DRIVER NAND SAMSUNG

## **NAND\_TypeDef**

**Описание**

Структура, отображающая в себе регистры NAND контролера. Через указатель **LX\_NAND** (указывающий на эти регистры в памяти) следует обращаться к регистрам этой структуры. Структура имеет следующие поля:

* IO\_CFG;
* WCT\_CFG;
* NAND\_CFG;
* WR\_CFG;
* RD\_CFG;
* CR;
* SR;
* AR;
* CNTR;
* DR;
* ERR12;
* ERR32;
* SP\_BUF;

Значение регистров можно узнать в спецификации, глава 19.

**Пример 1**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

int main(void){

int nandSt;

nandSt = LX\_NAND->SR.word; // Чтение регистра статуса NAND контролера

return 0;

}

## **NANDSamsung\_TypeDef**

**Резюме**

**typedef** **enum** { *K9F8G08UXM*, *K9F2G08X0B*} NANDSamsung\_TypeDef;

**Описание**

Тип содержащий имена моделей памяти Samsung для которых создан драйвер. В дальнейшем возможно расширение этого типа.

## **Driver\_NANDSamsung\_WaitWrite**

**Резюме**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

void Driver\_NANDSamsung\_WaitWrite()

**Описание**

Функция будет исполняться, пока не будет разрешена запись данных в NAND Flash. Функция необходима в случае, если пользователь будет делать запись в NAND самостоятельно (без использования драйверов).

**Пример 1**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

int main(void){

Driver\_NANDSamsung\_Enable(*K9F2G08X0B*);

LX\_NAND->AR =0;

LX\_NAND->CR.word = 3;

LX\_NAND->CNTR = 1;

Driver\_NANDSamsung\_WaitWrite();

LX\_NAND->DR = 0x12345678;

Driver\_NANDSamsung\_WaitWrite();

LX\_NAND->CR.word = 0;

return 0;

}

## **Driver\_NANDSamsung\_WaitRead**

**Резюме**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

void Driver\_NANDSamsung\_WaitRead()

**Описание**

Функция будет исполняться, пока не будет разрешено чтение данных из NAND Flash. Функция необходима в случае, если пользователь будет производить чтение из NAND самостоятельно (без использования драйверов).

**Пример 1**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

int main(void){

int data;

Driver\_NANDSamsung\_Enable(*K9F2G08X0B*);

LX\_NAND->AR = 0;

LX\_NAND->CR.word = 1;

LX\_NAND->CNTR = 1;

Driver\_NANDSamsung\_WaitRead();

data = LX\_NAND->DR;

LX\_NAND->CR.word = 0;

return 0;

}

## **Driver\_NANDSamsung\_WaitErase**

**Резюме**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

void Driver\_NANDSamsung\_WaitErase()

**Описание**

Функция будет исполняться, пока не будет разрешена очистка NAND Flash. Функция необходима в случае, если пользователь будет делать запись в NAND самостоятельно (без использования драйверов). Обращаю внимание на то, что после Erase необходимо выполнить чтение из NAND контролера. Для подробной информации см. Спецификацию, пункт 19.4.4.

**Пример 1**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

int main(void){

int tmp;

Driver\_NANDSamsung\_Enable(*K9F2G08X0B*);

LX\_NAND->NAND\_CFG.word = 0x00E5225C; //only address row

LX\_NAND->RD\_CFG.word = ERASE\_COMM;

LX\_NAND->SR.word = (1<<13); //reset bit 13 from Status Register

LX\_NAND->AR = 0; //reset 0 block

LX\_NAND->CR.word = 0x301;

LX\_NAND->CNTR = 1;

Driver\_NANDSamsung\_WaitErase(); //wait Erase

Driver\_NANDSamsung\_WaitRead();

tmp = LX\_NAND->DR; //read trash byte

LX\_NAND->CR.word = 0;

return 0;

}

## **Driver\_NANDSamsung\_Enable**

**Резюме**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

void Driver\_NANDSamsung\_Enable(NANDSamsung\_TypeDef numModel)

NANDSamsung\_TypeDef numModel – модель памяти Samsung

**Описание**

Функция настраивает NAND контролер на работу с соответствующей памятью от компании Samsung и включает альтернативные функции для GPIO (порты PxD[16:31] и PxA[21:0]) на работу с NAND.

Памяти, реализованные на данный момент:

* *K9F8G08UXM*
* *K9F2G08X0B*

**Пример 1**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

int main(void){

Driver\_NANDSamsung\_Enable(*K9F2G08X0B*);

return 0;

}

## **Driver\_NANDSamsung\_Disable**

**Резюме**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

void Driver\_NANDSamsung\_Disable()

**Описание**

Функция выключает NAND контролер и альтернативные функции для GPIO (порты PxD[16:31] и PxA[21:0]).

