# **Описание используемых типов**

## **ARINC\_RX\_type**

|  |  |
| --- | --- |
| **Имя поля** | **Описание** |
| RX\_CONTROL | Регистр контроля |
| RSV | Резерв |
| RX\_DATA | Регистр данных |
| RX\_STATUS | Регистр статуса |

## **ARINC\_TX\_type**

|  |  |
| --- | --- |
| **Имя поля** | **Описание** |
| TX\_CONTROL | Регистр контроля |
| RSV | Резерв |
| TX\_DATA | Регистр данных |
| TX\_STATUS | Регистр статуса |

## **IT type**

**typedef** **enum**

{

*ARINC\_RX\_IT\_nEmptyFIFO*, - В FIFO есть данные

*ARINC\_RX\_IT\_Err*, - обнаружена ошибка

*ARINC\_RX\_IT\_HalfFIFO*, - FIFO наполовину полно

*ARINC\_RX\_IT\_FullFIFO -* FIFO полно

} ARINC\_RX\_IT;

**typedef** **enum**

{

*ARINC\_TX\_IT\_EmptyFIFO*, - FIFO пусто

*ARINC\_TX\_IT\_HalfFIFO*, - FIFO наполовину полно

*ARINC\_TX\_IT\_FullFIFO -* FIFO полно

} ARINC\_TX\_IT;

## **ARINC\_RX\_Init\_type**

|  |  |  |
| --- | --- | --- |
| **Имя поля** | **Описание** | **Значение** |
| Clk | Частота на который будет работать ARINC | *ARINC\_Clk\_100kHz* – ARINC работает на частоте 100кГц  *ARINC\_Clk\_12p5kHz* – ARINC работает на частоте 12,5кГц |
| LB | Управление обнаружением меток | *ARINC\_LB\_Dis* – запретить обнаружение меток  *ARINC\_LB\_En –* разрешить обнаружение меток |
| SD | Управление декодированием бит данных 9 и 10 | *ARINC\_SD\_Dis* – запретить декодирование бит данных 9 и 10  *ARINC\_SD\_En –* разрешить декодирование бит данных 9 и 10 |
| DA | Управление прямым доступом | *ARINC\_LB\_Dis* – обычный режим работы FIFO  *ARINC\_LB\_En –* память приема канала работает не в режиме FIFO, доступ к ней осуществляется в диапазоне адресов в зависимости от номера канала |
| SDI1 | Корректное значение бита SDI1 принимаемых данных | *ARINC\_SDI\_Null –* бит равен нулю  *ARINC\_SDI\_One –* бит равен единице |
| SDI2 | Корректное значение бита SDI2 принимаемых данных | *ARINC\_SDI\_Null –* бит равен нулю  *ARINC\_SDI\_One –* бит равен единице |
| Sync | Управление работы входов приемника в режиме данных и синхросигнала | *ARINC\_Sync\_Dis –* запрещено, обычный режим работы*.*  *ARINC\_Sync\_En -* разрешено, для соответствующего канала вход IN\_A работает как данные (D), вход IN\_B как синхросигнал (SYN) |
| Parity | Управление четностью | *ARINC\_Parity\_Dis –* 32-битом передается бит данных  *ARINC\_Parity\_Odd –* 32-битом передается бит паритета. Бит паритета формируется как дополнение до нечетности (если сумма всех разрядов данных по модулю 2 равно нулю, бит паритета устанавливается в 1, в противном случае в 0)  *ARINC\_Parity\_Even -* 32-битом передается бит паритета. Бит паритета формируется как дополнение до четности (если сумма всех разрядов данных по модулю 2 равна единице, бит паритета устанавливается в 1, в противном случае в 0) |
| ITnEmptyFIFO | Управление запросом прерывания при не пустом приемнике | *ARINC\_IT\_Dis –* запрещено  *ARINC\_IT\_En -* разрешено |
| ITErr | Управление запросом прерывания при обнаружении ошибки | *ARINC\_IT\_Dis –* запрещено  *ARINC\_IT\_En -* разрешено |
| ITFullFIFO | Управление запросом прерывания заполоненном полностью FIFO | *ARINC\_IT\_Dis –* запрещено  *ARINC\_IT\_En -* разрешено |
| ITHalfFIFO | Управление запросом прерывания при заполненным на половину FIFO | *ARINC\_IT\_Dis –* запрещено  *ARINC\_IT\_En -* разрешено |

