Libnds Documentation

Introduction

Welcome to the libnds reference documentation.

2D engine API

video.h File Reference

contains the basic defnitions for controlling the video hardware. More...

```
#include <nds/ndstypes.h>
#include <nds/arm9/sassert.h>
```

Defines

#define ARGB16(a, r, g, b)   (((a) << 15) | (r)|((g)<<5)|((b)<<10))
    Macro to convert 5 bit r g b components plus 1 bit alpha into a single 16 bit ARGB triplet.

#define BG_GFX   ((u16*)0x6000000)
    background graphics memory

#define BG_GFX_SUB   ((u16*)0x6200000)
    background graphics memory (sub engine)

#define BG_PALETTE   ((u16*)0x05000000)
    background palette memory

#define BG_PALETTE_SUB   ((u16*)0x05000400)
    background palette memory (sub engine)

#define OAM   ((u16*)0x07000000)
    pointer to Object Attribute Memory

#define OAM_SUB   ((u16*)0x07000400)
    pointer to Object Attribute Memory (Sub engine)

#define RGB15(r, g, b)   ((r)|((g)<<5)|((b)<<10))
    Macro to convert 5 bit r g b components into a single 15 bit RGB triplet.

#define SCREEN_HEIGHT   192
    Screen height in pixels.

#define SCREEN_WIDTH   256
    Screen width in pixels.

#define SPRITE_GFX   ((u16*)0x6400000)
    sprite graphics memory

#define SPRITE_GFX_SUB   ((u16*)0x6600000)
    sprite graphics memory (sub engine)

#define SPRITE_PALETTE   ((u16*)0x05000200)
    sprite palette memory

#define	SPRITE_PALETTE_SUB ((u16*)0x05000600)
	sprite palette memory (sub engine)
#define	VRAM_A ((u16*)0x6800000)
	pointer to vram bank A mapped as LCD
#define	VRAM_B ((u16*)0x6820000)
	pointer to vram bank B mapped as LCD
#define	VRAM_C ((u16*)0x6840000)
	pointer to vram bank C mapped as LCD
#define	VRAM_D ((u16*)0x6860000)
	pointer to vram bank D mapped as LCD
#define	VRAM_E ((u16*)0x6880000)
	pointer to vram bank E mapped as LCD
#define	VRAM_E_EXT_PALETTE ((_ext_palette *)VRAM_E)
	Used for accessing vram E as an extended palette.
#define	VRAM_F ((u16*)0x6890000)
	pointer to vram bank F mapped as LCD
#define	VRAM_F_EXT_PALETTE ((_ext_palette *)VRAM_F)
	Used for accessing vram F as an extended palette.
#define	VRAM_F_EXT_SPR_PALETTE ((_palette *)VRAM_F)
	Used for accessing vram F as an extended sprite palette.
#define	VRAM_G ((u16*)0x6894000)
	pointer to vram bank G mapped as LCD
#define	VRAM_G_EXT_PALETTE ((_ext_palette *)VRAM_G)
	Used for accessing vram G as an extended palette.
#define	VRAM_G_EXT_SPR_PALETTE ((_palette *)VRAM_G)
	Used for accessing vram G as an extended sprite palette.
#define	VRAM_H ((u16*)0x6898000)
	pointer to vram bank H mapped as LCD
#define	VRAM_H_EXT_PALETTE ((_ext_palette *)VRAM_H)
	Used for accessing vram H as an extended palette.
#define	VRAM_I ((u16*)0x68A0000)
	pointer to vram bank I mapped as LCD
#define	VRAM_I_EXT_SPR_PALETTE ((_palette *)VRAM_I)
	Used for accessing vram I as an extended sprite palette.

Typedefs

typedef _palette	_ext_palette [16]
	An array of 16 256-color palettes.
typedef u16	_palette [256]
	an array of 256 15-bit RGB values

Enumerations

```
enum  VideoMode {
        MODE_0_2D = 0x10000,
        MODE_1_2D = 0x10001,
        MODE_2_2D = 0x10002,
        MODE_3_2D = 0x10003,
        MODE_4_2D = 0x10004,
        MODE_5_2D = 0x10005,
        MODE_6_2D = 0x10006,
        MODE_0_3D = (0x10000 | (1 << 8) | (1<<3) ),
        MODE_1_3D = (0x10001 | (1 << 8) | (1<<3) ),
        MODE_2_3D = (0x10002 | (1 << 8) | (1<<3) ),
        MODE_3_3D = (0x10003 | (1 << 8) | (1<<3) ),
        MODE_4_3D = (0x10004 | (1 << 8) | (1<<3) ),
        MODE_5_3D = (0x10005 | (1 << 8) | (1<<3) ),
        MODE_6_3D = (0x10006 | (1 << 8) | (1<<3) ),
        MODE_FIFO = (3<<16),
        MODE_FB0 = (0x00020000),
        MODE_FB1 = (0x00060000),
        MODE_FB2 = (0x000A0000),
        MODE_FB3 = (0x000E0000)
      }
```
The allowed video modes of the 2D processors

More...

```
enum  VRAM_A_TYPE {
        VRAM_A_LCD = 0,
        VRAM_A_MAIN_BG = 1,
        VRAM_A_MAIN_BG_0x06000000 = 1 | (( 0 )<<3),
        VRAM_A_MAIN_BG_0x06020000 = 1 | (( 1 )<<3),
        VRAM_A_MAIN_BG_0x06040000 = 1 | (( 2 )<<3),
        VRAM_A_MAIN_BG_0x06060000 = 1 | (( 3 )<<3),
        VRAM_A_MAIN_SPRITE = 2,
        VRAM_A_MAIN_SPRITE_0x06400000 = 2 | (( 0 )<<3),
        VRAM_A_MAIN_SPRITE_0x06420000 = 2 | (( 1 )<<3),
        VRAM_A_TEXTURE = 3,
        VRAM_A_TEXTURE_SLOT0 = 3 | (( 0 )<<3),
        VRAM_A_TEXTURE_SLOT1 = 3 | (( 1 )<<3),
        VRAM_A_TEXTURE_SLOT2 = 3 | (( 2 )<<3),
        VRAM_A_TEXTURE_SLOT3 = 3 | (( 3 )<<3)
      }
```
Allowed VRAM bank A modes.

More...

```
enum  VRAM_B_TYPE {
        VRAM_B_LCD = 0,
        VRAM_B_MAIN_BG = 1 | (( 1 )<<3),
        VRAM_B_MAIN_BG_0x06000000 = 1 | (( 0 )<<3),
        VRAM_B_MAIN_BG_0x06020000 = 1 | (( 1 )<<3),
        VRAM_B_MAIN_BG_0x06040000 = 1 | (( 2 )<<3),
        VRAM_B_MAIN_BG_0x06060000 = 1 | (( 3 )<<3),
        VRAM_B_MAIN_SPRITE = 2,
```

VRAM_B_MAIN_SPRITE_0x06400000 = 2 | ((0)<<3),
VRAM_B_MAIN_SPRITE_0x06420000 = 2 | ((1)<<3),
VRAM_B_TEXTURE = 3 | ((1)<<3),
VRAM_B_TEXTURE_SLOT0 = 3 | ((0)<<3),
VRAM_B_TEXTURE_SLOT1 = 3 | ((1)<<3),
VRAM_B_TEXTURE_SLOT2 = 3 | ((2)<<3),
VRAM_B_TEXTURE_SLOT3 = 3 | ((3)<<3)
}

Allowed VRAM bank B modes.

More...

enum VRAM_C_TYPE {
VRAM_C_LCD = 0,
VRAM_C_MAIN_BG = 1 | ((2)<<3),
VRAM_C_MAIN_BG_0x06000000 = 1 | ((0)<<3),
VRAM_C_MAIN_BG_0x06020000 = 1 | ((1)<<3),
VRAM_C_MAIN_BG_0x06040000 = 1 | ((2)<<3),
VRAM_C_MAIN_BG_0x06060000 = 1 | ((3)<<3),
VRAM_C_ARM7 = 2,
VRAM_C_ARM7_0x06000000 = 2 | ((0)<<3),
VRAM_C_ARM7_0x06020000 = 2 | ((1)<<3),
VRAM_C_SUB_BG = 4,
VRAM_C_SUB_BG_0x06200000 = 4 | ((0)<<3),
VRAM_C_TEXTURE = 3 | ((2)<<3),
VRAM_C_TEXTURE_SLOT0 = 3 | ((0)<<3),
VRAM_C_TEXTURE_SLOT1 = 3 | ((1)<<3),
VRAM_C_TEXTURE_SLOT2 = 3 | ((2)<<3),
VRAM_C_TEXTURE_SLOT3 = 3 | ((3)<<3)
}

Allowed VRAM bank C modes.

More...

enum VRAM_D_TYPE {
VRAM_D_LCD = 0,
VRAM_D_MAIN_BG = 1 | ((3)<<3),
VRAM_D_MAIN_BG_0x06000000 = 1 | ((0)<<3),
VRAM_D_MAIN_BG_0x06020000 = 1 | ((1)<<3),
VRAM_D_MAIN_BG_0x06040000 = 1 | ((2)<<3),
VRAM_D_MAIN_BG_0x06060000 = 1 | ((3)<<3),
VRAM_D_ARM7 = 2 | ((1)<<3),
VRAM_D_ARM7_0x06000000 = 2 | ((0)<<3),
VRAM_D_ARM7_0x06020000 = 2 | ((1)<<3),
VRAM_D_SUB_SPRITE = 4,
VRAM_D_TEXTURE = 3 | ((3)<<3),
VRAM_D_TEXTURE_SLOT0 = 3 | ((0)<<3),
VRAM_D_TEXTURE_SLOT1 = 3 | ((1)<<3),
VRAM_D_TEXTURE_SLOT2 = 3 | ((2)<<3),
VRAM_D_TEXTURE_SLOT3 = 3 | ((3)<<3)
}

Allowed VRAM bank D modes.

