# **Описание используемых типов**

## **RTC\_type**

|  |  |
| --- | --- |
| **Имя поля** | **Описание** |
| RTC\_CNT | Счетчик секунд |
| RTC\_MR | Регистр сравнения |
| WDT\_CNT | Текущее значение сторожевого счетчика |
| RTC\_TDIV | Текущее значение делителя тик-импульса |
| RTC\_CR | Регистр управления |
| RTC\_SDIV | Текущее значение делителя секунды |
| TIC\_VAL | Период тик-импульса минус 1. Max 20 bit (1048575) |
| SEC\_VAL | Количество тик-импульсов в секунде |
| RTC\_BUSY | Флаг занятости интерфейса |

# **Описание используемых функций**

## **HAL\_RTC\_InitTicPeriod**

**Резюме**

**void HAL\_RTC\_InitTicPeriod**( uint32\_t NumOfXTI )

uint32\_t NumOfXTI – период TIC импульсов в количестве клоков XTI

**Описание**

Функция устанавливает частоту срабатывания TIC импульсов, в клоках XTI. Счетчиком Тик-импульсов является регистр RTC\_SDIV. ВНИМАНИЕ! Функция является аналог для HAL\_RTC\_InitTicPeriodUS, с той разницей, что задает период в количествах клока XTI, а не в мкс как HAL\_RTC\_InitTicPeriodUS. Функция включает в себя ожидании завершение работы RTC.

**Пример**

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

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

{

HAL\_RTC\_InitTicPeriod(25); //Формирование Тик-импульса каждые 25 клоков XTI

**return** 0;

}

## **HAL\_RTC\_InitTicPeriodUS**

**Резюме**

uint32\_t **HAL\_RTC\_InitTicPeriodUS**( uint32\_t XTI\_khz, uint32\_t   
Period\_us )

uint32\_t XTI\_khz – XTI частота в килогерцах

uint32\_t Period\_us – период TIC импульсов в микросекундах

**Описание**

Функция устанавливает частоту срабатывания TIC импульсов, в микросекундах. Счетчиком Тик-импульсов является регистр RTC\_SDIV.

**Пример**

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

**#define** XTI 25000

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

{

**int** ticPerNs;

ticPerNs = HAL\_RTC\_InitTicPeriodUS(XTI, 25); //Формирование Тик-импульса каждые 25 мкс

**return** 0;

}

## **HAL\_RTC\_InitSecPeriod**

**Резюме**

**Void** **HAL\_RTC\_InitSecPeriod**( uint32\_t NumOfTic )

uint32\_t NumOfTic – частота псевдо-секунд в «тиках».

**Описание**

Функция устанавливает частоту срабатывания импульсов секунд, в TIC импульсах. Счетчиком импульсов секунд является регистр RTC\_CNT. Функция включает в себя ожидании завершение работы RTC.

**Пример**

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

**#define** XTI 25000

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

{

**int** ticPerNs;

ticPerNs = HAL\_RTC\_InitTicPeriodUS(XTI, 25); //Формирование Тик-импульса каждые 25 мкс

HAL\_RTC\_InitSecPeriod(4000); //Период импульсов псевдо-секунд = 10 милисекунд

**return** 0;

}

## **HAL\_RTC\_RegRead**

**Резюме**

uint32\_t **HAL\_RTC\_RegRead**( RTC\_Reg\_type RTCReg )

RTC\_Reg\_type RTCReg – имя регистра RTC который необходимо прочитать (*rtcCNT*, *rtcMR*, *rtcWDT*, *rtcTDIV*, *rtcCR*, *rtcSDIV*, *rtcTICVAL*, *rtcSECVAL*)

**Описание**

Функция делает корректное чтение регистров RTC(в соответствии со спецификацией, для корректного чтение регистров RTC необходимо сделать два чтение одного и того же регистра подряд). И возвращает значение прочитанного регистра. Функция включает в себя ожидании завершение работы RTC.