## **ARINC\_TX\_Init\_type**

|  |  |  |
| --- | --- | --- |
| **Имя поля** | **Описание** | **Значение** |
| Clk | Частота на который будет работать ARINC | *ARINC\_Clk\_100kHz* – ARINC работает на частоте 100кГц  *ARINC\_Clk\_12p5kHz* – ARINC работает на частоте 12,5кГц |
| Parity | Управление четностью | *ARINC\_Parity\_Dis –* 32-битом передается бит данных  *ARINC\_Parity\_Odd –* 32-битом передается бит паритета. Бит паритета формируется как дополнение до нечетности (если сумма всех разрядов данных по модулю 2 равно нулю, бит паритета устанавливается в 1, в противном случае в 0)  *ARINC\_Parity\_Even -* 32-битом передается бит паритета. Бит паритета формируется как дополнение до четности (если сумма всех разрядов данных по модулю 2 равна единице, бит паритета устанавливается в 1, в противном случае в 0) |
| Sync | Управление работы входов приемника в режиме данных и синхросигнала | *ARINC\_Sync\_Dis –* запрещено, обычный режим работы*.*  *ARINC\_Sync\_En -* разрешено, для соответствующего канала вход IN\_A работает как данные (D), вход IN\_B как синхросигнал (SYN) |
| ITnEmptyFIFO | Управление запросом прерывания при не пустом приемнике | *ARINC\_IT\_Dis –* запрещено  *ARINC\_IT\_En -* разрешено |
| ITFullFIFO | Управление запросом прерывания заполоненном полностью FIFO | *ARINC\_IT\_Dis –* запрещено  *ARINC\_IT\_En -* разрешено |
| ITHalfFIFO | Управление запросом прерывания при заполненным на половину FIFO | *ARINC\_IT\_Dis –* запрещено  *ARINC\_IT\_En -* разрешено |

## **Макросы**

|  |  |
| --- | --- |
| **Имя макроса** | **Описание** |
| LX\_ARINC\_RX\_CH0 - LX\_ARINC\_RX\_CH7 | Указатели на регистры каналов приёмника ARINC 0-7 |
| LX\_ARINC\_RX\_MEM0 - LX\_ARINC\_RX\_MEM7 | Указатели на буферы приёмников каналов 0-7 |
| LX\_ARINC\_RX\_LABEL0 - LX\_ARINC\_RX\_LABEL7 | Указатели на буферы меток каналов приемника 0-7 |
| LX\_ARINC\_TX\_CH0 - LX\_ARINC\_TX\_CH3 | Указатели на регистры каналов передатчика ARINC 0-7 |
| LX\_ARINC\_TX\_MEM0 - LX\_ARINC\_TX\_MEM7 | Указатели на буферы передатчиков каналов 0-7 |

# **Описание используемых функций**

## **HAL\_ARINC\_RX\_Init**

**Резюме**

**void** **HAL\_ARINC\_RX\_Init(** ARINC\_RX\_type \*LX\_ARINC\_RX\_CHx, ARINC\_RX\_Init\_type \*InitStruct **)**

ARINC\_RX\_type \*LX\_ARINC\_RX\_CHx – указатель на структуру с регистрами канала приёмника ARINC

ARINC\_RX\_Init\_type \*InitStruct – указатель на структуру инициализации приёмника

**Описание**

Функция инициализирует выбранный канал приемника ARINC в соответствии с InitStruct. Обращаю внимание на то, что для работы с приемником ARINC НЕ нужно включать альтернативные функции у GPIO. Для вычисления значения делителя используется частота системной шины. При старте системы она равняется частоте XTI\_KHZ, определенной в файле hal\_pll.h. При настройке частоты ядра функцией HAL\_PLL\_CoreSetup все частоты будут рассчитаны и записаны в структуру PLL\_Freq.

**Пример**

#include <hal\_1967VN044.h>

int main(void)

{

//PLL Configure

HAL\_PLL\_CoreSetup(CPU\_CLK);

HAL\_PLL\_BusSetup(BUS\_CLK);

// RX Configure

RxInit.Clk = *ARINC\_Clk\_100kHz*;

RxInit.DA = *ARINC\_DA\_Dis*;

RxInit.LB = *ARINC\_LB\_En*;

RxInit.Parity = *ARINC\_Parity\_Dis*;

RxInit.SD = *ARINC\_SD\_En*;

RxInit.SDI1 = *ARINC\_SDI\_One*;

RxInit.SDI2 = *ARINC\_SDI\_Null*;

RxInit.Sync = *ARINC\_Sync\_Dis*;

RxInit.ITErr = *ARINC\_IT\_En*;

RxInit.ITnEmptyFIFO = *ARINC\_IT\_Dis*;

RxInit.ITHalfFIFO = *ARINC\_IT\_Dis*;

RxInit.ITFullFIFO = *ARINC\_IT\_Dis*;

// RX Init

HAL\_ARINC\_RX\_Init(LX\_ARINC\_RX\_CH0, &RxInit);

return 0;

}

## **HAL\_ARINC\_RX\_DeInit**

**Резюме**

**void** **HAL\_ARINC\_RX\_DeInit(** ARINC\_RX\_type \*LX\_ARINC\_RX\_CHx **)**

ARINC\_RX\_type \*LX\_ARINC\_RX\_CHx – указатель на структуру с регистрами канала приёмника ARINC

**Описание**

Функции записывает 0 в регистр контроля выбранного канала ARINC.

