# **Описание используемых типов**

## **LinkRx\_Init\_type**

|  |  |  |
| --- | --- | --- |
| Имя поля | Описание | Значение |
| CheckSum | Настройка проверки CheckSum | *Link\_CheckSum\_En* – разрешено  *Link\_CheckSum\_Dis –* запрещено |
| CheckBCMPI | Настройка анализа BCMPI | *Link\_CheckBCMP\_En* – разрешено  *Link\_CheckBCMP\_Dis –* запрещено |
| DataSize | Размер шины для принимаемых данных | *Link\_DataSize\_1bit*  *Link\_DataSize\_4bit*  *Link\_DataSize\_8bit*  *Link\_DataSize\_16bit* |
| TOIntEn | Разрешения прерывания по ситуации time out | *Link\_TOIT\_En* – разрешено  *Link\_TOIT\_Dis –* запрещено |
| OVRIntEn | Разрешения прерывания при переполнении буфера приемника | *Link\_OvrIT\_En* – разрешено  *Link\_OvrIT\_Dis* - запрещено |

## **LinkRxEx\_type**

|  |  |  |
| --- | --- | --- |
| Имя поля | Описание | Значение |
| GpsClk | Разрешение генерации клока GPS | *Link\_GpsClk\_Dis–* запрещено  *Link\_GpsClk\_En –* разрешено |
| RxDest | Выбор приемника данных | *Link\_RxDest\_Buff* – буфер порта связи  *Link\_RxDest\_UpDown* – модуль UP-DOWN |
| AdcDataSize | Режим для ADC 8 бит шины данных | *Link\_AdcDataSize\_10b*  *Link\_AdcDataSize\_12b*  *Link\_AdcDataSize\_14b*  *Link\_AdcDataSize\_16b* |
| Rcode | Преобразование смещенного кода в дополнительный для 8 бит шины данных | *Link\_Rcode\_Dis* – запрещено  *Link\_Rcode\_En* - разрешено |
| BitOrder | Порядок выдачи бит | *Link\_BitOrder\_LoFrontHiCut* – первая половина бит передается по фронту, вторая по срезу  *Link\_BitOrder\_EvenFrontOddCut* – чётные биты передаются по фронту, нечетные – по срезу |

## **LinkTx\_Init\_type**

|  |  |  |
| --- | --- | --- |
| Имя поля | Описание | Значение |
| CheckSum | Настройка проверки CheckSum | *Link\_CheckSum\_En* – разрешено  *Link\_CheckSum\_Dis –* запрещено |
| CheckBCMPO | Разрешение формирования BCMPO | *Link\_CheckBCMP\_En* – разрешено  *Link\_CheckBCMP\_Dis –* запрещено |
| DataSize | Размер шины для принимаемых данных | *Link\_DataSize\_1bit*  *Link\_DataSize\_4bit*  *Link\_DataSize\_8bit*  *Link\_DataSize\_16bit* |
| TOIntEn | Разрешения прерывания по ситуации time out | *Link\_TOIT\_En* – разрешено  *Link\_TOIT\_Dis –* запрещено |
| ClkSource | Источник синхросигнала передатчика | *Link\_TxClk\_Master* – внутренний генератор  *Link\_TxClk\_Slave* – синхросигнал приемника  *Link\_TxClk\_InvSlave* – инвертированный синхросигнал приемника |

## **LinkTxEx\_type**

|  |  |  |
| --- | --- | --- |
| Имя поля | Описание | Значение |
| TxSource | Выбор источника данных для передачи | *Link\_TxDest\_Buff* – буфер порта связи  *Link\_TxDest\_UpDown* – модуль UP-DOWN |
| Tcode | Преобразование дополнительного кода в смещенный | *Link\_Tcode\_Dis* – запрещено  *Link\_Tcode\_En –* разрешено |
| CheckACKI | Анализ сигнала LxACKI | *Link\_CheckACKI\_Dis* – запрещен  *Link\_CheckACKI\_En –* разрешен |

# **Описание используемых функций**

## **HAL\_LinkRx\_Enable**

**Резюме**

int32\_t **HAL\_LinkRx\_Enable**( uint32\_t link\_num, LinkRx\_Init\_type \*LRxInit, LinkRxEx\_type \*LRxExInit )

uint32\_t link\_num – номер порта связи. Принимает значение 0 либо 1.

LinkRx\_Init\_type \*LRxInit – указатель на структуры инициализации портов связи.

LinkRxEx\_type \*LRxExInit – указатель на структуры дополнительных параметров инициализации портов связи.

