

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

HAL\_GPIO\_Init (LX\_GPIO\_PA, *GPIO\_PIN\_0* | *GPIO\_PIN\_1* | *GPIO\_PIN\_2*, *GPIO\_PinMode\_Out*);

// PA[2:0] = 0x5:

HAL\_GPIO\_WritePins (LX\_GPIO\_PA, *GPIO\_PIN\_0* | *GPIO\_PIN\_1* | *GPIO\_PIN\_2*, 0x0005);

// PA[2:0] = 0x2:

HAL\_GPIO\_TogglePins (LX\_GPIO\_PA, *GPIO\_PIN\_0* | *GPIO\_PIN\_1* | *GPIO\_PIN\_2*);

**while** (1) {

}

**return** 0;

}

## HAL\_GPIO\_ReadPins

**Резюме**

uint32\_t **HAL\_GPIO\_ReadPins**( GPIO\_PortPi \* **const** GPIO\_Pi, **const** uint32\_t Pins ) – функция возвращает текущие значения набора пинов **const** uint32\_t Pins из порта GPIO\_PortPi \* **const** GPIO\_Pi (PA, PB или PC).

**Описание**

Значения входного аргумента GPIO\_PortPi \* **const** GPIO\_Pi, задающего порт, указаны в описании к функции **HAL\_GPIO\_Init(…)**.

Значения входного аргумента **const** uint32\_t Pins, определяющего пины для записи, аналогичны значениям в функции **HAL\_GPIO\_Init(…)**.

В возвращаемом функции значении каждый бит соответствует значению пина с соответствующем номером, то есть пину Pi[0] соответствует 0-й бит, пину Px[1] – 1-й бит и т.д.

**Пример**

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

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

**int** A;

HAL\_GPIO\_Init (LX\_GPIO\_PA, *GPIO\_PIN\_0* | *GPIO\_PIN\_1* | *GPIO\_PIN\_9*, *GPIO\_PinMode\_Out*);

HAL\_GPIO\_WritePins (LX\_GPIO\_PA, *GPIO\_PIN\_0* | *GPIO\_PIN\_1* | *GPIO\_PIN\_9*, 0x0203);

A = HAL\_GPIO\_ReadPins (LX\_GPIO\_PA, *GPIO\_PIN\_0* | *GPIO\_PIN\_9*); // A = 0x0201

**while** (1) {

}

**return** 0;

}

## HAL\_GPIO\_Px\_WritePins

**Резюме**

**void** **HAL\_GPIO\_Px\_WritePins**( **const** GPIO\_PxPort Px, **const** uint32\_t Pins, **const** uint32\_t Data ) – функция записывает значения **const** uint32\_t Data в набор пинов **const** uint32\_t Pins порта **const** GPIO\_PxPort Px (PxA или PxD).

**Описание**

Входной аргумент **const** GPIO\_PxPort Px определяет порт для записи и может принимать следующие значения:

**typedef** **enum** {

*GPIO\_PxPort\_A* - порт PxA,

*GPIO\_PxPort\_D* - порт PxD,

} GPIO\_PxPort;

Входной аргумент **const** uint Pins определяет состав пинов для записи и аналогичен входному аргументу функции **HAL\_GPIO\_Init(…)**. Для порта PxA запись значений в пины с номерами больше 21 не запрещена, но и лишена смысла.

Входной аргумент **const** uint32\_t Data определяет записываемые в пины данные и аналогичен входному аргументу **HAL\_GPIO\_WritePins(…)**.

**Пример**

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

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

HAL\_GPIO\_PxD\_Init (*GPIO\_PIN\_0* | *GPIO\_PIN\_1*, *GPIO\_PinMode\_Out*);

HAL\_GPIO\_Px\_WritePins (*GPIO\_PxPort\_D*, *GPIO\_PIN\_0* | *GPIO\_PIN\_1*, 0x0002);

// PxD[1:0] = 0x0002

HAL\_GPIO\_Px\_WritePins (*GPIO\_PxPort\_A*, *GPIO\_PIN\_12* | *GPIO\_PIN\_13*, 0x3000);

// PxA[13:12] = 0x3000

**while** (1) {

}

**return** 0;

}

## HAL\_GPIO\_Px\_ReadPins

**Резюме**

uint32\_t **HAL\_GPIO\_Px\_ReadPins**( **const** GPIO\_PxPort Px, **const** uint32\_t Pins ) – функция возвращает текущие значения набора пинов **const** uint32\_t Pins из порта **const** GPIO\_PxPort Px (PxA или PxD).