**Пример**

#include <hal\_1967VN044.h>

int main(void)

{

HAL\_ARINC\_RX\_DeInit(LX\_ARINC\_RX\_CH7);

return 0;

}

## **HAL\_ARINC\_RX\_ITEn**

**Резюме**

**void HAL\_ARINC\_RX\_ITEn(** ARINC\_RX\_type \*LX\_ARINC\_RX\_CHx, ARINC\_RX\_IT ITName **)**

ARINC\_RX\_type \*LX\_ARINC\_RX\_CHx – указатель на структуру с регистрами канала приёмника ARINC

ARINC\_RX\_IT ITName – разрешаемое прерывание.

**Описание**

Функция разрешает запросы прерывания в соответствии со значением ITName. Так же прерывания можно разрешить или запретить при инициализации ARINC (см. HAL\_ARINC\_RX\_Init).

**Пример**

#include <hal\_1967VN044.h>

int main(void)

{

HAL\_ARINC\_RX\_ITEn(LX\_ARINC\_RX\_CH4, ARINC\_RX\_IT\_Err);

return 0;

}

## **HAL\_ARINC\_RX\_ITDis**

**Резюме**

**void** **HAL\_ARINC\_RX\_ITDis(** ARINC\_RX\_type \*LX\_ARINC\_RX\_CHx, ARINC\_RX\_IT ITName **)**

ARINC\_RX\_type \*LX\_ARINC\_RX\_CHx – указатель на структуру с регистрами канала приёмника ARINC

ARINC\_RX\_IT ITName – запрещаемое прерывание

**Описание**

Функция запрещает запросы прерывания в соответствии со значением ITName. Так же прерывания можно разрешить или запретить при инициализации ARINC (см. HAL\_ARINC\_RX\_Init).

**Пример**

#include <hal\_1967VN044.h>

int main(void)

{

HAL\_ARINC\_RX\_ITDis(LX\_ARINC\_RX\_CH4, ARINC\_RX\_IT\_Err);

return 0;

}

## **HAL\_ARINC\_RX\_AllDataRead**

**Резюме**

**uint32\_t** **HAL\_ARINC\_RX\_AllDataRead(** ARINC\_RX\_type \*LX\_ARINC\_RX\_CHx, **void** \*PtrRxArr **)**

ARINC\_RX\_type \*LX\_ARINC\_RX\_CHx – указатель на структуру с регистрами канала приёмника ARINC

**void** \*PtrRxArr – указатель на начало массива, в который будут записаны прочитанные данные.

**Описание**

Функция читает данные из буфера приемника до тех пор, пока установлен бит DR регистра RX\_STATUS (то есть до тех пор пока в FIFO есть данные). Возвращает количество прочитанных данных.

**Пример**

#include <hal\_1967VN044.h>

int main(void)

{

unsigned int count = 0, data[14];

count+=HAL\_ARINC\_RX\_AllDataRead(LX\_ARINC\_RX\_CH3, &data[0]);

return 0;

}

## **HAL\_ARINC\_RX\_MDataRead**

**Резюме**

**void** **HAL\_ARINC\_RX\_MDataRead(** uint32\_t \*LX\_ARINC\_RX\_MEMx, **void** \*PtrRxArr, uint32\_t Size **)**

uint32\_t \*LX\_ARINC\_RX\_MEMx – указатель на буфер приемника

**void** \*PtrRxArr – указатель на начало массива, в который будут записаны прочитанные данные.

uint32\_t Size – количество прочитанных слов.

**Описание**

Функция читает Size слов из буфера приемника по указателю LX\_ARINC\_RX\_MEMx.

**Пример**

#include <hal\_1967VN044.h>

#define N 7

int main(void)

{

unsigned int data[14];

HAL\_ARINC\_RX\_MDataRead(LX\_ARINC\_RX\_MEM6, &data[0], N);

return 0;

}

## **HAL\_ARINC\_RX\_DataRead**

**Резюме**

**void HAL\_ARINC\_RX\_DataRead(** ARINC\_RX\_type\*LX\_ARINC\_RX\_CHx**, void** \*PtrRxArr**,** uint32\_tSize **)**

ARINC\_RX\_type \*LX\_ARINC\_RX\_CHx – указатель на структуру с регистрами канала приёмника ARINC

**void** \*PtrRxArr – указатель на начало массива, в который будут записаны прочитанные данные.

uint32\_t Size – количество прочитанных слов.