**Описание**

Функция инициализирует приемник выбранного порта связи(link-порт). Включить альтернативные функции необходимо самостоятельно (установить в единицу соответствующие биты регистра GPC\_ALT) для регистра GPC для 24 или 28 бита, в зависимости от выбранного порта связи. Для этого можно использовать HAL\_GPIO. Для полей структуры необходимо использовать значения, описанные в начале документа. Функция возвращает -1 если номер порта связи указан неверно, в противном случае функция возвращает ноль.

**Пример**

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

**#define** LINK\_NUM 1

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

{

LinkRx\_Init\_type LRx;

int LInitSt;

LRx.CheckSum = *Link\_CheckSum\_Dis*;

LRx.CheckBCMPI = *Link\_CheckBCMP\_Dis*;

LRx.OVRIntEn = *Link\_OvrIT\_Dis*;

LRx.TOIntEn = *Link\_TOIT\_Dis*;

LRx.DataSize = *Link\_DataSize\_1bit*;

//Link Enable

LInitSt = HAL\_LinkRx\_Enable(LINK\_NUM, &LRx, 0);

**return** 0;

}

## **HAL\_LinkTx\_Enable**

**Резюме**

int32\_t **HAL\_LinkTx\_Enable**( uint32\_t link\_num, LinkTx\_Init\_type \*LTxInit, LinkTxEx\_type \*LTxExInit )

uint32\_t link\_num – номер порта связи. Принимает значение 0 либо 1.

LinkTx\_Init\_type \*LTxInit – указатель на структуры инициализации портов связи.

LinkTxEx\_type \*LTxExInit – указатель на структуры дополнительных параметров инициализации портов связи

**Описание**

Функция инициализирует передатчик выбранного порта связи(link-порт). Так же, для разрешения формирование сигнала BCMPO, необходимо включить альтернативные функции для регистра GPC для 26 или 30 бита, в зависимости от выбранного порта связи. Если формирование BCMPO запрещено, то альтернативные функции желательно выключить для регистра GPC для 26 или 30 бита. Для полей структуры необходимо использовать значения, описанные в начале документа. Функция возвращает -1 если номер порта связи указан неверно, в противном случае функция возвращает ноль.

**Пример**

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

**#define** LINK\_NUM 1

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

{

LinkTx\_Init\_type LTx;

int LInitSt;

LTx.CheckBCMPO = *Link\_CheckBCMP\_Dis*;

LTx.CheckSum = *Link\_CheckSum\_Dis*;

LTx.TOIntEn = *Link\_TOIT\_Dis*;

LTx.ClkSource = *Link\_TxClk\_Master*;

LTx.DataSize = *Link\_DataSize\_4bit*;

//Link Enable

LInitSt = HAL\_LinkTx\_Enable(LINK\_NUM, &LTx, 0);

**return** 0;

}

## **HAL\_LinkRx\_GetStat**

**Резюме**

int32\_t **HAL\_LinkRx\_GetStat(** uint32\_t link\_num )

uint32\_t link\_num – номер порта связи. Принимает значение 0 либо 1.

**Описание**

Функция возвращает значение регистра статуса приемника заданного порта связи.

**Пример**

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

**#define** LINK\_NUM 0

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

{

**int** LRxST;

LRxST = HAL\_LinkRx\_GetStat(LINK\_NUM);

**return** 0;

}

## **HAL\_LinkRx\_GetStatCl**

**Резюме**

int32\_t **HAL\_LinkRx\_GetStatCl**( uint32\_t link\_num )

uint32\_t link\_num – номер порта связи. Принимает значение 0 либо 1.

**Описание**

Функция возвращает значение регистра статуса приемника заданного порта связи и сбрасывает в нем биты ошибок.

**Пример**

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

**#define** LINK\_NUM 0

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

{

**int** LRxST;

LRxST = HAL\_LinkRx\_GetStatCl(LINK\_NUM);

**return** 0;

}

## **HAL\_LinkTx\_GetStat**

**Резюме**

int32\_t **HAL\_LinkTx\_GetStat**( uint32\_t link\_num )

uint32\_t link\_num – номер порта связи. Принимает значение 0 либо 1.

**Описание**

Функция возвращает значение регистра статуса передатчика заданного порта связи.

**Пример**

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

**#define** LINK\_NUM 0

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

{

**int** LTxST;

LTxST = HAL\_LinkTx\_GetStat(LINK\_NUM);

**return** 0;

}

## **HAL\_LinkTx\_GetStatCl**

**Резюме**

int32\_t **HAL\_LinkTx\_GetStatCl**( uint32\_t link\_num )

uint32\_t link\_num – номер порта связи. Принимает значение 0 либо 1.