More...

enum VRAM_E_TYPE {
  VRAM_E_LCD = 0,
  VRAM_E_MAIN_BG = 1,
  VRAM_E_MAIN_SPRITE = 2,
  VRAM_E_TEX_PALETTE = 3,
  VRAM_E_BG_EXT_PALETTE = 4
}

Allowed VRAM bank E modes.

More...

enum VRAM_F_TYPE {
  VRAM_F_LCD = 0,
  VRAM_F_MAIN_BG = 1,
  VRAM_F_MAIN_BG_0x06000000 = 1 | ((0)<<3),
  VRAM_F_MAIN_BG_0x06004000 = 1 | ((1)<<3),
  VRAM_F_MAIN_BG_0x06010000 = 1 | ((2)<<3),
  VRAM_F_MAIN_BG_0x06014000 = 1 | ((3)<<3),
  VRAM_F_MAIN_SPRITE = 2,
  VRAM_F_MAIN_SPRITE_0x06400000 = 2 | ((0)<<3),
  VRAM_F_MAIN_SPRITE_0x06404000 = 2 | ((1)<<3),
  VRAM_F_MAIN_SPRITE_0x06410000 = 2 | ((2)<<3),
  VRAM_F_MAIN_SPRITE_0x06414000 = 2 | ((3)<<3),
  VRAM_F_TEX_PALETTE = 3,
  VRAM_F_TEX_PALETTE_SLOT0 = 3 | ((0)<<3),
  VRAM_F_TEX_PALETTE_SLOT1 = 3 | ((1)<<3),
  VRAM_F_TEX_PALETTE_SLOT4 = 3 | ((2)<<3),
  VRAM_F_TEX_PALETTE_SLOT5 = 3 | ((3)<<3),
  VRAM_F_BG_EXT_PALETTE = 4,
  VRAM_F_BG_EXT_PALETTE_SLOT01 = 4 | ((0)<<3),
  VRAM_F_BG_EXT_PALETTE_SLOT23 = 4 | ((1)<<3),
  VRAM_F_SPRITE_EXT_PALETTE = 5
}

Allowed VRAM bank F modes.

More...

enum VRAM_G_TYPE {
  VRAM_G_LCD = 0,
  VRAM_G_MAIN_BG = 1,
  VRAM_G_MAIN_BG_0x06000000 = 1 | ((0)<<3),
  VRAM_G_MAIN_BG_0x06004000 = 1 | ((1)<<3),
  VRAM_G_MAIN_BG_0x06010000 = 1 | ((2)<<3),
  VRAM_G_MAIN_BG_0x06014000 = 1 | ((3)<<3),
  VRAM_G_MAIN_SPRITE = 2,
  VRAM_G_MAIN_SPRITE_0x06400000 = 2 | ((0)<<3),
  VRAM_G_MAIN_SPRITE_0x06404000 = 2 | ((1)<<3),
  VRAM_G_MAIN_SPRITE_0x06410000 = 2 | ((2)<<3),
  VRAM_G_MAIN_SPRITE_0x06414000 = 2 | ((3)<<3),
  VRAM_G_TEX_PALETTE = 3,
  VRAM_G_TEX_PALETTE_SLOT0 = 3 | ((0)<<3),
  VRAM_G_TEX_PALETTE_SLOT1 = 3 | ((1)<<3),

VRAM_G_TEX_PALETTE_SLOT4 = 3 | ((2)<<3),
VRAM_G_TEX_PALETTE_SLOT5 = 3 | ((3)<<3),
VRAM_G_BG_EXT_PALETTE = 4,
VRAM_G_BG_EXT_PALETTE_SLOT01 = 4 | ((0)<<3),
VRAM_G_BG_EXT_PALETTE_SLOT23 = 4 | ((1)<<3),
VRAM_G_SPRITE_EXT_PALETTE = 5
}
Allowed VRAM bank G modes.

More...

enum VRAM_H_TYPE {
VRAM_H_LCD = 0,
VRAM_H_SUB_BG = 1,
VRAM_H_SUB_BG_EXT_PALETTE = 2
}
Allowed VRAM bank H modes.

More...

enum VRAM_I_TYPE {
VRAM_I_LCD = 0,
VRAM_I_SUB_BG_0x06208000 = 1,
VRAM_I_SUB_SPRITE = 2,
VRAM_I_SUB_SPRITE_EXT_PALETTE = 3
}
Allowed VRAM bank I modes.

More...

Functions

static void	setBackdropColor (const u16 color)
	sets the backdrop color of the main engine.
static void	setBackdropColorSub (const u16 color)
	sets the backdrop color of the sub engine.
void	setBrightness (int screen, int level)
	sets the screens brightness.
static bool	video3DEnabled ()
	determine if 3D is enabled
static void	videoBgDisable (int number)
	disables the specified background on the main engine
static void	videoBgDisableSub (int number)
	disables the specified background on the sub engine
static void	videoBgEnable (int number)
	enables the specified background on the main engine
static void	videoBgEnableSub (int number)
	enables the specified background on the sub engine
static int	videoGetMode ()

	return the main 2D engine video mode
static int	videoGetModeSub ()
	return the main 2D engine video mode
static void	videoSetMode (u32 mode)
	the main 2D engine video mode
static void	videoSetModeSub (u32 mode)
	the sub 2D engine video mode
u32	vramDefault ()
	Set VRAM banks to basic default.
void	vramRestoreBanks_EFG (u32 vramTemp)
	Restore the E,F,G bank modes.
void	vramRestorePrimaryBanks (u32 vramTemp)
	Restore the main 4 bank modes.
static void	vramSetBankA (VRAM_A_TYPE a)
	Set bank A to the indicated mapping.
static void	vramSetBankB (VRAM_B_TYPE b)
	Set bank B to the indicated mapping.
static void	vramSetBankC (VRAM_C_TYPE c)
	Set bank C to the indicated mapping.
static void	vramSetBankD (VRAM_D_TYPE d)
	Set bank D to the indicated mapping.
static void	vramSetBankE (VRAM_E_TYPE e)
	Set bank E to the indicated mapping.
static void	vramSetBankF (VRAM_F_TYPE f)
	Set bank F to the indicated mapping.
static void	vramSetBankG (VRAM_G_TYPE g)
	Set bank G to the indicated mapping.
static void	vramSetBankH (VRAM_H_TYPE h)
	Set bank H to the indicated mapping.
static void	vramSetBankI (VRAM_I_TYPE i)
	Set bank I to the indicated mapping.
u32	vramSetBanks_EFG (VRAM_E_TYPE e, VRAM_F_TYPE f, VRAM_G_TYPE g)
	Set E,F,G bank modes.
u32	vramSetPrimaryBanks (VRAM_A_TYPE a, VRAM_B_TYPE b, VRAM_C_TYPE c, VRAM_D_TYPE d)
	Set the main 4 bank modes.

Detailed Description

contains the basic defnitions for controlling the video hardware.

Intro

Video.h contains the basic defnitions for controlling the video hardware.

Video Ram Banks

The Nintendo DS has nine banks of video memory which may be put to a variety of uses. They can hold the graphics for your sprites, the textures for your 3D space ships, the tiles for your 2D platformer, or a direct map of pixels to render to the screen. Figuring out how to effectively utilize this flexible but limited amount of memory will be one the most challenging endeavors you will face early homebrew development.

The nine banks can be utilized as enumerated by the VRAM types. Banks are labled A-I. In order to utilize 2D or 3D texture graphics, memory must be mapped for these purposes.

For instance: If you initialize a 2D background on the main engine you will be expected to define both an offset for its map data and an offset for its tile graphics (bitmapped backgrounds differ slightly). These offsets are referenced from the start of 2D background graphics memory. On the main display 2D background graphics begin at 0x6000000.

Without mapping a VRAM bank to this location data written to your background tile and map offsets will be lost.