**Описание**

Функция читает Size слов из FIFO приёмника (тое есть из регистра RX\_DATA в который данные попадают из буфера приёмника данного канала по указателю, значение которого отображается в регистре RX\_STATUS в поле DATA\_RP). Значение указателя инкрементируется автоматически.

**Пример**

#include <hal\_1967VN044.h>

#define N 17

int main(void)

{

unsigned int data[14];

HAL\_ARINC\_RX\_DataRead(LX\_ARINC\_RX\_MEM3, &data[0], N);

return 0;

}

## **HAL\_ARINC\_RX\_Enable**

**Резюме**

**void HAL\_ARINC\_RX\_Enable(** ARINC\_RX\_type \*LX\_ARINC\_RX\_CHx **)**

ARINC\_RX\_type \*LX\_ARINC\_RX\_CHx – указатель на структуру с регистрами канала приёмника ARINC

**Описание**

Функция включает выбранный канала приемника ARINC (устанавливает бит CH\_EN регистра RX\_CONTROL), и разрешает прием данных.

**Пример**

#include <hal\_1967VN044.h>

int main(void)

{

HAL\_ARINC\_RX\_Enable(LX\_ARINC\_RX\_CH1);

return 0;

}

## **HAL\_ARINC\_RX\_Disable**

**Резюме**

**void HAL\_ARINC\_RX\_Disable(** ARINC\_RX\_type \*LX\_ARINC\_RX\_CHx **)**

ARINC\_RX\_type \*LX\_ARINC\_RX\_CHx– указатель на структуру с регистрами канала приёмника ARINC

**Описание**

Функция выключает выбранный канала приемника ARINC (сбрасывает бит CH\_EN регистра RX\_CONTROL), и запрещает прием данных.

**Пример**

#include <hal\_1967VN044.h>

int main(void)

{

HAL\_ARINC\_RX\_Disable(LX\_ARINC\_RX\_CH2);

return 0;

}

## **HAL\_ARINC\_RX\_LabelSet**

**Резюме**

**void HAL\_ARINC\_RX\_LabelSet(** uint32\_t \*LX\_ARINC\_RX\_LABELx, uint32\_t \*CorrectLabel, uint32\_t Size **)**

uint32\_t \*LX\_ARINC\_RX\_LABELx – указатель на поле меток выбранного канала ARINC

uint32\_t \*CorrectLabel – указатель на корректные метки (т.е данные с эти метками будут считаться корректными и будут приняты)

uint32\_t Size – количество меток.

**Описание**

Функция устанавливает корректными Size меток из массива CorrectLabel для выбранного канала приемника ARINC. Перед первым использованием функции необходимо сбросить поле меток выбранного канала, т.к по умолчанию там лежит случайное значение. При повторном вызове данной функции для того же самого канала уже установленные метки НЕ будут сброшены. Для сброса всех меток используйте функция HAL\_ARINC\_RX\_LabelAllReset, для сброса определённых меток HAL\_ARINC\_RX\_LabelReset.

**Пример**

#include <hal\_1967VN044.h>

#define LN 5

int main(void)

{

**unsigned** **int** CorrectLabel[LN];

**for**(i=0;i<LN;i++){

CorrectLabel[i] = rand()&0xFF;

}

//Label Init

HAL\_ARINC\_RX\_LabelAllReset(LX\_ARINC\_RX\_LABELx);

HAL\_ARINC\_RX\_LabelSet(LX\_ARINC\_RX\_LABELx, &CorrectLabel[0], LN);

return 0;

}

## **HAL\_ARINC\_RX\_LabelReset**

**Резюме**

**void HAL\_ARINC\_RX\_LabelReset(** uint32\_t \*LX\_ARINC\_RX\_LABELx, uint32\_t \*IncorrectLabel, uint32\_t Size **)**

uint32\_t \*LX\_ARINC\_RX\_LABELx – указатель на поле меток выбранного канала ARINC

uint32\_t \*IncorrectLabel – указатель на некорректные метки (т.е данные с эти метками не будут считаться корректными, и не будут приняты)

uint32\_t Size – количество меток.

**Описание**

Функция устанавливает некорректными (сбрасывает) Size меток из массива IncorrectLabel для выбранного канала приемника ARINC.

**Пример**

#include <hal\_1967VN044.h>

#define LN 12

int main(void)

{

**unsigned** **int** InCorrectLabel[LN];

**for**(i=0;i<LN;i++){

InCorrectLabel[i] = rand()&0xFF;

}

//Label DeInit

HAL\_ARINC\_RX\_LabelReset(LX\_ARINC\_RX\_LABELx, &InCorrectLabel[0], LN);

return 0;

}

## **HAL\_ARINC\_RX\_LabelAllReset**

**Резюме**

**void HAL\_ARINC\_RX\_LabelAllReset(** uint32\_t \*LX\_ARINC\_RX\_LABELx **)**

uint32\_t \*LX\_ARINC\_RX\_LABELx – указатель на поле меток выбранного канала ARINC

**Описание**

Функция сбрасывает (объявляет некорректными) все метки для выбранного канала ARINC.