**Описание**

Функция возвращает значение регистра статуса передатчика заданного порта связи и сбрасывает в нем биты ошибок.

**Пример**

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

**#define** LINK\_NUM 0

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

{

**int** LTxST;

LTxST = HAL\_LinkTx\_GetStatCl(LINK\_NUM);

**return** 0;

}

## **HAL\_LinkRx\_Disable**

**Резюме**

**void** **HAL\_LinkRx\_Disable**( uint32\_t link\_num )

uint32\_t link\_num – номер порта связи. Принимает значение 0 либо 1.

**Описание**

Функция выключает приемник заданного порта связи. Альтернативные функции для битов GPIO необходимо выключить самостоятельно.

**Пример**

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

**#define** LINK\_NUM 0

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

{

HAL\_LinkRx\_Disable(LINK\_NUM);

**return** 0;

}

## **HAL\_LinkTx\_Disable**

**Резюме**

**void** **HAL\_LinkTx\_Disable**( uint32\_t link\_num )

uint32\_t link\_num – номер порта связи. Принимает значение 0 либо 1.

**Описание**

Функция выключает передатчик заданного порта связи. Альтернативные функции для битов GPIO необходимо выключить самостоятельно

**Пример**

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

**#define** LINK\_NUM 0

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

{

HAL\_LinkTx\_Disable(LINK\_NUM);

**return** 0;

}

## **HAL\_LinkDMA\_Receive**

**Резюме**

**void** **HAL\_LinkDMA\_Receive**( uint32\_t link\_num, **void** \*tcb )

uint32\_t link\_num – номер порта связи. Принимает значение 0 либо 1.

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

**Описание**

Функция запускает прием данных через заданный порт связи, по соответствующему каналу DMA. Например, для порта связи 0, для прима данных используется канал DMA 8. Таким образом, функция изменяет значения битов регистра DMACFGH, отвечающие за выбор периферии 8го канала DMA, выбирая для 8го канала в качестве периферии приемник порта связи 0. Также перезаписываются регистры TCB 8го канала DMA, на значения, заданные по указателю \*tcb. Для порта связи 1, используется канал DMA 9. Для ожидания конца приема данных рекомендовано использовать функцию HAL\_DMA\_WaitForChannel(8+LINK\_NUM).

**Пример**

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

**#define** LINK\_NUM 0

**#define** DATA\_SIZE 1024

**int** **\_\_attribute**((aligned(4 \* (32/\_\_CHAR\_BIT\_\_)))) data\_rx32[DATA\_SIZE];

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

{

LinkRx\_Init\_type LRx;

int LInitSt;

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

**int** error\_ct, DMARxStat;

LRx.CheckSum = *Link\_CheckSum\_Dis*;

LRx.CheckBCMPI = *Link\_CheckBCMP\_Dis*;

LRx.OVRIntEn = *Link\_OvrIT\_Dis*;

LRx.TOIntEn = *Link\_TOIT\_Dis*;

LRx.DataSize = *Link\_DataSize\_1bit*;

//Link Enable

LInitSt = HAL\_LinkRx\_Enable(LINK\_NUM, &LRx, 0);

**if** (LInitSt == -1) {error\_ct++;}

//TCB Init

tcbRx[0] = (**int**)&data\_rx32;

tcbRx[1] = (DATA\_SIZE<<16) | 4;

tcbRx[2] = 0;

tcbRx[3] = TCB\_INTMEM | TCB\_QUAD;//Работа с портом связи ВСЕГДА осуществляется //квадрословами.

//Receive Start

HAL\_LinkDMA\_Receive(LINK\_NUM, tcbRx);

//DMA Wait

DMARxStat = HAL\_DMA\_WaitForChannel(8+LINK\_NUM); //TX

**if** (DMARxStat != 0) {error\_ct++;}

**return** 0;

}

## **HAL\_LinkDMA\_Transmit**

**Резюме**

**void** **HAL\_LinkDMA\_Transmit**( uint32\_t link\_num, **void** \*tcb )

uint32\_t link\_num – номер порта связи. Принимает значение 0 либо 1.

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

**Описание**

Функция запускает передачу данных через заданный порт связи, по соответствующему каналу DMA. Например, для порта связи 0, для передачи данных используется канал DMA 4. Таким образом, функция изменяет значения битов регистра DMACFGL, отвечающие за выбор периферии 4го канала DMA, выбирая для 4го канала в качестве периферии передатчик порта связи 0. Также перезаписываются регистры TCB 4го канала DMA, на значения, заданные по указателю \*tcb. Для порта связи 1, используется канал DMA 5. Для ожидания конца приема данных рекомендовано использовать функцию HAL\_DMA\_WaitForChannel(4+LINK\_NUM).