VRAM banks can be mapped to specific addresses for specific purposes. In our case, any of the 4 main banks amd several of the smaller ones can be mapped to the main 2D background engine.(A B C and D banks are refered to as ìmainî because they are 128KB and flexible in usage)

```
        vramSetBankA(VRAM_A_MAIN_BG);
```

The above would map the 128KB of VRAM_A to 0x6000000 for use as main background graphics and maps (you can offset the mapping as well and the available offsets are defined in the VRAM_A_TYPE enumberation)

Video Ram Bank sizes

- VRAM A: 128kb
- VRAM B: 128kb
- VRAM C: 128kb
- VRAM D: 128kb
- VRAM E: 64kb
- VRAM F: 16kb
- VRAM G: 16kb
- VRAM H: 32kb
- VRAM I: 16kb

Enumeration Type Documentation

enum VideoMode

The allowed video modes of the 2D processors

Main 2D engine

Mode	BG0	BG1	BG2	BG3
0	T	T	T	T
1	T	T	T	R
2	T	T	R	R
3	T	T	T	E
4	T	T	R	E
5	T	T	E	E
6		L		

```
T = Text
R = Rotation
E = Extended Rotation
L = Large Bitmap background
```

Sub 2D engine

Mode	BG0	BG1	BG2	BG3
0	T	T	T	T
1	T	T	T	R
2	T	T	R	R
3	T	T	T	E
4	T	T	R	E
5	T	T	E	E

Enumerator:

MODE_0_2D	4 2D backgrounds
MODE_1_2D	4 2D backgrounds
MODE_2_2D	4 2D backgrounds
MODE_3_2D	4 2D backgrounds
MODE_4_2D	4 2D backgrounds
MODE_5_2D	4 2D backgrounds
MODE_6_2D	4 2D backgrounds
MODE_0_3D	3 2D backgrounds 1 3D background (Main engine only)
MODE_1_3D	3 2D backgrounds 1 3D background (Main engine only)
MODE_2_3D	3 2D backgrounds 1 3D background (Main engine only)
MODE_3_3D	3 2D backgrounds 1 3D background (Main engine only)
MODE_4_3D	3 2D backgrounds 1 3D background (Main engine only)
MODE_5_3D	3 2D backgrounds 1 3D background (Main engine only)
MODE_6_3D	3 2D backgrounds 1 3D background (Main engine only)

MODE_FIFO	video display from main memory
MODE_FB0	video display directly from VRAM_A in LCD mode
MODE_FB1	video display directly from VRAM_B in LCD mode
MODE_FB2	video display directly from VRAM_C in LCD mode
MODE_FB3	video display directly from VRAM_D in LCD mode

enum VRAM_A_TYPE
Allowed VRAM bank A modes.

Enumerator:

VRAM_A_LCD	maps vram a to lcd.
VRAM_A_MAIN_BG	maps vram a to main engine background slot 0.
VRAM_A_MAIN_BG_0x06000000	maps vram a to main engine background slot 0.
VRAM_A_MAIN_BG_0x06020000	maps vram a to main engine background slot 1.
VRAM_A_MAIN_BG_0x06040000	maps vram a to main engine background slot 2.
VRAM_A_MAIN_BG_0x06060000	maps vram a to main engine background slot 3.
VRAM_A_MAIN_SPRITE	maps vram a to main engine sprites slot 0.
VRAM_A_MAIN_SPRITE_0x06400000	maps vram a to main engine sprites slot 0.
VRAM_A_MAIN_SPRITE_0x06420000	maps vram a to main engine sprites slot 1.
VRAM_A_TEXTURE	maps vram a to 3d texture slot 0.
VRAM_A_TEXTURE_SLOT0	maps vram a to 3d texture slot 0.
VRAM_A_TEXTURE_SLOT1	maps vram a to 3d texture slot 1.
VRAM_A_TEXTURE_SLOT2	maps vram a to 3d texture slot 2.
VRAM_A_TEXTURE_SLOT3	maps vram a to 3d texture slot 3.

enum VRAM_B_TYPE
Allowed VRAM bank B modes.

Enumerator:

VRAM_B_LCD	maps vram b to lcd.
VRAM_B_MAIN_BG	maps vram b to main engine background slot 1.
VRAM_B_MAIN_BG_0x06000000	maps vram b to main engine background slot 0.
VRAM_B_MAIN_BG_0x06020000	maps vram b to main engine background slot 1.
VRAM_B_MAIN_BG_0x06040000	maps vram b to main engine background slot 2.
VRAM_B_MAIN_BG_0x06060000	maps vram b to main engine background slot 3.
VRAM_B_MAIN_SPRITE	maps vram b to main engine sprites slot 0.
VRAM_B_MAIN_SPRITE_0x06400000	maps vram b to main engine sprites slot 0.
VRAM_B_MAIN_SPRITE_0x06420000	maps vram b to main engine sprites slot 1.
VRAM_B_TEXTURE	maps vram b to 3d texture slot 1.
VRAM_B_TEXTURE_SLOT0	maps vram b to 3d texture slot 0.
VRAM_B_TEXTURE_SLOT1	maps vram b to 3d texture slot 1.
VRAM_B_TEXTURE_SLOT2	maps vram b to 3d texture slot 2.
VRAM_B_TEXTURE_SLOT3	maps vram b to 3d texture slot 3.

enum VRAM_C_TYPE
Allowed VRAM bank C modes.

Enumerator:

VRAM_C_LCD	maps vram c to lcd.
VRAM_C_MAIN_BG	maps vram c to main engine background slot 2.
VRAM_C_MAIN_BG_0x06000000	maps vram c to main engine background slot 0.
VRAM_C_MAIN_BG_0x06020000	maps vram c to main engine background slot 1.
VRAM_C_MAIN_BG_0x06040000	maps vram c to main engine background slot 2.
VRAM_C_MAIN_BG_0x06060000	maps vram c to main engine background slot 3.
VRAM_C_ARM7	maps vram c to ARM7 workram slot 0.

VRAM_C_ARM7_0x06000000	maps vram c to ARM7 workram slot 0.
VRAM_C_ARM7_0x06020000	maps vram c to ARM7 workram slot 1.
VRAM_C_SUB_BG	maps vram c to sub engine background slot 0.
VRAM_C_SUB_BG_0x06200000	maps vram c to sub engine background slot 0.
VRAM_C_TEXTURE	maps vram c to 3d texture slot 2.
VRAM_C_TEXTURE_SLOT0	maps vram c to 3d texture slot 0.
VRAM_C_TEXTURE_SLOT1	maps vram c to 3d texture slot 1.
VRAM_C_TEXTURE_SLOT2	maps vram c to 3d texture slot 2.
VRAM_C_TEXTURE_SLOT3	maps vram c to 3d texture slot 3.

enum VRAM_D_TYPE
Allowed VRAM bank D modes.

Enumerator:

VRAM_D_LCD	maps vram d to lcd.
VRAM_D_MAIN_BG	maps vram d to main engine background slot 3.
VRAM_D_MAIN_BG_0x06000000	maps vram d to main engine background slot 0.
VRAM_D_MAIN_BG_0x06020000	maps vram d to main engine background slot 1.
VRAM_D_MAIN_BG_0x06040000	maps vram d to main engine background slot 2.
VRAM_D_MAIN_BG_0x06060000	maps vram d to main engine background slot 3.
VRAM_D_ARM7	maps vram d to ARM7 workram slot 1.
VRAM_D_ARM7_0x06000000	maps vram d to ARM7 workram slot 0.
VRAM_D_ARM7_0x06020000	maps vram d to ARM7 workram slot 1.
VRAM_D_SUB_SPRITE	maps vram d to sub engine sprites slot 0.
VRAM_D_TEXTURE	maps vram d to 3d texture slot 3.
VRAM_D_TEXTURE_SLOT0	maps vram d to 3d texture slot 0.
VRAM_D_TEXTURE_SLOT1	maps vram d to 3d texture slot 1.
VRAM_D_TEXTURE_SLOT2	maps vram d to 3d texture slot 2.
VRAM_D_TEXTURE_SLOT3	maps vram d to 3d texture slot 3.

enum VRAM_E_TYPE
Allowed VRAM bank E modes.

Enumerator:

VRAM_E_LCD	maps vram e to lcd.
VRAM_E_MAIN_BG	maps vram e to main engine background first half of slot 0.
VRAM_E_MAIN_SPRITE	maps vram e to main engine sprites first half of slot 0.
VRAM_E_TEX_PALETTE	maps vram e to 3d texture palette slot 0-3.
VRAM_E_BG_EXT_PALETTE	maps vram e to main engine background extended palette.

enum VRAM_F_TYPE
Allowed VRAM bank F modes.

Enumerator:

VRAM_F_LCD	maps vram f to lcd.
VRAM_F_MAIN_BG	maps vram f to main engine background first part of slot 0.
VRAM_F_MAIN_BG_0x06000000	maps vram f to main engine background first part of slot 0.
VRAM_F_MAIN_BG_0x06004000	maps vram f to main engine background second part of slot 0.
VRAM_F_MAIN_BG_0x06010000	maps vram f to main engine background second half of slot 0.
VRAM_F_MAIN_BG_0x06014000	maps vram f to main engine background second part of second half of slot 0.
VRAM_F_MAIN_SPRITE	maps vram f to main engine sprites first part of slot 0.
VRAM_F_MAIN_SPRITE_0x06400000	maps vram f to main engine sprites first part of slot 0.
VRAM_F_MAIN_SPRITE_0x06404000	maps vram f to main engine sprites second part of slot 0.
VRAM_F_MAIN_SPRITE_0x06410000	maps vram f to main engine sprites second half of slot 0.
VRAM_F_MAIN_SPRITE_0x06414000	maps vram f to main engine sprites second part of second half of slot 0.
VRAM_F_TEX_PALETTE	maps vram f to 3d texture palette slot 0.