**Описание**

Входной аргумент **const** GPIO\_PxPort Px определяет порт для записи (PxA или PxB) и может принимать значения, описанные в функции **HAL\_GPIO\_Px\_WritePins(…)**.

Значения входного аргумента **const** uint32\_t Pins, определяющего пины для чтения, аналогичны значениям в функции **HAL\_GPIO\_Init(…)**.

В возвращаемом функции значении каждый бит соответствует значению пина с соответствующем номером, то есть пину Pi[0] соответствует 0-й бит, пину Px[1] – 1-й бит и т.д.

**Пример**

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

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

**int** D;

HAL\_GPIO\_PxD\_Init (*GPIO\_PIN\_0* | *GPIO\_PIN\_1*, *GPIO\_PinMode\_Out*);

HAL\_GPIO\_Px\_WritePins (*GPIO\_PxPort\_D*, *GPIO\_PIN\_0* | *GPIO\_PIN\_1*, 0x0003);

// PxD[1:0] = 0x0003

D = HAL\_GPIO\_Px\_ReadPins (*GPIO\_PxPort\_D*, *GPIO\_PIN\_1*);

// D = 0x0002

**while** (1) {

}

**return** 0;

}

## HAL\_GPIO\_InterruptConfig

**Резюме**

**void** **HAL\_GPIO\_InterruptConfig**( GPIO\_PortPi \* **const** GPIO\_Pi, **const** uint32\_t Pins, **const** GPIO\_InterruptMode Mode ) – функция конфигурирует и запрещает/разрешает прерывание с типом **const** GPIO\_InterruptMode Mode от пинов **const** uint32\_t Pins порта GPIO\_PortPi \* **const** GPIO\_Pi (PA, PB или PC).

**Описание**

Значения входного аргумента GPIO\_PortPi \* **const** GPIO\_Pi, задающего порт, указаны в описании к функции **HAL\_GPIO\_Init(…)**.

Значения входного аргумента **const** uint32\_t Pins, определяющего пины, аналогичны значениям в функции **HAL\_GPIO\_Init(…)**.

Входной аргумент **const** GPIO\_InterruptMode Mode определяет тип прерывания и может принимать следующие значения:

**typedef** **enum** {

*GPIO\_InterruptMode\_Off* - прерывание выключено,

*GPIO\_InterruptMode\_Posedge* - прерывание по возрастающему фронту,

*GPIO\_InterruptMode\_Negedge* - прерывание по спадающему фронту,

*GPIO\_InterruptMode\_LowLvl* - прерывание по низкому уровню,

*GPIO\_InterruptMode\_HighLvl* - прерывание по высокому уровню,

} GPIO\_InterruptMode;

При вызове функции сбрасываются флаги возникновения прерываний по возрастающему или спадающему фронту.

**Пример**

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

**void** **GPIO\_Handler**();

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

// Конфигурация прерывыания от GPIO:

HAL\_Interrupt\_IMASKClear();

HAL\_Interrupt\_ILATClear();

HAL\_Interrupt\_GlobalEnable();

HAL\_Interrupt\_Enable (*intGPIO*, &GPIO\_Handler);

// Настройка пина:

HAL\_GPIO\_Init (LX\_GPIO\_PA, *GPIO\_PIN\_0*, *GPIO\_PinMode\_InPU*);

HAL\_GPIO\_InterruptConfig (LX\_GPIO\_PA, *GPIO\_PIN\_0*, *GPIO\_InterruptMode\_LowLvl*);