**Пример**

#include <hal\_1967VN044.h>

int main(void)

{

HAL\_ARINC\_RX\_LabelAllReset(LX\_ARINC\_RX\_LABEL5);

return 0;

}

## **HAL\_ARINC\_TX\_Init**

**Резюме**

**void HAL\_ARINC\_TX\_Init(** ARINC\_TX\_type \*LX\_ARINC\_TX\_CHx, ARINC\_TX\_Init\_type \*InitStruct **)**

ARINC\_TX\_type \*LX\_ARINC\_TX\_CHx – указатель на структуру с регистрами передатчика ARINC

ARINC\_TX\_Init\_type \*InitStruct– указатель на структуру инициализации передатчика ARINC

**Описание**

Функция инициализирует выбранный канал передатчика ARINC в соответствии с InitStruct. Обращаю внимание на то, что для работы с передатчиком ARINC нужно включать альтернативные функции у GPIO и установить бит, в соответствии с выбранным каналом, в регистре CFG1 из блока CMU. Для вычисления значения делителя используется частота системной шины. При старте системы она равняется частоте XTI\_KHZ, определенной в файле hal\_pll.h. При настройке частоты ядра функцией HAL\_PLL\_CoreSetup все частоты будут рассчитаны и записаны в структуру PLL\_Freq.

**Пример**

#include <hal\_1967VN044.h>

int main(void)

{

// PLL Configure

HAL\_PLL\_CoreSetup(CPU\_CLK);

HAL\_PLL\_BusSetup(BUS\_CLK);

// GPIO Alt Func Enable

HAL\_GPIO\_Init(LX\_GPIO\_PB, GPIO\_PIN\_20| GPIO\_PIN\_21, *GPIO\_PinMode\_Alt*);

// TX CMU Enabmle

LX\_CMU->CFG1.b.ARINC\_T0\_EN = 1;

// TX Configure

TxInit.Clk = *ARINC\_Clk\_100kHz*;

TxInit.Parity = *ARINC\_Parity\_Dis*;

TxInit.Sync = *ARINC\_Sync\_Dis*;

TxInit.ITEmptyFIFO = *ARINC\_IT\_Dis*;

TxInit.ITFullFIFO = *ARINC\_IT\_Dis*;

TxInit.ITHalfFIFO = *ARINC\_IT\_Dis*;

// TX Init

HAL\_ARINC\_TX\_Init(LX\_ARINC\_TX\_CH0, &TxInit, PLL\_Freq.CoreClk/2);

// TX Start

LX\_ARINC\_TX\_CH0->TX\_CONTROL.b.CH\_EN = 1;

return 0;

}

## **HAL\_ARINC\_TX\_DeInit**

**Резюме**

**void HAL\_ARINC\_TX\_DeInit(** ARINC\_TX\_type \*LX\_ARINC\_TX\_CHx **)**

ARINC\_TX\_type \*LX\_ARINC\_TX\_CHx– указатель на структуру с регистрами передатчика ARINC

**Описание**

Функция записывает ноль в регистр TX\_CONTROL выбранного канала ARINC

**Пример**

#include <hal\_1967VN044.h>

int main(void)

{

HAL\_ARINC\_TX\_DeInit(LX\_ARINC\_TX\_CH1);

return 0;

}

## **HAL\_ARINC\_TX\_ITEn**

**Резюме**

**void HAL\_ARINC\_TX\_ITEn(** ARINC\_TX\_type \*LX\_ARINC\_TX\_CHx, ARINC\_TX\_IT ITName **)**

ARINC\_TX\_type \*LX\_ARINC\_TX\_CHx CHx – указатель на структуру с регистрами передатчика ARINC

ARINC\_TX\_IT ITName– имя прерывания

**Описание**

Функция разрешает прерывание в соответствии с ITName. Так же прерывания можно разрешить или запретить при инициализации ARINC (см. HAL\_ARINC\_TX\_Init).

**Пример**

#include <hal\_1967VN044.h>

int main(void)

{

HAL\_ARINC\_TX\_ITEn(LX\_ARINC\_TX\_CH2, *ARINC\_TX\_IT\_HalfFIFO*);

return 0;

}

## **HAL\_ARINC\_TX\_ITDis**

**Резюме**

**void HAL\_ARINC\_TX\_ITDis(** ARINC\_TX\_type \*LX\_ARINC\_TX\_CHx, ARINC\_TX\_IT ITName **)**

ARINC\_TX\_type \*LX\_ARINC\_TX\_CHx – указатель на структуру с регистрами передатчика ARINC

ARINC\_TX\_IT ITName – имя прерывания

**Описание**

Функция запрещает прерывание в соответствии с ITName. Так же прерывания можно разрешить или запретить при инициализации ARINC (см. HAL\_ARINC\_TX\_Init).