VRAM_F_TEX_PALETTE_SLOT0	maps vram f to 3d texture palette slot 0.
VRAM_F_TEX_PALETTE_SLOT1	maps vram f to 3d texture palette slot 1.
VRAM_F_TEX_PALETTE_SLOT4	maps vram f to 3d texture palette slot 4.
VRAM_F_TEX_PALETTE_SLOT5	maps vram f to 3d texture palette slot 5.
VRAM_F_BG_EXT_PALETTE	maps vram f to main engine background extended palette slot 0 and 1.
VRAM_F_BG_EXT_PALETTE_SLOT01	maps vram f to main engine background extended palette slot 0 and 1.
VRAM_F_BG_EXT_PALETTE_SLOT23	maps vram f to main engine background extended palette slot 2 and 3.
VRAM_F_SPRITE_EXT_PALETTE	maps vram f to main engine sprites extended palette.

enum VRAM_G_TYPE

Allowed VRAM bank G modes.

Enumerator:

VRAM_G_LCD	maps vram g to lcd.
VRAM_G_MAIN_BG	maps vram g to main engine background first part of slot 0.
VRAM_G_MAIN_BG_0x06000000	maps vram g to main engine background first part of slot 0.
VRAM_G_MAIN_BG_0x06004000	maps vram g to main engine background second part of slot 0.
VRAM_G_MAIN_BG_0x06010000	maps vram g to main engine background second half of slot 0.
VRAM_G_MAIN_BG_0x06014000	maps vram g to main engine background second part of second half of slot 0.
VRAM_G_MAIN_SPRITE	maps vram g to main engine sprites first part of slot 0.
VRAM_G_MAIN_SPRITE_0x06400000	maps vram g to main engine sprites first part of slot 0.
VRAM_G_MAIN_SPRITE_0x06404000	maps vram g to main engine sprites second part of slot 0.
VRAM_G_MAIN_SPRITE_0x06410000	maps vram g to main engine sprites second half of slot 0.
VRAM_G_MAIN_SPRITE_0x06414000	maps vram g to main engine sprites second part of

	second half of slot 0.
VRAM_G_TEX_PALETTE	maps vram g to 3d texture palette slot 0.
VRAM_G_TEX_PALETTE_SLOT0	maps vram g to 3d texture palette slot 0.
VRAM_G_TEX_PALETTE_SLOT1	maps vram g to 3d texture palette slot 1.
VRAM_G_TEX_PALETTE_SLOT4	maps vram g to 3d texture palette slot 4.
VRAM_G_TEX_PALETTE_SLOT5	maps vram g to 3d texture palette slot 5.
VRAM_G_BG_EXT_PALETTE	maps vram g to main engine background extended palette slot 0 and 1.
VRAM_G_BG_EXT_PALETTE_SLOT01	maps vram g to main engine background extended palette slot 0 and 1.
VRAM_G_BG_EXT_PALETTE_SLOT23	maps vram g to main engine background extended palette slot 2 and 3.
VRAM_G_SPRITE_EXT_PALETTE	maps vram g to main engine sprites extended palette.

enum VRAM_H_TYPE
Allowed VRAM bank H modes.

Enumerator:

VRAM_H_LCD	maps vram h to lcd.
VRAM_H_SUB_BG	maps vram h to sub engine background first 2 parts of slot 0.
VRAM_H_SUB_BG_EXT_PALETTE	maps vram h to sub engine background extended palette.

enum VRAM_I_TYPE
Allowed VRAM bank I modes.

Enumerator:

VRAM_I_LCD	maps vram i to lcd.
VRAM_I_SUB_BG_0x06208000	maps vram i to sub engine background thirth part of slot 0.
VRAM_I_SUB_SPRITE	maps vram i to sub engine sprites.
VRAM_I_SUB_SPRITE_EXT_PALETTE	maps vram i to sub engine sprites extended palette.

Function Documentation

static void setBackdropColor (const u16 *color*) `[inline, static]`
sets the backdrop color of the main engine.

the backdrop color is displayed when all pixels at a given location are transparent (no sprite or background is visible there).

Parameters:
 color the color that the backdrop of the main engine should display.
<background palette memory

static void setBackdropColorSub (const u16 *color*) `[inline, static]`
sets the backdrop color of the sub engine.

the backdrop color is displayed when all pixels at a given location are transparent (no sprite or background is visible there).

Parameters:
 color the color that the backdrop of the sub engine should display.
<background palette memory (sub engine)

void setBrightness (int *screen*,
 int *level*
)
sets the screens brightness.

Parameters:

screen	1 = main screen, 2 = subscreen, 3 = both
level	-16 = black, 0 = full brightness, 16 = white

static bool video3DEnabled ( ) `[inline, static]`
determine if 3D is enabled

Returns:
 true if 3D is enabled

static void videoBgDisable (int *number*) `[inline, static]`
disables the specified background on the main engine

Parameters:
 number the background number (0-3)

static void videoBgDisableSub (int *number*) `[inline, static]`
disables the specified background on the sub engine

Parameters:
 number the background number (0-3)

static void videoBgEnable (int *number*) `[inline, static]`
enables the specified background on the main engine

Parameters:
 number the background number (0-3)

static void videoBgEnableSub (int *number*) `[inline, static]`
enables the specified background on the sub engine

Parameters:

number the background number (0-3)

static int videoGetMode ( ) `[inline, static]`

return the main 2D engine video mode

Returns:

the video mode

static int videoGetModeSub ( ) `[inline, static]`

return the main 2D engine video mode

Returns:

the video mode

static void videoSetMode (u32 *mode*) `[inline, static]`

the main 2D engine video mode

Parameters:

mode the video mode to set

Examples:

audio/maxmod/song_events_example/source/template.c, audio/maxmod/song_events_example2/source/template.c, capture/ScreenShot/source/main.cpp, Graphics/3D/3D_Both_Screens/source/template.c, Graphics/3D/BoxTest/source/main.cpp, Graphics/3D/Display_List/source/main.cpp, Graphics/3D/Display_List_2/source/main.cpp, Graphics/3D/Env_Mapping/source/main.cpp, Graphics/3D/nehe/lesson01/source/nehe1.cpp, Graphics/3D/nehe/lesson02/source/nehe2.cpp, Graphics/3D/nehe/lesson03/source/nehe3.cpp, Graphics/3D/nehe/lesson04/source/nehe4.cpp, Graphics/3D/nehe/lesson05/source/nehe5.cpp, Graphics/3D/nehe/lesson06/source/nehe6.cpp, Graphics/3D/nehe/lesson07/source/nehe7.cpp, Graphics/3D/nehe/lesson08/source/nehe8.cpp, Graphics/3D/nehe/lesson09/source/nehe9.cpp, Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/3D/nehe/lesson10b/source/nehe10b.cpp, Graphics/3D/nehe/lesson11/source/nehe11.cpp, Graphics/3D/Ortho/source/main.cpp, Graphics/3D/Paletted_Cube/source/main.cpp, Graphics/3D/Picking/source/main.cpp, Graphics/3D/Simple_Quad/source/main.cpp, Graphics/3D/Simple_Tri/source/main.cpp, Graphics/3D/Textured_Cube/source/main.cpp, Graphics/3D/Textured_Quad/source/main.cpp, Graphics/3D/Toon_Shading/source/main.cpp, Graphics/Backgrounds/16bit_color_bmp/source/template.cpp, Graphics/Backgrounds/256_color_bmp/source/main.cpp, Graphics/Backgrounds/all_in_one/source/advanced.cpp, Graphics/Backgrounds/all_in_one/source/basic.cpp, Graphics/Backgrounds/all_in_one/source/handmade.cpp, Graphics/Backgrounds/all_in_one/source/scrolling.cpp, Graphics/Backgrounds/Double_Buffer/source/main.cpp, Graphics/Backgrounds/rotation/source/main.cpp,

Graphics/Printing/print_both_screens/source/template.c,
Graphics/Printing/rotscale_text/source/main.c,
Graphics/Sprites/allocation_test/source/main.c,
Graphics/Sprites/animate_simple/source/template.c,
Graphics/Sprites/fire_and_sprites/source/main.cpp,
Graphics/Sprites/simple/source/template.c,
Graphics/Sprites/sprite_extended_palettes/source/template.c,
Graphics/Sprites/sprite_rotate/source/template.c,
input/Touch_Pad/touch_look/source/main.cpp,
input/Touch_Pad/touch_test/source/main.c, and time/RealTimeClock/source/main.c.

static void videoSetModeSub (u32 *mode*) `[inline, static]`

the sub 2D engine video mode

Parameters:

mode the video mode to set

Examples:

audio/maxmod/song_events_example2/source/template.c,
Graphics/3D/3D_Both_Screens/source/template.c,
Graphics/3D/Paletted_Cube/source/main.cpp,
Graphics/Backgrounds/16bit_color_bmp/source/template.cpp,
Graphics/Backgrounds/all_in_one/source/main.cpp,
Graphics/Printing/custom_font/source/main.c,
Graphics/Printing/print_both_screens/source/template.c,
Graphics/Printing/rotscale_text/source/main.c,
Graphics/Sprites/allocation_test/source/main.c,
Graphics/Sprites/animate_simple/source/template.c,
Graphics/Sprites/bitmap_sprites/source/main.cpp, Graphics/Sprites/fire_and_sprites/source/main.cpp, and Graphics/Sprites/simple/source/template.c.

u32 vramDefault ( )

Set VRAM banks to basic default.

Returns:

the previous settings

void vramRestoreBanks_EFG (u32 *vramTemp*)

Restore the E,F,G bank modes.

Parameters:

vramTemp	restores the E,F,G bank modes to the value encoded in vramTemp (returned from vramSetBanks_EFG)

void vramRestorePrimaryBanks (u32 *vramTemp*)

Restore the main 4 bank modes.