**while** (1) {

}

**return** 0;

}

**#pragma** interrupt

**void** **GPIO\_Handler** () {

HAL\_GPIO\_InterruptConfig (LX\_GPIO\_PA, *GPIO\_PIN\_0*, *GPIO\_InterruptMode\_Off*);

}

## HAL\_GPIO\_InterruptCheck

**Резюме**

uint32\_t **HAL\_GPIO\_InterruptCheck**( GPIO\_PortPi \* **const** GPIO\_Pi, **const** uint32\_t Pins ) – функция проверяет события возникновения прерываний от пинов **const** uint32\_t Pins порта GPIO\_PortPi \* **const** GPIO\_Pi (PA, PB или PC).

**Описание**

Функция возвращает значение регистра PiINTREQ с учетом выбранных пинов, при этом в возвращаемом значении пину с номером 0 соответствует 0-ой бит, пину с номером 1 – 1-ый бит т. д.

Значения входного аргумента GPIO\_PortPi \* **const** GPIO\_Pi, задающего порт, указаны в описании к функции **HAL\_GPIO\_Init(…)**.

Значения входного аргумента **const** uint32\_t Pins, определяющего пины, аналогичны значениям в функции **HAL\_GPIO\_Init(…)**.

**Пример**

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

**void** **GPIO\_Handler**();

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

// Конфигурация прерывыания от GPIO:

HAL\_Interrupt\_IMASKClear();

HAL\_Interrupt\_ILATClear();

HAL\_Interrupt\_GlobalEnable();

HAL\_Interrupt\_Enable (*intGPIO*, &GPIO\_Handler);

// Настройка пина:

HAL\_GPIO\_Init (LX\_GPIO\_PA, *GPIO\_PIN\_0*, *GPIO\_PinMode\_InPU*);

HAL\_GPIO\_InterruptConfig (LX\_GPIO\_PA, *GPIO\_PIN\_0* | *GPIO\_PIN\_1*, *GPIO\_InterruptMode\_LowLvl*);

**while** (1) {

}

**return** 0;

}

**#pragma** interrupt

**void** **GPIO\_Handler** () {

**int** i = 0;

**if** (HAL\_GPIO\_InterruptCheck (LX\_GPIO\_PA, *GPIO\_PIN\_0*)) i++;

**if** (HAL\_GPIO\_InterruptCheck (LX\_GPIO\_PA, *GPIO\_PIN\_1*)) i--;

}

## HAL\_GPIO\_InterruptEdgeClear

**Резюме**

**void** **HAL\_GPIO\_InterruptEdgeClear**( GPIO\_PortPi \* **const** GPIO\_Pi, **const** uint32\_t Pins ) – функция сбрасывает флаги запросов прерываний по фронту в регистре PiECLR от пинов **const** uint32\_t Pins порта GPIO\_PortPi \* **const** GPIO\_Pi (PA, PB или PC).

**Описание**

Значения входного аргумента GPIO\_PortPi \* **const** GPIO\_Pi, задающего порт, и входного аргумента **const** uint32\_t Pins, определяющего пины, указаны в описании к функции **HAL\_GPIO\_Init(…)**.

**Пример**

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

**void** **GPIO\_Handler**();

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

// Конфигурация прерывыания от GPIO:

HAL\_Interrupt\_IMASKClear();

HAL\_Interrupt\_ILATClear();

HAL\_Interrupt\_GlobalEnable();

HAL\_Interrupt\_Enable (*intGPIO*, &GPIO\_Handler);

// Настройка пина:

HAL\_GPIO\_Init (LX\_GPIO\_PA, *GPIO\_PIN\_0*, *GPIO\_PinMode\_InPU*);

HAL\_GPIO\_InterruptConfig (LX\_GPIO\_PA, *GPIO\_PIN\_0*, *GPIO\_InterruptMode\_Negedge*);

**while** (1) {

}

**return** 0;

}

**#pragma** interrupt

**void** **GPIO\_Handler** () {

HAL\_GPIO\_InterruptEdgeClear (LX\_GPIO\_PA, *GPIO\_PIN\_0*);

}