**Пример**

#include <hal\_1967VN044.h>

int main(void)

{

HAL\_ARINC\_TX\_ITDis(LX\_ARINC\_TX\_CH3, *ARINC\_TX\_IT\_EmptyFIFO*);

return 0;

}

## **HAL\_ARINC\_TX\_DataWrite**

**Резюме**

**void HAL\_ARINC\_TX\_DataWrite(** ARINC\_TX\_type \*LX\_ARINC\_TX\_CHx, **void** \*PtrTxArr, uint32\_t Size **)**

ARINC\_TX\_type \*LX\_ARINC\_TX\_CHx – указатель на структуру с регистрами передатчика ARINC

**void** \*PtrTxArr – указатель на начало массива с передаваемыми данными

uint32\_t Size– количество передаваемых данных

**Описание**

Функция записывает данные в буфер передатчика через регистр TX\_DATA, при установленном бите CH\_EN регистра TX\_CONTROL данные начинают передаваться.

**Пример**

#include <hal\_1967VN044.h>

#define N 28

int main(void)

{

unsigned int data\_tx[28];

// TX Write Data

HAL\_ARINC\_TX\_DataWrite(LX\_ARINC\_TX\_CH0, &data\_tx[0], N);

return 0;

}

## **HAL\_ARINC\_TX\_Enable**

**Резюме**

**void HAL\_ARINC\_TX\_Enable(** ARINC\_TX\_type \*LX\_ARINC\_TX\_CHx **)**

ARINC\_TX\_type \*LX\_ARINC\_TX\_CHx– указатель на структуру с регистрами передатчика ARINC

**Описание**

Функция включает выбранный канал передатчика ARINC и разрешает передачу данных (устанавливает бит CH\_EN регистра TX\_CONTROL).

**Пример**

#include <hal\_1967VN044.h>

int main(void)

{

HAL\_ARINC\_TX\_Enable(LX\_ARINC\_TX\_CH1);

return 0;

}

## **HAL\_ARINC\_TX\_Disable**

**Резюме**

**void HAL\_ARINC\_TX\_Disable(** ARINC\_TX\_type \*LX\_ARINC\_TX\_CHx **)**

ARINC\_TX\_type \*LX\_ARINC\_TX\_CHx– указатель на структуру с регистрами передатчика ARINC

**Описание**

Функция выключает выбранный канал передатчика ARINC и запрещает передачу данных (сбрасывает бит CH\_EN регистра TX\_CONTROL).

**Пример**

#include <hal\_1967VN044.h>

int main(void){

HAL\_ARINC\_TX\_Disable(LX\_ARINC\_TX\_CH2);

return 0;