**Пример**

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

**#define** LINK\_NUM 1

**#define** DATA\_SIZE 1024

**int** **\_\_attribute**((aligned(4 \* (32/\_\_CHAR\_BIT\_\_)))) data\_tx32[DATA\_SIZE];

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

{

LinkTx\_Init\_type LTx;

int LInitSt;

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

**int** error\_ct, DMATxStat;

LTx.CheckBCMPO= *Link\_CheckBCMP\_Dis*;

LTx.CheckSum = *Link\_CheckSum\_Dis*;

LTx.TOIntEn = *Link\_TOIT\_Dis*;

LTx.ClkSource = *Link\_TxClk\_Master*;

LTx.DataSize = *Link\_DataSize\_4bit*;

//Link Enable

LInitSt = HAL\_LinkTx\_Enable(LINK\_NUM, &LTx, 0);

**if** (LInitSt == -1) {error\_ct++;}

//TCB Init

tcbTx[0] = (**int**)&data\_tx32;

tcbTx[1] = (DATA\_SIZE<<16) | 4;

tcbTx[2] = 0;

tcbTx[3] = TCB\_INTMEM | TCB\_QUAD;//Работа с портом связи ВСЕГДА осуществляется //квадрословами.

//Transmit Start

HAL\_LinkDMA\_Transmit(LINK\_NUM, tcbTx);

//DMA Wait

DMATxStat = HAL\_DMA\_WaitForChannel(4+LINK\_NUM); //TX

**if** (DMATxStat != 0) {error\_ct++;}

**return** 0;

}

## **HAL\_LinkCore\_Receive**

**Резюме**

int32\_t **HAL\_LinkCore\_Receive**( uint32\_t link\_num, **void** \*BuffPtr, uint32\_t size )

uint32\_t link\_num – номер порта связи. Принимает значение 0 либо 1.

**void** \*BuffPtr – указатель на место сохранения принимаемых данных

uint32\_t size – размер принимаемых данных (в словах, т.е. должен быть кратен 4)

**Описание**

Функция осуществляет прием данных по средствам чтения буфера приемника порта связи и сохраняет их по указателю BuffPtr. В случаи установления бита ошибки в регистре статуса приемника, функция прерывается и возвращает значение -1. В случае успешного приема данных, функция возвращает 0.

**Пример**

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

**#define** LINK\_NUM 0

**#define** DATA\_SIZE 1024

**int** data\_rx32[DATA\_SIZE];

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

{

LinkRx\_Init\_type LRx;

int LInitSt;

**int** RxStat, \*ptrDataRx;

**int** error\_ct , i;

LRx.CheckSum = *Link\_CheckSum\_Dis*;

LRx.CheckBCMPI = *Link\_CheckBCMP\_Dis*;

LRx.OVRIntEn = *Link\_OvrIT\_Dis*;

LRx.TOIntEn = *Link\_TOIT\_Dis*;

LRx.DataSize = *Link\_DataSize\_1bit*;

//Link Enable

LInitSt = HAL\_LinkRx\_Enable(LINK\_NUM, &LRx, 0);

**if** (LInitSt == -1) {error\_ct++;}

//Receive Data

ptrDataRx = (**int**)&data\_rx32;

**for**(i=0;i<DATA\_SIZE/4;i++){

RxStat = HAL\_LinkCore\_Receive(LINK\_NUM, &ptrDataRx, 4);

ptrDataRx+=4;

**if** (RxStat != 0) {error\_ct++;}

}

**return** 0;

}

## **HAL\_LinkCore\_Transmit**

**Резюме**

int32\_t **HAL\_LinkCore\_Transmit** ( uint32\_t link\_num, **void** \*BuffPtr, uint32\_t size )

uint32\_t link\_num – номер порта связи. Принимает значение 0 либо 1.

**void** \*BuffPtr – указатель на массив передаваемых данных, который должен быть выровнен на границу квадрослова.

uint32\_t size – размер передаваемых данных (в словах, т.е. должен быть кратен 4)

**Описание**

Функция осуществляет передачу данных из BuffPtr, по средствам записи в буфер передатчика порта связи. В случаи установления бита ошибки в регистре статуса передатчика, функция прерывается и возвращает значение -1. В случае успешной передачи данных, функция возвращает 0.