**Пример 1**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

int main(void){

Driver\_NANDSamsung\_Enable(*K9F2G08X0B*);

Driver\_NANDSamsung\_Disable();

return 0;

}

## **Driver\_DMANANDSamsung\_WriteInsidePage**

**Резюме**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

void Driver\_DMANANDSamsung\_WriteInsidePage (int dmaCh, void \*tcb, void \* dst,)

int dmaCh – номер канала DMA для записи (только 4-7 каналы)

void \*tcb – указатель на значения для конфигурации регистров TCB

void \*dst – адрес по которому происходит запись данных в NAND Flash

**Описание**

Функция выполняет запись данных в NAND Flash по выбранному каналу DMA. Ограничение – запись данных должна быть выполнена в пределах страницы. Размер станицы можно получить с помощью функции Driver\_NANDSamsung\_GetPageSize(). *Преимуществом* этой функцией является то, что запись происходит по DMA, поэтому функция исполняется значительно быстрее своих аналогов, т.к. она не ожидает завершения записи, а только программирует и запускает выбранный канал DMA, и пока идет запись по DMA, параллельно можно продолжать исполнять остальную часть программы. Обращаю ВНИМАНИЕ на то, что отслеживать завершение чтение можно при помощи функции HAL\_DMA\_WaitForChannel()(см. HAL\_DMA) не совсем корректно, т.к. канал DMA завершит свою работу раньше, чем произойдет запись последних данных в NAND Flash. Для отслеживания завершения записи данных можно использовать такую конструкцию **while**(LX\_NAND->CNTR). Запись может производится как словами, так и квадрословами, в зависимости от того, какой режим будет выбран для DMA. (TCB\_NORMAL или TCB\_QUAD). Также рекомендуется для определения завершения записи сначала использовать функцию HAL\_DMA\_WaitForChannel(),читая значение, которое она возвращает для отслеживания ошибки при работе канала DMA, а затем использовать конструкцию **while**(LX\_NAND->CNTR)(см. Пример).

**Пример 1**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

#define DATA\_SIZE 0x50

\_\_attribute\_\_((aligned(4)))

int dataArr[DATA\_SIZE];

int main(void){

int dmaSt, errCount = 0;

unsigned int \_\_attribute\_\_((aligned(4))) tcb[4];

Driver\_NANDSamsung\_Enable(*K9F2G08X0B*);

//TCB Config

tcb[0] = (unsigned int) &dataArr;

tcb[1] = DATA\_SIZE<<16 | 4;

tcb[2] = 0;

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

Driver\_DMANANDSamsung\_WriteInsidePage(4,&tcb,(unsigned int \*) 0x210);

dmaSt = HAL\_DMA\_WaitForChannel(4);

if (dmaSt) errCount++;

**while**(LX\_NAND->CNTR);

return 0;

}

## **Driver\_NANDSamsung\_WriteInsidePage**

**Резюме**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

void Driver\_NANDSamsung\_WriteInsidePage(void \*src, void \*dst, int size)

void \*src – указатель на источник данных

void \*dst –адрес записи данных в NAND Flash памяти

int size – количество записываемых в NAND Flash данных

**Описание**

Функция выполняет запись данных в NAND Flash процессором. *Ограничение* состоит в том, что пользователь должен сам следить за тем, чтобы записываемы данные лежали в пределах одной страницы. Размер страницы в Кбайтах можно получить при помощи функции Driver\_NANDSamsung\_GetPageSize(). *Преимуществом* этой функцией является то, что она работает несколько быстрее, чем Driver\_NANDSamsung\_WriteBurst(). Запись производится процессором и в словах (4 байта).

**Пример 1**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

#define DATA\_SIZE 16

int data[DATA\_SIZE];

int main(void){

int i;

for(i=0; i<DATA\_SIZE; i++)

data[i] = i;

Driver\_NANDSamsung\_Enable(*K9F2G08X0B*);

Driver\_NANDSamsung\_WriteInsidePage(&data, (unsigned int \*) 0, DATA\_SIZE);

return 0;

}

## **Driver\_NANDSamsung\_WriteBurst**

**Резюме**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

void Driver\_NANDSamsung\_WriteBurst(void \*src, void \*dst, int size)

void \*src – указатель на источник данных

void \*dst –адрес записи данных в NAND Flash памяти

int size – количество записываемых в NAND Flash данных

**Описание**

Функция выполняет запись данных в NAND Flash. *Преимуществом* этой функции над ее аналогами является то, что она делает запись вне зависимости от того, выходят ли записываемые данные за предел страницы или нет. *Недостатком* этой функции является скорость ее работы. Она медленнее своих аналогов. Запись производится процессором и в словах (4 байта).