**Пример**

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

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

{

**int** tmp32;

tmp32 = HAL\_RTC\_RegRead(rtcCR); //чтение регистра RTC\_CR

**return** 0;

}

## **HAL\_RTC\_Busy**

**Резюме**

**void** **HAL\_RTC\_Busy**( **void** )

**Описание**

Функция не завершается пока RTC блок работает (в состоянии busy). Так как RTC работает на частоте XTI, а ядро может работать на частотах выше XTI, то после каждой записи/чтение регистров RTC необходимо ждать их завершения. Стоит отметить, что по ВСЕМУ функции HAL\_RTC имеют в конце проверку, того, что модуль RTC завершил работу. Т.е после функций HAL\_RTC не нужно использовать HAL\_RTC\_Busy

**Пример**

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

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

{

**int** tmp32;

LX\_RTC->RTC\_CR = 0x700;

HAL\_RTC\_Busy();

**return** 0;

}

## **HAL\_RTC\_AlarmIntEn**

**Резюме**

**void HAL\_RTC\_AlarmIntEn**( uint32\_t MR\_sec )

uint32\_t MR\_sec – время в псведо-секундах, ЧЕРЕЗ которое должен сработать будильник

**Описание**

Функция инициализирует будильник(alarm) на разрешение генерации запросов прерываний и устанавливает время, через которое должен быть сгенерирован запрос от будильника(alarm). ВНИМАНИЕ! Запрос на прерывание от будильника происходит ОДИН раз (до переполнения счетчика RTC\_CNT), а не через каждые MR\_sec.

**Пример**

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

**#define** XTI 25000

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

{

**int** tmp32;

**int** ticPerNs;

ticPerNs = HAL\_RTC\_InitTicPeriodUS(XTI, 25); //Формирование Тик-импульса //каждые 25 мкс

HAL\_RTC\_InitSecPeriod(4000);//Период импульсов псевдо-секунд = 10 //милисекунд

HAL\_RTC\_AlarmIntEn(5); //Запрос на прерывание от будильника случится //через 5 псевдо-секунд(50 милисекунд)

**return** 0;

}

## **HAL\_RTC\_AlarmIntEnExactTime**

**Резюме**

**void** **HAL\_RTC\_AlarmIntEnExactTime**( uint32\_t MRExact\_sec )

uint32\_t MRExact\_sec – точное время в псевдо-секунда, когда должен сработать будильник

**Описание**

Функция инициализирует будильник(alarm) на разрешение генерации запросов прерываний и устанавливает время, в которое должен быть сгенерирован запрос от будильника(alarm).

**Пример**

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

**#define** XTI 25000

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

{

**int** tmp32;

**int** ticPerNs;

ticPerNs = HAL\_RTC\_InitTicPeriodUS(XTI, 25); //Формирование Тик-импульса //каждые 25 мкс

HAL\_RTC\_InitSecPeriod(4000);//Период импульсов псевдо-секунд = 10 //милисекунд

HAL\_RTC\_AlarmIntEnExactTime (50); //Запрос на прерывание от будильника случится //на 50 псевдо секунде(т.е когда RTC-CNT == 50)

**return** 0;

}

## **HAL\_RTC\_TicIntEn**

**Резюме**

**void** **HAL\_RTC\_TicIntEn**( **void** )

**Описание**

Функция разрешает генерировать запросы на прерывания по приходу тик-импульса.

**Пример**

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

**#define** XTI 25000

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

{

**int** tmp32;

**int** ticPerNs;

ticPerNs = HAL\_RTC\_InitTicPeriodUS(XTI, 25); //Формирование Тик-импульса //каждые 25 мкс

HAL\_RTC\_TicIntEn();

**return** 0;

}

## **HAL\_RTC\_WDogIntEn**

**Резюме**

**void** **HAL\_RTC\_WDogIntEn**( uint32\_t WDT\_tic )

uint32\_t WDT\_tic – время в тиках, через которые watch dog таймер сгенерирует запрос

**Описание**

Функция разрешает генерацию запросов на прерывание от Watch Dog таймера

**Ограничения**

uint32\_tWDT\_tic – должно быть кратно степени двойки. Максимальное значение 128.