Parameters:

vramTemp	restores the main 4 banks to the value encoded in vramTemp (returned from vramSetMainBanks)

static void vramSetBankA (VRAM_A_TYPE *a*) `[inline, static]`

Set bank A to the indicated mapping.

Parameters:

a the mapping of the bank

Examples:

audio/maxmod/song_events_example/source/template.c, audio/maxmod/song_events_example2/source/template.c, Graphics/3D/Env_Mapping/source/main.cpp, Graphics/3D/nehe/lesson06/source/nehe6.cpp, Graphics/3D/nehe/lesson07/source/nehe7.cpp, Graphics/3D/nehe/lesson08/source/nehe8.cpp, Graphics/3D/nehe/lesson09/source/nehe9.cpp, Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/3D/nehe/lesson10b/source/nehe10b.cpp, Graphics/3D/nehe/lesson11/source/nehe11.cpp, Graphics/3D/Ortho/source/main.cpp, Graphics/3D/Textured_Cube/source/main.cpp, Graphics/3D/Textured_Quad/source/main.cpp, Graphics/Backgrounds/16bit_color_bmp/source/template.cpp, Graphics/Backgrounds/256_color_bmp/source/main.cpp, Graphics/Backgrounds/all_in_one/source/advanced.cpp, Graphics/Backgrounds/all_in_one/source/basic.cpp, Graphics/Backgrounds/all_in_one/source/handmade.cpp, Graphics/Backgrounds/all_in_one/source/scrolling.cpp, Graphics/Backgrounds/rotation/source/main.cpp, Graphics/Printing/print_both_screens/source/template.c, Graphics/Sprites/allocation_test/source/main.c, Graphics/Sprites/animate_simple/source/template.c, Graphics/Sprites/simple/source/template.c, Graphics/Sprites/sprite_extended_palettes/source/template.c, Graphics/Sprites/sprite_rotate/source/template.c, and input/Touch_Pad/touch_look/source/main.cpp.

static void vramSetBankB (VRAM_B_TYPE *b*) `[inline, static]`

Set bank B to the indicated mapping.

Parameters:

b the mapping of the bank

Examples:

Graphics/3D/Paletted_Cube/source/main.cpp, Graphics/3D/Textured_Cube/source/main.cpp, Graphics/Backgrounds/all_in_one/source/basic.cpp, and Graphics/Sprites/allocation_test/source/main.c.

static void vramSetBankC (VRAM_C_TYPE *c*) `[inline, static]`

Set bank C to the indicated mapping.

Parameters:

c the mapping of the bank

Examples:

Graphics/3D/3D_Both_Screens/source/template.c, Graphics/3D/Paletted_Cube/source/main.cpp, Graphics/Backgrounds/all_in_one/source/basic.cpp, Graphics/Printing/custom_font/source/main.c, Graphics/Printing/print_both_screens/source/template.c, and Graphics/Printing/rotscale_text/source/main.c.

static void vramSetBankD (VRAM_D_TYPE *d*) `[inline, static]`

Set bank D to the indicated mapping.

Parameters:

d the mapping of the bank

Examples:

Graphics/3D/3D_Both_Screens/source/template.c, Graphics/Backgrounds/all_in_one/source/basic.cpp, Graphics/Sprites/allocation_test/source/main.c, Graphics/Sprites/animate_simple/source/template.c, Graphics/Sprites/bitmap_sprites/source/main.cpp, and Graphics/Sprites/simple/source/template.c.

static void vramSetBankE (VRAM_E_TYPE *e*) `[inline, static]`

Set bank E to the indicated mapping.

Parameters:

e the mapping of the bank

static void vramSetBankF (VRAM_F_TYPE *f*) `[inline, static]`

Set bank F to the indicated mapping.

Parameters:

f the mapping of the bank

Examples:

Graphics/3D/Paletted_Cube/source/main.cpp, and Graphics/Sprites/sprite_extended_palettes/source/template.c.

static void vramSetBankG (VRAM_G_TYPE *g*) `[inline, static]`

Set bank G to the indicated mapping.

Parameters:

g the mapping of the bank

Examples:

Graphics/3D/Paletted_Cube/source/main.cpp.

static void vramSetBankH (VRAM_H_TYPE *h*) `[inline, static]`

Set bank H to the indicated mapping.

Parameters:

h the mapping of the bank

static void vramSetBankI (VRAM_I_TYPE *i*) `[inline, static]`

Set bank I to the indicated mapping.

Parameters:

i the mapping of the bank

Examples:

Graphics/3D/Paletted_Cube/source/main.cpp.

```
u32 vramSetBanks_EFG ( VRAM_E_TYPE  e,
                       VRAM_F_TYPE  f,
                       VRAM_G_TYPE  g
                     )
```

Set E,F,G bank modes.

Parameters:

e	mapping mode of VRAM_E
f	mapping mode of VRAM_F
g	mapping mode of VRAM_G

Returns:

the previous mode

```
u32 vramSetPrimaryBanks  ( VRAM_A_TYPE  a,
                           VRAM_B_TYPE  b,
                           VRAM_C_TYPE  c,
                           VRAM_D_TYPE  d
                         )
```

Set the main 4 bank modes.

Parameters:

a	mapping mode of VRAM_A
b	mapping mode of VRAM_B
c	mapping mode of VRAM_C
d	mapping mode of VRAM_D

Returns:

the previous mode

Examples:

capture/ScreenShot/source/main.cpp, Graphics/Backgrounds/Double_Buffer/source/main.cpp, and Graphics/Sprites/fire_and_sprites/source/main.cpp.

background.h File Reference

nds background defines and functionality. More...

```
#include <nds/ndstypes.h>
#include <nds/arm9/video.h>
#include <nds/arm9/sassert.h>
#include <nds/memory.h>
#include <nds/dma.h>
```

Data Structures

struct bg_attribute
: register overlay for background attribute registers More...

struct bg_scroll
: register overlay for scroll registers More...

struct bg_transform
: register overlay for affine matrix registers More...

Defines

#define BACKGROUND (*((bg_attribute *)0x04000008))
: Overlay for main screen background attributes. Setting the properties of this struct directly sets background registers.

#define BACKGROUND_SUB (*((bg_attribute *)0x04001008))
: Overlay for sub screen background attributes. Setting the properties of this struct directly sets background registers.

#define BG_BMP_BASE(base) ((base) << MAP_BASE_SHIFT)
: Macro to set the graphics base in background control.

#define BG_BMP_RAM(base) ((u16*)(((base)*0x4000) + 0x06000000))
: A macro which returns a u16* pointer to background graphics memory ram (Main Engine)

#define BG_BMP_RAM_SUB(base) ((u16*)(((base)*0x4000) + 0x06200000))
: A macro which returns a u16* pointer to background graphics ram (Sub Engine)

#define BG_MAP_BASE(base) ((base) << MAP_BASE_SHIFT)
: Macro to set the map base in background control.

#define BG_MAP_RAM(base) ((u16*)(((base)*0x800) + 0x06000000))
: A macro which returns a u16* pointer to background map ram (Main Engine)

#define BG_MAP_RAM_SUB(base) ((u16*)(((base)*0x800) + 0x06200000))
: A macro which returns a u16* pointer to background map ram (Sub Engine)

#define BG_OFFSET ((bg_scroll *)(0x04000010))
: Overlay for main screen background scroll registers. Setting the properties of this struct directly sets background registers.

#define BG_OFFSET_SUB ((bg_scroll *)(0x04001010))

Overlay for sub screen background scroll registers. Setting the properties of this struct directly sets background registers.

#define BG_PRIORITY(n)   (n)

Macro to set the priority in background control.

#define BG_TILE_BASE(base)   ((base) << TILE_BASE_SHIFT)

Macro to set the tile base in background control.

#define BG_TILE_RAM(base)   ((u16*)(((base)*0x4000) + 0x06000000))

A macro which returns a u16* pointer to background tile ram (Main Engine)

#define BG_TILE_RAM_SUB(base)   ((u16*)(((base)*0x4000) + 0x06200000))

A macro which returns a u16* pointer to background tile ram (Sub Engine)

#define BGCTRL   ((vu16*)0x4000008)

Access to all Main screen background control registers via: BGCTRL[x] GBATEK Reference

#define BGCTRL_SUB   ((vu16*)0x4001008)

Access to all Sub screen background control registers via: BGCTRL[x] GBATEK Reference

#define MAP_BASE_SHIFT   8

The shift to apply to map base when storing it in a background control register.