**Пример 1**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

#define DATA\_SIZE 32300

int data[DATA\_SIZE];

int main(void){

int i;

for(i=0; i<DATA\_SIZE; i++)

data[i] = i;

Driver\_NANDSamsung\_Enable(*K9F2G08X0B*);

Driver\_NANDSamsung\_WriteBurst( &data, (unsigned int \*) 0, DATA\_SIZE);

return 0;

}

## **Driver\_NANDSamsung\_WriteWord**

**Резюме**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

void Driver\_NANDSamsung\_WriteWord(int data, void \*dst)

int data – записываемое слово

void \*dst – адрес записи данных в NAND Flash памяти

**Описание**

Функция выполняет запись слова (4 байта) в NAND Flash. Функцию выгодно использовать, когда необходимо записать одно слово. При записи большого количества слов лучше использовать Driver\_NANDSamsung\_WriteBurst или Driver\_NANDSamsung\_WriteInsidePage, чем вызывать данную функцию несколько раз подряд.

**Пример 1**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

int main(void){

Driver\_NANDSamsung\_Enable(*K9F2G08X0B*);

Driver\_NANDSamsung\_WriteWord( 0x01234567, (unsigned int \*) 0x103);

return 0;

}

## **Driver\_DMANANDSamsung\_ReadInsidePage**

**Резюме**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

void Driver\_DMANANDSamsung\_ReadInsidePage (int dmaCh, void \*tcb, void \* src,)

int dmaCh – номер канала DMA для чтение (только 8-11 каналы)

void \*tcb – указатель на значения для конфигурации регистров TCB

void \*src – адрес, по которому происходит чтение данных из NAND Flash

**Описание**

Функция выполняет чтение данных из NAND Flash по выбранному каналу DMA. Ограничение – чтение данных должно быть выполнено в пределах страницы. Размер станицы можно получить с помощью функции Driver\_NANDSamsung\_GetPageSize(). *Преимуществом* этой функцией является то, что чтение происходит по DMA, поэтому функция исполняется значительно быстрее своих аналогов, т.к. она не ожидает завершения чтения, а только програмирует и запускает выбранный канал DMA, и пока идет чтение по DMA, параллельно можно продолжать исполнять остальную часть программы. Отслеживать завершение чтения можно при помощи функции HAL\_DMA\_WaitForChannel() (см. HAL\_DMA). Чтение может производится как словами, так и квадрословами, в зависимости от того какой режим будет выбран для DMA. (TCB\_NORMAL или TCB\_QUAD). Так же рекомендуется при использовании функции HAL\_DMA\_WaitForChannel читать значение, которое она возвращает, для отслеживания ошибки при работе канала DMA.

**Пример 1**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

#define DATA\_SIZE 0x50

int dataArr[DATA\_SIZE];

int main(void){

int dmaSt, errCount = 0;

unsigned int \_\_attribute\_\_((aligned(4))) tcb[4];

Driver\_NANDSamsung\_Enable(*K9F2G08X0B*);

//TCB Config

tcb[0] = (unsigned int) &dataArr;

tcb[1] = DATA\_SIZE<<16 | 1;

tcb[2] = 0;

tcb[3] = TCB\_INTMEM | TCB\_NORMAL;

Driver\_DMANANDSamsung\_ReadInsidePage(10,&tcb,(unsigned int\*) 0x210);

dmaSt = HAL\_DMA\_WaitForChannel(10);

if (dmaSt) errCount++;

return 0;

}

## **Driver\_NANDSamsung\_ReadInsidePage**

**Резюме**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

void Driver\_NANDSamsung\_ReadInsidePage (void \*src, void \*dst, int size)

void \*src – адрес, по которому происходит чтение данных из NAND Flash

void \*dst – указатель на место сохранения данных

int size – количество данных для чтения

**Описание**

Функция выполняет чтение данных из NAND Flash. Ограничение – чтение данных должно быть выполнено в пределах одной страницы. Размер станицы можно получить с помощью функции Driver\_NANDSamsung\_GetPageSize(). *Преимуществом* этой функцией является то, что она работает несколько быстрее, чем Driver\_NANDSamsung\_ReadBurst(). Чтение производится процессором и в словах (4 байта).