**Пример**

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

**#define** LINK\_NUM 0

**#define** DATA\_SIZE 1024

**int** **\_\_attribute**((aligned(4 \* (32/\_\_CHAR\_BIT\_\_)))) data\_tx32[DATA\_SIZE];

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

{

LinkTx\_Init\_type LTx;

int LInitSt;

**int** TxStat, ptrDataTx;

**int** error\_ct , i;

LTx.CheckBCMPO = *Link\_CheckBCMP\_Dis*;

LTx.CheckSum = *Link\_CheckSum\_Dis*;

LTx.TOIntEn = *Link\_TOIT\_Dis*;

LTx.ClkSource = *Link\_TxClk\_Master*;

LTx.DataSize = *Link\_DataSize\_4bit*;

//Link Enable

LInitSt = HAL\_LinkTx\_Enable(LINK\_NUM, &LTx, 0);

**if** (LInitSt == -1) {error\_ct++;}

//Transmit Data

ptrDataTx = (**int**)&data\_tx32;

**for**(i=0;i<DATA\_SIZE/4;i++){

TxStat = HAL\_LinkCore\_Transmit(LINK\_NUM, &ptrDataTx, 4);

ptrDataTx+=4;

**if** (TxStat != 0) {error\_ct++;}

}

**return** 0;

}

## **Пример использования**

**#include** "hal\_1967VN044.h"

**#include** <stdint.h>

**#include** <stdlib.h>

**#define** LINK\_CLK 200000

**#define** LINK\_RX 1

**#define** LINK\_TX 0

**#define** DMA\_CH\_Rx (LINK\_RX + 8)

**#define** DMA\_CH\_Tx (LINK\_TX + 4)

**#define** SAMPLE\_SIZE 128

**#define** DMA\_WIDTH 4

**#define** LINK\_Rx\_GPIO\_Pins ( (GPIO\_PIN\_24 | GPIO\_PIN\_25 ) << (LINK\_RX\*4) )

**#define** LINK\_Tx\_GPIO\_Pins ( (GPIO\_PIN\_26 | GPIO\_PIN\_27 ) << (LINK\_TX\*4) )

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

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

**int** **\_\_attribute**((aligned(4 \* (32/\_\_CHAR\_BIT\_\_)))) tx\_buff[SAMPLE\_SIZE];

**int** **\_\_attribute**((aligned(4 \* (32/\_\_CHAR\_BIT\_\_)))) rx\_buff[SAMPLE\_SIZE];

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

{

LinkRx\_Init\_type LRx;

LinkTx\_Init\_type LTx;

**int** i;

**for** (i = 0; i < SAMPLE\_SIZE; i++)

tx\_buff[i] = **rand**() % 100;

HAL\_PLL\_LinkSetup(LINK\_CLK);

HAL\_DMA\_Stop(DMA\_CH\_Rx);

HAL\_DMA\_Stop(DMA\_CH\_Tx);

HAL\_LinkRx\_Disable(LINK\_RX);

HAL\_LinkTx\_Disable(LINK\_TX);

HAL\_GPIO\_Init(LX\_GPIO\_PC, LINK\_Rx\_GPIO\_Pins, *GPIO\_PinMode\_Alt*);

tcb\_rx[0] = (uint32\_t)&rx\_buff;

tcb\_rx[1] = (SAMPLE\_SIZE<<16) | DMA\_WIDTH;

tcb\_rx[2] = 0;

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

tcb\_tx[0] = (uint32\_t)&tx\_buff;

tcb\_tx[1] = (SAMPLE\_SIZE<<16) | DMA\_WIDTH;

tcb\_tx[2] = 0;

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

LRx.CheckSum = *Link\_CheckSum\_Dis*;

LRx.CheckBCMPI = *Link\_CheckBCMP\_Dis*;

LRx.OVRIntEn = *Link\_OvrIT\_Dis*;

LRx.TOIntEn = *Link\_TOIT\_Dis*;

LRx.DataSize = *Link\_DataSize\_4bit*;

LTx.CheckBCMPO = *Link\_CheckBCMP\_Dis*;

LTx.CheckSum = *Link\_CheckSum\_Dis*;

LTx.TOIntEn = *Link\_TOIT\_Dis*;

LTx.ClkSource = *Link\_TxClk\_Master*;

LTx.DataSize = *Link\_DataSize\_4bit*;

HAL\_LinkTx\_Enable(LINK\_TX, &LTx, 0);

HAL\_LinkRx\_Enable(LINK\_RX, &LRx, 0);

HAL\_LinkDMA\_Receive(LINK\_RX, tcb\_rx);

HAL\_LinkDMA\_Transmit (LINK\_TX, tcb\_tx);

HAL\_DMA\_WaitForChannel(DMA\_CH\_Tx);

HAL\_DMA\_WaitForChannel(DMA\_CH\_Rx);

**while**(1);

**return** 0;

}