#define REG_BG0CNT   (*(vu16*)0x4000008)

Background 0 Control register (main engine) GBATEK Reference

#define REG_BG0CNT_SUB   (*(vu16*)0x4001008)

Background 0 Control register (sub engine) GBATEK Reference

#define REG_BG0HOFS   (*(vu16*)0x4000010)

Background 0 horizontal scroll register (main engine)

#define REG_BG0HOFS_SUB   (*(vu16*)0x4001010)

Background 0 horizontal scroll register (sub engine)

#define REG_BG0VOFS   (*(vu16*)0x4000012)

Background 0 vertical scroll register (main engine)

#define REG_BG0VOFS_SUB   (*(vu16*)0x4001012)

Background 0 vertical scroll register (sub engine)

#define REG_BG1CNT   (*(vu16*)0x400000A)

Background 1 Control register (main engine) GBATEK Reference

#define REG_BG1CNT_SUB   (*(vu16*)0x400100A)

Background 1 Control register (sub engine) GBATEK Reference

#define REG_BG1HOFS   (*(vu16*)0x4000014)

Background 1 horizontal scroll register (main engine)

#define REG_BG1HOFS_SUB   (*(vu16*)0x4001014)

Background 1 horizontal scroll register (sub engine)

#define REG_BG1VOFS   (*(vu16*)0x4000016)

Background 1 vertical scroll register (main engine)

#define REG_BG1VOFS_SUB   (*(vu16*)0x4001016)

Background 1 vertical scroll register (sub engine)

#define REG_BG2CNT   (*(vu16*)0x400000C)

Background 2 Control register (main engine) GBATEK Reference

#define REG_BG2CNT_SUB (*(vu16*)0x400100C)
Background 2 Control register (sub engine) GBATEK Reference

#define REG_BG2HOFS (*(vu16*)0x4000018)
Background 2 horizontal scroll register (main engine)

#define REG_BG2HOFS_SUB (*(vu16*)0x4001018)
Background 2 horizontal scroll register (sub engine)

#define REG_BG2PA (*(vs16*)0x4000020)
Background 2 Affine transform (main engine)

#define REG_BG2PA_SUB (*(vs16*)0x4001020)
Background 2 Affine transform (sub engine)

#define REG_BG2PB (*(vs16*)0x4000022)
Background 2 Affine transform (main engine)

#define REG_BG2PB_SUB (*(vs16*)0x4001022)
Background 2 Affine transform (sub engine)

#define REG_BG2PC (*(vs16*)0x4000024)
Background 2 Affine transform (main engine)

#define REG_BG2PC_SUB (*(vs16*)0x4001024)
Background 2 Affine transform (sub engine)

#define REG_BG2PD (*(vs16*)0x4000026)
Background 2 Affine transform (main engine)

#define REG_BG2PD_SUB (*(vs16*)0x4001026)
Background 2 Affine transform (sub engine)

#define REG_BG2VOFS (*(vu16*)0x400001A)
Background 2 vertical scroll register (main engine)

#define REG_BG2VOFS_SUB (*(vu16*)0x400101A)
Background 2 vertical scroll register (sub engine)

#define REG_BG2X (*(vs32*)0x4000028)
Background 2 Screen Offset (main engine)

#define REG_BG2X_SUB (*(vs32*)0x4001028)
Background 2 Screen Offset (sub engine)

#define REG_BG2Y (*(vs32*)0x400002C)
Background 2 Screen Offset (main engine)

#define REG_BG2Y_SUB (*(vs32*)0x400102C)
Background 2 Screen Offset (sub engine)

#define REG_BG3CNT (*(vu16*)0x400000E)
Background 3 Control register (main engine) GBATEK Reference

#define REG_BG3CNT_SUB (*(vu16*)0x400100E)
Background 3 Control register (sub engine) GBATEK Reference

#define REG_BG3HOFS (*(vu16*)0x400001C)
Background 3 horizontal scroll register (main engine)

#define REG_BG3HOFS_SUB (*(vu16*)0x400101C)
Background 3 horizontal scroll register (sub engine)

#define	REG_BG3PA (*(vs16*)0x4000030)
	Background 3 Affine transform (main engine)
#define	REG_BG3PA_SUB (*(vs16*)0x4001030)
	Background 3 Affine transform (sub engine)
#define	REG_BG3PB (*(vs16*)0x4000032)
	Background 3 Affine transform (main engine)
#define	REG_BG3PB_SUB (*(vs16*)0x4001032)
	Background 3 Affine transform (sub engine)
#define	REG_BG3PC (*(vs16*)0x4000034)
	Background 3 Affine transform (main engine)
#define	REG_BG3PC_SUB (*(vs16*)0x4001034)
	Background 3 Affine transform (sub engine)
#define	REG_BG3PD (*(vs16*)0x4000036)
	Background 3 Affine transform (main engine)
#define	REG_BG3PD_SUB (*(vs16*)0x4001036)
	Background 3 Affine transform (sub engine)
#define	REG_BG3VOFS (*(vu16*)0x400001E)
	Background 3 vertical scroll register (main engine)
#define	REG_BG3VOFS_SUB (*(vu16*)0x400101E)
	Background 3 vertical scroll register (sub engine)
#define	REG_BG3X (*(vs32*)0x4000038)
	Background 3 Screen Offset (main engine)
#define	REG_BG3X_SUB (*(vs32*)0x4001038)
	Background 3 Screen Offset (sub engine)
#define	REG_BG3Y (*(vs32*)0x400003C)
	Background 3 Screen Offset (main engine)
#define	REG_BG3Y_SUB (*(vs32*)0x400103C)
	Background 3 Screen Offset (sub engine)
#define	TILE_BASE_SHIFT 2
	The shift to apply to tile base when storing it in a background control register.
#define	TILE_FLIP_H BIT(10)
	The horizontal flip bit for a 16 bit tile index.
#define	TILE_FLIP_V BIT(11)
	The vertical flip bit for a 16 bit tile index.
#define	TILE_PALETTE(n) ((n)<<12)
	Macro to set the palette entry of a 16 bit tile index.

Typedefs

typedef struct bg_attribute	bg_attribute
	register overlay for background attribute registers
typedef struct	bg_scroll

	bg_scroll
	register overlay for scroll registers
typedef struct bg_transform	bg_transform
	register overlay for affine matrix registers

Enumerations

```
enum  BackgroundControl {
        BG_32x32 = (0 << 14),
        BG_64x32 = (1 << 14),
        BG_32x64 = (2 << 14),
        BG_64x64 = (3 << 14),
        BG_RS_16x16 = (0 << 14),
        BG_RS_32x32 = (1 << 14),
        BG_RS_64x64 = (2 << 14),
        BG_RS_128x128 = (3 << 14),
        BG_BMP8_128x128 = ((0 << 14) | BIT(7)),
        BG_BMP8_256x256 = ((1 << 14) | BIT(7)),
        BG_BMP8_512x256 = ((2 << 14) | BIT(7)),
        BG_BMP8_512x512 = ((3 << 14) | BIT(7)),
        BG_BMP8_1024x512 = BIT(14),
        BG_BMP8_512x1024 = 0,
        BG_BMP16_128x128 = ((0 << 14) | BIT(7) | BIT(2)),
        BG_BMP16_256x256 = ((1 << 14) | BIT(7) | BIT(2)),
        BG_BMP16_512x256 = ((2 << 14) | BIT(7) | BIT(2)),
        BG_BMP16_512x512 = ((3 << 14) | BIT(7) | BIT(2)),
        BG_MOSAIC_ON = (BIT(6)),
        BG_MOSAIC_OFF = (0),
        BG_PRIORITY_0 = (0),
        BG_PRIORITY_1 = (1),
        BG_PRIORITY_2 = (2),
        BG_PRIORITY_3 = (3),
        BG_WRAP_OFF = (0),
        BG_WRAP_ON = (1 << 13),
        BG_PALETTE_SLOT0 = 0,
        BG_PALETTE_SLOT1 = 0,
        BG_PALETTE_SLOT2 = BIT(13),
        BG_PALETTE_SLOT3 = BIT(13),
        BG_COLOR_256 = 0x80,
        BG_COLOR_16 = 0x00
      }
```

Bit defines for the background control registers.

More...

```
enum  BgSize {
        BgSize_R_128x128 = (0 << 14),
        BgSize_R_256x256 = (1 << 14),
        BgSize_R_512x512 = (2 << 14),
        BgSize_R_1024x1024 = (3 << 14),
        BgSize_T_256x256 = (0 << 14) | (1 << 16),
```

BgSize_T_512x256 = (1 << 14) | (1 << 16),
BgSize_T_256x512 = (2 << 14) | (1 << 16),
BgSize_T_512x512 = (3 << 14) | (1 << 16),
BgSize_ER_128x128 = (0 << 14) | (2 << 16),
BgSize_ER_256x256 = (1 << 14) | (2 << 16),
BgSize_ER_512x512 = (2 << 14) | (2 << 16),
BgSize_ER_1024x1024 = (3 << 14) | (2 << 16),
BgSize_B8_128x128 = ((0 << 14) | BIT(7) | (3 << 16)),
BgSize_B8_256x256 = ((1 << 14) | BIT(7) | (3 << 16)),
BgSize_B8_512x256 = ((2 << 14) | BIT(7) | (3 << 16)),
BgSize_B8_512x512 = ((3 << 14) | BIT(7) | (3 << 16)),
BgSize_B8_1024x512 = (1 << 14) | (3 << 16),
BgSize_B8_512x1024 = (0) | (3 << 16),
BgSize_B16_128x128 = ((0 << 14) | BIT(7) | BIT(2) | (4 << 16)),
BgSize_B16_256x256 = ((1 << 14) | BIT(7) | BIT(2) | (4 << 16)),
BgSize_B16_512x256 = ((2 << 14) | BIT(7) | BIT(2) | (4 << 16)),
BgSize_B16_512x512 = ((3 << 14) | BIT(7) | BIT(2) | (4 << 16))
}

Allowed background Sizes The lower 16 bits of these defines can be used directly to set the background control register bits.