}

# Примеры работы с ARINC

**#define** CPU\_CLK 200000

**#define** BUS\_CLK 50000

**#define** N 255

**#define** LN 7

**#define** RX\_EN

**#define** TX\_EN

**#define** CHECK\_DATA

**#define** DATA\_MAKE\_EN

**#define** ARINC\_RX 0

**#define** ARINC\_TX 0

**#if** ARINC\_RX==0

**#define** LX\_ARINC\_RX\_CHx LX\_ARINC\_RX\_CH0

**#define** LX\_ARINC\_RX\_MEMx LX\_ARINC\_RX\_MEM0

**#define** GPIO\_PIN\_RX GPIO\_PIN\_4|GPIO\_PIN\_5

**#define** LX\_ARINC\_RX\_LABELx LX\_ARINC\_RX\_LABEL0

**#elif** ARINC\_RX==1

**#define** LX\_ARINC\_RX\_CHx LX\_ARINC\_RX\_CH1

**#define** LX\_ARINC\_RX\_MEMx LX\_ARINC\_RX\_MEM1

**#define** GPIO\_PIN\_RX GPIO\_PIN\_6|GPIO\_PIN\_7

**#define** LX\_ARINC\_RX\_LABELx LX\_ARINC\_RX\_LABEL1

**#elif** ARINC\_RX==2

**#define** LX\_ARINC\_RX\_CHx LX\_ARINC\_RX\_CH2

**#define** LX\_ARINC\_RX\_MEMx LX\_ARINC\_RX\_MEM2

**#define** GPIO\_PIN\_RX GPIO\_PIN\_8|GPIO\_PIN\_9

**#define** LX\_ARINC\_RX\_LABELx LX\_ARINC\_RX\_LABEL2

**#elif** ARINC\_RX==3

**#define** LX\_ARINC\_RX\_CHx LX\_ARINC\_RX\_CH3

**#define** LX\_ARINC\_RX\_MEMx LX\_ARINC\_RX\_MEM3

**#define** GPIO\_PIN\_RX GPIO\_PIN\_10|GPIO\_PIN\_11

**#define** LX\_ARINC\_RX\_LABELx LX\_ARINC\_RX\_LABEL3

**#elif** ARINC\_RX==4

**#define** LX\_ARINC\_RX\_CHx LX\_ARINC\_RX\_CH4

**#define** LX\_ARINC\_RX\_MEMx LX\_ARINC\_RX\_MEM4

**#define** GPIO\_PIN\_RX GPIO\_PIN\_12|GPIO\_PIN\_13

**#define** LX\_ARINC\_RX\_LABELx LX\_ARINC\_RX\_LABEL4

**#elif** ARINC\_RX==5

**#define** LX\_ARINC\_RX\_CHx LX\_ARINC\_RX\_CH5

**#define** LX\_ARINC\_RX\_MEMx LX\_ARINC\_RX\_MEM5

**#define** GPIO\_PIN\_RX GPIO\_PIN\_14|GPIO\_PIN\_15

**#define** LX\_ARINC\_RX\_LABELx LX\_ARINC\_RX\_LABEL5

**#elif** ARINC\_RX==6

**#define** LX\_ARINC\_RX\_CHx LX\_ARINC\_RX\_CH6

**#define** LX\_ARINC\_RX\_MEMx LX\_ARINC\_RX\_MEM6

**#define** GPIO\_PIN\_RX GPIO\_PIN\_16|GPIO\_PIN\_17

**#define** LX\_ARINC\_RX\_LABELx LX\_ARINC\_RX\_LABEL6

**#elif** ARINC\_RX==7

**#define** LX\_ARINC\_RX\_CHx LX\_ARINC\_RX\_CH7

**#define** LX\_ARINC\_RX\_MEMx LX\_ARINC\_RX\_MEM7

**#define** GPIO\_PIN\_RX GPIO\_PIN\_18|GPIO\_PIN\_19

**#define** LX\_ARINC\_RX\_LABELx LX\_ARINC\_RX\_LABEL7

**#endif**

**#if** ARINC\_TX==0

**#define** LX\_ARINC\_TX\_CHx LX\_ARINC\_TX\_CH0

**#define** LX\_ARINC\_TX\_MEMx LX\_ARINC\_TX\_MEM0

**#define** GPIO\_PIN\_TX GPIO\_PIN\_20|GPIO\_PIN\_21

**#define** ARINC\_TX\_EN ARINC\_T0\_EN

**#elif** ARINC\_TX==1

**#define** LX\_ARINC\_TX\_CHx LX\_ARINC\_TX\_CH1

**#define** LX\_ARINC\_TX\_MEMx LX\_ARINC\_TX\_MEM1

**#define** GPIO\_PIN\_TX GPIO\_PIN\_26|GPIO\_PIN\_27

**#define** ARINC\_TX\_EN ARINC\_T1\_EN

**#elif** ARINC\_TX==2

**#define** LX\_ARINC\_TX\_CHx LX\_ARINC\_TX\_CH2

**#define** LX\_ARINC\_TX\_MEMx LX\_ARINC\_TX\_MEM2

**#define** GPIO\_PIN\_TX GPIO\_PIN\_28|GPIO\_PIN\_29

**#define** ARINC\_TX\_EN ARINC\_T2\_EN

**#elif** ARINC\_TX==3

**#define** LX\_ARINC\_TX\_CHx LX\_ARINC\_TX\_CH3

**#define** LX\_ARINC\_TX\_MEMx LX\_ARINC\_TX\_MEM3

**#define** GPIO\_PIN\_TX GPIO\_PIN\_30|GPIO\_PIN\_31

**#define** ARINC\_TX\_EN ARINC\_T3\_EN

**#endif**

**#ifdef** DATA\_MAKE\_EN

**#pragma** align 64

ARINC\_Word\_type data\_tx[N];

**#endif**

**#pragma** align 64

ARINC\_Word\_type data\_rx[N];

**void** **Error**(**int** ErrCode){

\_\_NOP;

**while**(1);

}

**#pragma** interrupt

**void** **ARINC\_RX\_HNDLR**(**void**){

Error(1000);

}

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

**int** i, j, tmp;

ARINC\_RX\_Init\_type RxInit;

ARINC\_TX\_Init\_type TxInit;

**unsigned** **int** CorrectLabel[LN];

//Create Label

**for**(i=0;i<LN;i++){

CorrectLabel[i] = rand()&0xFF;