**Пример 1**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

#define DATA\_SIZE 39

int data[DATA\_SIZE];

int main(void){

Driver\_NANDSamsung\_Enable(*K9F2G08X0B*);

Driver\_NANDSamsung\_ReadInsidePage ((unsigned int \*) 0x34, &data, DATA\_SIZE);

return 0;

}

## **Driver\_NANDSamsung\_ReadBurst**

**Резюме**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

void Driver\_NANDSamsung\_ReadBurst(void \*src, void \*dst, int size)

void \*src – адрес, по которому происходит чтение данных из NAND Flash

void \*dst – указатель на место сохранения данных

int size – количество данных для чтения

**Описание**

Функция выполняет чтение из NAND Flash, при этом считываемые данные могут выходит за пределы одной страницы. *Недостаток* этой функции – она работает медленее, чем Driver\_NANDSamsung\_ReadInsidePage. Чтения производится процессором и в словах (4 байта).

**Пример 1**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

#define DATA\_SIZE 31007

int data[DATA\_SIZE];

int main(void){

Driver\_NANDSamsung\_Enable(*K9F2G08X0B*);

Driver\_NANDSamsung\_ReadBurst((unsigned int \*) 4, &data, DATA\_SIZE);

return 0;

}

## **Driver\_NANDSamsung\_ReadWord**

**Резюме**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

int Driver\_NANDSamsung\_ReadWord(void \*src)

void \*src – адрес читаемого слова

**Описание**

Функция возвращает считанное слово (4 байта) из NAND Flash по переданному ей адресу. Чтение производится процессором. Функцию выгодно использовать, если необходимо прочитать одно слово, для чтения большого количества слов лучше использовать Driver\_NANDSamsung\_ReadBurst или Driver\_NANDSamsung\_ReadInsidePage.

**Пример 1**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

int main(void){

int data;

Driver\_NANDSamsung\_Enable(*K9F2G08X0B*);

data = Driver\_NANDSamsung\_ReadWord((unsigned int \*) 0x307);

return 0;

}

## **Driver\_NANDSamsung\_EraseBlocks**

**Резюме**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

void Driver\_NANDSamsung\_EraseBlocks(int numBlock, int amount)

int numBlock – номер блока, с которого начнется очищение NAND Flash памяти

int amount – количество стираемых блоков

**Описание**

Функция очищает память NAND Flash блоками, начиная с блока numBlock и заканчивая блоком с номером numBlock + (amount-1).

**Пример 1**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

int main(void){

Driver\_NANDSamsung\_Enable(*K9F2G08X0B*);

Driver\_NANDSamsung\_EraseBlocks(0, 1); // Очищение только нулевого блока

return 0;

}

## **Driver\_NANDSamsung\_EraseMem**

**Резюме**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

void Driver\_NANDSamsung\_EraseMem(int memSize)

int memSize – количество очищаемой памяти в Мбайтах.

**Описание**

Функция очищает заданное количество Мбайт, начиная с нулевого адреса. В случае если размер блока превышает 1 Мбайт, в функцию необходимо передавать значение кратное границы блока. Например, если размер блока памяти NAND Flash = 2 Мбайта, в функцию нужно передавать только четные значения. Для памятей *K9F2G08X0B* и *K9F2G08X0B* размер блока меньше чем 1Мбайт. Функцию удобно использовать для очищения ВСЕЙ памяти.

**Пример 1**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

int main(void){

Driver\_NANDSamsung\_Enable(*K9F2G08X0B*);

Driver\_NANDSamsung\_EraseMem(256); //Очищение всей памяти данной NAND Flash

return 0;

}

## **Driver\_NANDSamsung\_GetPageSize**

**Резюме**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

int Driver\_NANDSamsung\_GetPageSize(void)

**Описание**

Функция возвращает размер страницы в Кбайтах, на работу с которым сейчас настроен NAND контролер.

**Пример 1**

#include <hal\_1967VN044.h>

#include <driver1967VN044\_NANDSamsung.h>

int main(void){

int pageSize;

Driver\_NANDSamsung\_Enable(*K9F2G08X0B*);

pageSize = Driver\_NANDSamsung\_GetPageSize();

return 0;

}

## **Пример использования**

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

**#include** <driver1967VN044\_NANDSamsung.h>

**#include** <stdint.h>

**#define** DATA\_SIZE 40

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

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

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

{

**int** i;

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

data\_tx[i] = i+5;

Driver\_NANDSamsung\_Enable(*K9F2G08X0B*);

Driver\_NANDSamsung\_EraseBlocks(0, 1);

Driver\_NANDSamsung\_WriteBurst( data\_tx, 0, DATA\_SIZE);

Driver\_NANDSamsung\_ReadBurst( 0 , data\_rx, DATA\_SIZE);

**while**(1);

**return** 0;

}