More...

enum BgType {
BgType_Text8bpp,
BgType_Text4bpp,
BgType_Rotation,
BgType_ExRotation,
BgType_Bmp8,
BgType_Bmp16
}

Allowed background types, used in bgInit and bgInitSub.

More...

Functions

static void bgClearControlBits (int id, u16 bits)
Clears the specified bits from the backgrounds control register.

static u16 * bgGetGfxPtr (int id)
Gets a pointer to the background graphics.

static int bgGetMapBase (int id)
Gets the current map base for the supplied background.

static u16 * bgGetMapPtr (int id)
Gets a pointer to the background map.

static int bgGetPriority (int id)
Gets the background priority.

static int bgGetTileBase (int id)
Gets the background tile base.

static void bgHide (int id)

	Hides the current background via the display control register.
static int	bgInit (int layer, BgType type, BgSize size, int mapBase, int tileBase)
	Initializes a background on the main display Sets up background control register with specified settings and defaults to 256 color mode for tiled backgrounds. Sets the rotation/scale attributes for rot/ex rot backgrounds to 1:1 scale and 0 angle of rotation.
static int	bgInitSub (int layer, BgType type, BgSize size, int mapBase, int tileBase)
	Initializes a background on the sub display Sets up background control register with specified settings and defaults to 256 color mode for tiled backgrounds. Sets the rotation/scale attributes for rot/ex rot backgrounds to 1:1 scale and 0 angle of rotation.
static void	bgMosaicDisable (int id)
	Disables mosaic on the specified background.
static void	bgMosaicEnable (int id)
	Enables mosaic on the specified background.
static void	bgRotate (int id, int angle)
	Rotates the background counter clockwise by the specified angle. (this rotation is cumulative)
static void	bgScroll (int id, int dx, int dy)
	Scrolls the background by the specified relative values.
static void	bgScrollf (int id, s32 dx, s32 dy)
	Scrolls the background by the specified relative values (fixed point)
static void	bgSet (int id, int angle, s32 sx, s32 sy, s32 scrollX, s32 scrollY, s32 rotCenterX, s32 rotCenterY)
	Sets the rotation and scale of the background and update background control registers.
static void	bgSetAffineMatrixScroll (int id, int hdx, int vdx, int hdy, int vdy, int scrollx, int scrolly)
	directly sets the affine matrix and scroll registers of a background.
static void	bgSetCenter (int id, int x, int y)
	Sets the center of rotation for the supplied background.
static void	bgSetCenterf (int id, s32 x, s32 y)
	Sets the center of rotation for the supplied background (fixed point)
static vuint16 *	bgSetControlBits (int id, u16 bits)
	allows direct access to background control for the chosen layer, returns a pointer to the current control bits
static void	bgSetMapBase (int id, unsigned int base)
	Sets the background map base.
static void	bgSetMosaic (unsigned int dx, unsigned int dy)
	Sets the horizontal and vertical mosaic values for all backgrounds.
static void	bgSetMosaicSub (unsigned int dx, unsigned int dy)
	Sets the horizontal and vertical mosaic values for all backgrounds (Sub Display)
static void	bgSetPriority (int id, unsigned int priority)
	Sets the background priority.

static void	bgSetRotate (int id, int angle)
	Sets the rotation angle of the specified background and updates the transform matrix.
static void	bgSetRotateScale (int id, int angle, s32 sx, s32 sy)
	Sets the rotation and scale of the background and update background control registers.
static void	bgSetScale (int id, s32 sx, s32 sy)
	Sets the scale of the specified background.
static void	bgSetScroll (int id, int x, int y)
	Sets the scroll hardware to the specified location.
static void	bgSetScrollf (int id, s32 x, s32 y)
	Sets the scroll hardware to the specified location (fixed point)
static void	bgSetTileBase (int id, unsigned int base)
	Sets the background map base.
static void	bgShow (int id)
	Shows the current background via the display control register.
void	bgUpdate (void)
	Must be called once per frame to update scroll/scale/and rotation of backgrounds.
static void	bgWrapOff (int id)
	turns wrap off for a background. has no effect on text backgrounds, which are always wrapped.
static void	bgWrapOn (int id)
	turns wrap on for a background. has no effect on text backgrounds, which are always wrapped.

Detailed Description

nds background defines and functionality.

Background control is provided via an API or Direct register access. Usually these methods can be mixed. However, scrolling, scaling, and rotation will have unexpected results if API and Direct Register access is mixed. Effort is being directed at ensuring the API can access all hardware features without limitation.

- API Components
- Register Access Components

The DS contains two separate hardware 2D cores responsible for rendering 2D backgrounds. The definitions below outline the libnds api for utilizing these backgrounds.
The background engine provides basic initialization and management of the 8 2D backgrounds available on the DS. Other than initialization and hardware limitations background control is identical on both main and sub screens.
The following modes of operation are allowed:

Main 2D engine

Mode	BG0	BG1	BG2	BG3
0	T	T	T	T
1	T	T	T	R
2	T	T	R	R
3	T	T	T	E
4	T	T	R	E
5	T	T	E	E
6		L		

T = Text
R = Rotation
E = Extended Rotation
L = Large Bitmap background

Sub 2D engine

Mode	BG0	BG1	BG2	BG3
0	T	T	T	T
1	T	T	T	R
2	T	T	R	R
3	T	T	T	E
4	T	T	R	E
5	T	T	E	E

On the main engine BG0 can be uses as a 3D rendering surface.

sprite.h File Reference

nds sprite functionality. More...

```
#include "nds/ndstypes.h"
#include "nds/arm9/video.h"
#include "nds/memory.h"
#include "nds/system.h"
```

Data Structures

struct	OamState
	Holds the state for a 2D sprite engine. More...
union	SpriteEntry
	A bitfield of sprite attribute goodness...ugly to look at but not so bad to use. More...
struct	SpriteRotation
	A sprite rotation entry. More...

Defines

#define	MATRIX_COUNT 32
	maximum number of affine matrices per engine available.
#define	SPRITE_COUNT 128
	maximum number of sprites per engine available.

Typedefs

typedef struct OamState	OamState
	Holds the state for a 2D sprite engine.
typedef union SpriteEntry	SpriteEntry
	A bitfield of sprite attribute goodness...ugly to look at but not so bad to use.
typedef struct SpriteRotation	SpriteRotation
	A sprite rotation entry.

Enumerations

enum ObjBlendMode {
OBJMODE_NORMAL,
OBJMODE_BLENDED,
OBJMODE_WINDOWED,
OBJMODE_BITMAP
}

The blending mode of the sprite.

More...

enum ObjColMode {
  OBJCOLOR_16,
  OBJCOLOR_256
}
The color mode of the sprite.

More...

enum ObjPriority {
  OBJPRIORITY_0,
  OBJPRIORITY_1,
  OBJPRIORITY_2,
  OBJPRIORITY_3
}
The priority of the sprite.

More...

enum ObjShape {
  OBJSHAPE_SQUARE,
  OBJSHAPE_WIDE,
  OBJSHAPE_TALL,
  OBJSHAPE_FORBIDDEN
}
The shape of the sprite.

More...

enum ObjSize {
  OBJSIZE_8,
  OBJSIZE_16,
  OBJSIZE_32,
  OBJSIZE_64
}
The size of the sprite.

More...

enum SpriteColorFormat {
  SpriteColorFormat_16Color = OBJCOLOR_16,
  SpriteColorFormat_256Color = OBJCOLOR_256,
  SpriteColorFormat_Bmp = OBJMODE_BITMAP
}
Color formats for sprite graphics.

More...

enum SpriteMapping {
  SpriteMapping_1D_32 = DISPLAY_SPR_1D | DISPLAY_SPR_1D_SIZE_32 | (0 << 28) | 0,
  SpriteMapping_1D_64 = DISPLAY_SPR_1D | DISPLAY_SPR_1D_SIZE_64 | (1 << 28) | 1,

SpriteMapping_1D_128 = DISPLAY_SPR_1D | DISPLAY_SPR_1D_SIZE_128 | (2 << 28) | 2,
SpriteMapping_1D_256 = DISPLAY_SPR_1D | DISPLAY_SPR_1D_SIZE_256 | (3 << 28) | 3,
SpriteMapping_2D = DISPLAY_SPR_2D | (4 << 28),
SpriteMapping_Bmp_1D_128 = DISPLAY_SPR_1D | DISPLAY_SPR_1D_SIZE_128 | DISPLAY_SPR_1D_BMP | DISPLAY_SPR_1D_BMP_SIZE_128 | (5 << 28) | 2,
SpriteMapping_Bmp_1D_256 = DISPLAY_SPR_1D | DISPLAY_SPR_1D_SIZE_256 | DISPLAY_SPR_1D_BMP | DISPLAY_SPR_1D_BMP_SIZE_256 | (6 << 28) | 3,
SpriteMapping_Bmp_2D_128 = DISPLAY_SPR_2D | DISPLAY_SPR_2D_BMP_128 | (7 << 28) | 2,
SpriteMapping_Bmp_2D_256 = DISPLAY_SPR_2D | DISPLAY_SPR_2D_BMP_256 | (8 << 28) | 3
}
Graphics memory layout options.