}

//Interrupt Clear and Global Enable

HAL\_Interrupt\_IMASKClear();

HAL\_Interrupt\_ILATClear();

HAL\_Interrupt\_GlobalEnable();

\_\_RDS;

//Flag Enable

HAL\_SYS\_FlagEnable();

//PLL Configure

HAL\_PLL\_CoreSetup(CPU\_CLK);

HAL\_PLL\_BusSetup(BUS\_CLK);

// IT Configure

HAL\_Interrupt\_Enable(*intARINC\_RX*, ARINC\_RX\_HNDLR);

**#ifdef** TX\_EN

// TX GPIO Init

HAL\_GPIO\_Init(LX\_GPIO\_PB, GPIO\_PIN\_TX, *GPIO\_PinMode\_Alt*);

**#endif**

**#ifdef** DATA\_MAKE\_EN

j = 0;

**for**(i=0;i<N;i++){

data\_tx[i].b.DATA = rand()&0x7FFFF;

data\_tx[i].b.LABEL = CorrectLabel[j];

data\_tx[i].b.SDI = 1;

data\_tx[i].b.SSM = rand()&0x3;

**if** (j==(LN-1)) j = 0;

**else** j++;

}

**#endif**

**#ifdef** RX\_EN

// RX Configure

RxInit.Clk = *ARINC\_Clk\_100kHz*;

RxInit.DA = *ARINC\_DA\_Dis*;

RxInit.LB = *ARINC\_LB\_En*;

RxInit.Parity = *ARINC\_Parity\_Dis*;

RxInit.SD = *ARINC\_SD\_En*;

RxInit.SDI1 = *ARINC\_SDI\_One*;

RxInit.SDI2 = *ARINC\_SDI\_Null*;

RxInit.Sync = *ARINC\_Sync\_Dis*;

RxInit.ITErr = *ARINC\_IT\_En*;

RxInit.ITnEmptyFIFO = *ARINC\_IT\_Dis*;

RxInit.ITHalfFIFO = *ARINC\_IT\_Dis*;

RxInit.ITFullFIFO = *ARINC\_IT\_Dis*;

// RX Init

HAL\_ARINC\_RX\_Init(LX\_ARINC\_RX\_CHx, &RxInit);

//Label Init

HAL\_ARINC\_RX\_LabelAllReset(LX\_ARINC\_RX\_LABELx);

HAL\_ARINC\_RX\_LabelSet(LX\_ARINC\_RX\_LABELx, &CorrectLabel[0], LN);

// RX Start

LX\_ARINC\_RX\_CHx->RX\_CONTROL.b.CH\_EN = 1;

**#endif**

**#ifdef** TX\_EN

// TX CMU Enabmle

LX\_CMU->CFG1.b.ARINC\_TX\_EN = 1;

// TX Configure

TxInit.Clk = *ARINC\_Clk\_100kHz*;

TxInit.Parity = *ARINC\_Parity\_Dis*;

TxInit.Sync = *ARINC\_Sync\_Dis*;

TxInit.ITEmptyFIFO = *ARINC\_IT\_Dis*;

TxInit.ITFullFIFO = *ARINC\_IT\_Dis*;

TxInit.ITHalfFIFO = *ARINC\_IT\_Dis*;

// TX Init

HAL\_ARINC\_TX\_Init(LX\_ARINC\_TX\_CHx, &TxInit);

// TX Start

LX\_ARINC\_TX\_CHx->TX\_CONTROL.b.CH\_EN = 1;

// TX Write Data

HAL\_ARINC\_TX\_DataWrite(LX\_ARINC\_TX\_CHx, &data\_tx[0], N);

**#endif**

**#ifdef** RX\_EN

**while**(LX\_ARINC\_RX\_CHx->RX\_STATUS.b.CHAN\_WP!=N);

HAL\_ARINC\_RX\_DataRead (LX\_ARINC\_RX\_CHx, &data\_rx[0], N);

**#endif**

**#ifdef** CHECK\_DATA

**for**(i=0;i<N;i++){

**if** (data\_tx[i].word!=data\_rx[i].word) Error(i);

}

**#endif**

**while**(1);

**return** 0;

}

**Комментарии:**

1. Наличие дефайна RX\_EN включает в программу работу с приемником, наличие дефайна TE\_EN включает в программу работу с передатчиком, наличие дефайнов CHECK\_DATA и DATA\_MAKE включает в программу проверку и создание случайных данных.
2. Значение дефайнов ARINC\_RX и ARINC\_TX определяет канал приемника/передатчика который будет настраивается.
3. Дефайн LN – количество корректных меток. Дефайн N – количество передаваемых/принимаемых данных. Обращаю внимание на то, что при прием количества данных больше чем размер буфера, указатель записи данных сбрасывается в ноль и начинает инкрементироваться заново.