**Пример**

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

**#define** XTI 25000

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

{

**int** tmp32;

**int** ticPerNs;

ticPerNs = HAL\_RTC\_InitTicPeriodUS(XTI, 25); //Формирование Тик-импульса //каждые 25 мкс

HAL\_RTC\_WDogIntEn(32);//Запрос на прерывание от watch dog таймера //случится через 32 тика

**return** 0;

}

## **HAL\_RTC\_WDogHWResetEn**

**Резюме**

**void** **HAL\_RTC\_WDogHwResetEn**( uint32\_t WDT\_tic )

uint32\_t WDT\_tic – время в тиках, через которые watch dog таймер сгенерирует запрос

**Описание**

Функция разрешает сброс (hardware reset) от Watch Dog таймера.

**Ограничения**

uint32\_t WDT\_tic – должно быть кратно степени двойки. Максимальное значение 128.

**Пример**

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

**#define** XTI 25000

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

{

**int** tmp32;

**int** ticPerNs;

ticPerNs = HAL\_RTC\_InitTicPeriodUS(XTI, 25); //Формирование Тик-импульса //каждые 25 мкс

HAL\_RTC\_WDogHWResetEn(16);//Сброс от watch dog таймера случится через 16 //тиков

**return** 0;

}

## **HAL\_RTC\_AlarmIntDis**

**Резюме**

**void** **HAL\_RTC\_AlarmIntDis**( **void** )

**Описание**

Функция запрещает АСИНХРОННОГО прерывание от будильника(Alarm).

**Пример**

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

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

{

HAL\_RTC\_AlarmIntDis();

**return** 0;

}

## **HAL\_RTC\_TicIntDis**

**Резюме**

**void** **HAL\_RTC\_TicIntDis**( **void** )

**Описание**

Функция запрещает АСИНХРОННОЕ прерывание от тиков.

**Пример**

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

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

{

HAL\_RTC\_TicIntDis();

**return** 0;

}

## **HAL\_RTC\_WDogIntDis**

**Резюме**

**void** **HAL\_RTC\_WDogIntDis**( **void** )

**Описание**

Функция запрещает АСИНХРОННОЕ прерывание от watch dog таймера.

**Пример**

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

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

{

HAL\_RTC\_WDogIntDis();

**return** 0;

}

## **HAL\_RTC\_WDogHWResetDis**

**Резюме**

**void** **HAL\_RTC\_WDogHWResetDis**( **void** )

**Описание**

Функция запрещает сброс от watch dog таймера.

**Пример**

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

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

{

HAL\_RTC\_WDogHWResetDis();

**return** 0;

}

## **HAL\_RTC\_WDogLock**

**Резюме**

**void** **HAL\_RTC\_WDogLock**( **void** )

**Описание**

Функция блокирует сторожевой таймер (watch dog). После этого невозможно выключить сторожевой таймер.

**Пример**

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

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

{

HAL\_RTC\_WDogLock();

**return** 0;

}

## **HAL\_RTC\_WDogUnLock**

**Резюме**

**void** **HAL\_RTC\_WDogUnLock**( **void** )

**Описание**

Функция разблокирует сторожевой таймер. После этого, его можно выключить.

Функция не работает на 1967VN044.

**Пример**

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

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

{

HAL\_RTC\_WDogUnLock();

**return** 0;

}

## **HAL\_RTC\_Lock**

**Резюме**

**void** **HAL\_RTC\_Lock**( **void** )

**Описание**

Функция блокирует запись в регистры RTC.

**Пример**

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

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

{

HAL\_RTC\_Lock();

**return** 0;

}

## **HAL\_RTC\_UnLock**

**Резюме**

**void** **HAL\_RTC\_Lock**( **void** )

**Описание**

Функция разблокирует запись в регистры RTC.

Функция не работает на 1967VN044.

**Пример**

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

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

{

HAL\_RTC\_UnLock();

**return** 0;

}