More...

enum SpriteSize {
SpriteSize_8x8 = (OBJSIZE_8 << 14) | (OBJSHAPE_SQUARE << 12) | (8*8>>5),
SpriteSize_16x16 = (OBJSIZE_16 << 14) | (OBJSHAPE_SQUARE << 12) | (16*16>>5),
SpriteSize_32x32 = (OBJSIZE_32 << 14) | (OBJSHAPE_SQUARE << 12) | (32*32>>5),
SpriteSize_64x64 = (OBJSIZE_64 << 14) | (OBJSHAPE_SQUARE << 12) | (64*64>>5),
SpriteSize_16x8 = (OBJSIZE_8 << 14) | (OBJSHAPE_WIDE << 12) | (16*8>>5),
SpriteSize_32x8 = (OBJSIZE_16 << 14) | (OBJSHAPE_WIDE << 12) | (32*8>>5),
SpriteSize_32x16 = (OBJSIZE_32 << 14) | (OBJSHAPE_WIDE << 12) | (32*16>>5),
SpriteSize_64x32 = (OBJSIZE_64 << 14) | (OBJSHAPE_WIDE << 12) | (64*32>>5),
SpriteSize_8x16 = (OBJSIZE_8 << 14) | (OBJSHAPE_TALL << 12) | (8*16>>5),
SpriteSize_8x32 = (OBJSIZE_16 << 14) | (OBJSHAPE_TALL << 12) | (8*32>>5),
SpriteSize_16x32 = (OBJSIZE_32 << 14) | (OBJSHAPE_TALL << 12) | (16*32>>5),
SpriteSize_32x64 = (OBJSIZE_64 << 14) | (OBJSHAPE_TALL << 12) | (32*64>>5)
}
Enumerates all sizes supported by the 2D engine.

More...

Functions

static void	oamAffineTransformation (OamState *oam, int rotId, int hdx, int hdy, int vdx, int vdy)
	allows you to directly sets the affine transformation matrix.
u16 *	oamAllocateGfx (OamState *oam, SpriteSize size, SpriteColorFormat colorFormat)
	Allocates graphics memory for the supplied sprite attributes.
void	oamClear (OamState *oam, int start, int count)
	Hides the sprites in the supplied range: if count is zero all 128 sprites will be hidden.
static void	oamClearSprite (OamState *oam, int index)
	Hides a single sprite.
int	oamCountFragments (OamState *oam)
	determines the number of fragments in the allocation engine
void	oamDisable (OamState *oam)
	Disables sprite rendering.
void	oamEnable (OamState *oam)
	Enables sprite rendering.
void	oamFreeGfx (OamState *oam, const void *gfxOffset)
	free vram memory obtained with oamAllocateGfx.
u16 *	oamGetGfxPtr (OamState *oam, int gfxOffsetIndex)
	translates an oam offset into a video ram address
void	oamInit (OamState *oam, SpriteMapping mapping, bool extPalette)
	Initializes the 2D sprite engine In order to mix tiled and bitmap sprites use SpriteMapping_Bmp_1D_128 or SpriteMapping_Bmp_1D_256.
void	oamRotateScale (OamState *oam, int rotId, int angle, int sx, int sy)
	sets the specified rotation scale entry
void	oamSet (OamState *oam, int id, int x, int y, int priority, int palette_alpha, SpriteSize size, SpriteColorFormat format, const void *gfxOffset, int affineIndex, bool sizeDouble, bool hide, bool hflip, bool vflip, bool mosaic)
	sets an oam entry to the supplied values
static void	oamSetMosaic (unsigned int dx, unsigned int dy)
	sets engine A global sprite mosaic
static void	oamSetMosaicSub (unsigned int dx, unsigned int dy)
	sets engine B global sprite mosaic
void	oamUpdate (OamState *oam)
	causes oam memory to be updated...must be called during vblank if using oam api

Variables

OamState	oamMain
	oamMain an object representing the main 2D engine
OamState	oamSub
	oamSub an object representing the sub 2D engine

Detailed Description

nds sprite functionality.

Typedef Documentation

typedef struct OamState OamState

Holds the state for a 2D sprite engine.

There are two of these objects, oamMain and oamSub and these must be passed in to all oam functions.

Enumeration Type Documentation

enum ObjBlendMode

The blending mode of the sprite.

Enumerator:

OBJMODE_NORMAL	No special mode is on - Normal sprite state.
OBJMODE_BLENDED	Color blending is on - Sprite can use HW blending features.
OBJMODE_WINDOWED	Sprite can be seen only inside the sprite window.
OBJMODE_BITMAP	Sprite is not using tiles - per pixel image data.

enum ObjColMode

The color mode of the sprite.

Enumerator:

OBJCOLOR_16	sprite has 16 colors.
OBJCOLOR_256	sprite has 256 colors.

enum ObjPriority

The priority of the sprite.

Enumerator:

OBJPRIORITY_0	sprite priority level 0 - highest.
OBJPRIORITY_1	sprite priority level 1.
OBJPRIORITY_2	sprite priority level 2.
OBJPRIORITY_3	sprite priority level 3 - lowest.

enum ObjShape

The shape of the sprite.

Enumerator:

OBJSHAPE_SQUARE	Sprite shape is NxN (Height == Width).
OBJSHAPE_WIDE	Sprite shape is NxM with N > M (Height < Width).
OBJSHAPE_TALL	Sprite shape is NxM with N < M (Height > Width).
OBJSHAPE_FORBIDDEN	Sprite shape is undefined.

enum ObjSize
The size of the sprite.

Enumerator:

OBJSIZE_8	Major sprite size is 8px.
OBJSIZE_16	Major sprite size is 16px.
OBJSIZE_32	Major sprite size is 32px.
OBJSIZE_64	Major sprite size is 64px.

enum SpriteColorFormat
Color formats for sprite graphics.

Enumerator:

SpriteColorFormat_16Color	16 colors per sprite
SpriteColorFormat_256Color	256 colors per sprite
SpriteColorFormat_Bmp	16-bit sprites

enum SpriteMapping
Graphics memory layout options.

Enumerator:

SpriteMapping_1D_32	1D tile mapping 32 byte boundary between offset
SpriteMapping_1D_64	1D tile mapping 64 byte boundary between offset
SpriteMapping_1D_128	1D tile mapping 128 byte boundary between offset
SpriteMapping_1D_256	1D tile mapping 256 byte boundary between offset
SpriteMapping_2D	2D tile mapping 32 byte boundary between offset
SpriteMapping_Bmp_1D_128	1D bitmap mapping 128 byte boundary between offset
SpriteMapping_Bmp_1D_256	1D bitmap mapping 256 byte boundary between offset
SpriteMapping_Bmp_2D_128	2D bitmap mapping 128 pixels wide bitmap
SpriteMapping_Bmp_2D_256	2D bitmap mapping 256 pixels wide bitmap

enum SpriteSize

Enumerates all sizes supported by the 2D engine.

Enumerator:

SpriteSize_8x8	8x8
SpriteSize_16x16	16x16
SpriteSize_32x32	32x32
SpriteSize_64x64	64x64
SpriteSize_16x8	16x8
SpriteSize_32x8	32x8
SpriteSize_32x16	32x16
SpriteSize_64x32	64x32
SpriteSize_8x16	8x16
SpriteSize_8x32	8x32
SpriteSize_16x32	16x32
SpriteSize_32x64	32x64

Function Documentation

```
static void oamAffineTransformation ( OamState *  oam,
                                      int         rotId,
                                      int         hdx,
                                      int         hdy,
                                      int         vdx,
                                      int         vdy
                                    )             [inline, static]
```

allows you to directly sets the affine transformation matrix.

with this, you have more freedom to set the matrix, but it might be more difficult to use if you're not used to affine transformation matrix. this will erase the previous matrix stored at rotId.

Parameters:

oam	The oam engine, must be &oamMain or &oamSub.
rotId	The id of the rotscale item you want to change, must be 0-31.
hdx	The change in x per horizontal pixel.

hdy	The change in y per horizontal pixel.
vdx	The change in x per vertical pixel.
vdy	The change in y per vertical pixel.

```
u16* oamAllocateGfx  ( OamState *         oam,
                       SpriteSize         size,
                       SpriteColorFormat  colorFormat
                     )
```

Allocates graphics memory for the supplied sprite attributes.

Parameters:

oam	must be: &oamMain or &oamSub
size	the size of the sprite to allocate
colorFormat	the color format of the sprite

Returns:

the address in vram of the allocated sprite

Examples:

audio/maxmod/song_events_example/source/template.c, audio/maxmod/song_events_example2/source/template.c, Graphics/Sprites/allocation_test/source/main.c, Graphics/Sprites/animate_simple/source/template.c, Graphics/Sprites/bitmap_sprites/source/main.cpp, Graphics/Sprites/simple/source/template.c, Graphics/Sprites/sprite_extended_palettes/source/template.c, and Graphics/Sprites/sprite_rotate/source/template.c.

```
void oamClear  ( OamState *  oam,
                 int         start,
                 int         count
               )
```

Hides the sprites in the supplied range: if count is zero all 128 sprites will be hidden.

Parameters:

oam	must be: &oamMain or &oamSub
start	The first index to clear
count	The number of sprites to clear

```
static void oamClearSprite  ( OamState *  oam,
                              int         index
                            )             [inline, static]
```

Hides a single sprite.

Parameters:

oam	the oam engine, must be &oamMain or &oamSub.
index	the index of the sprite, must be 0-127.

```
int oamCountFragments  ( OamState *  oam  )
```

determines the number of fragments in the allocation engine

Parameters:

oam	must be: &oamMain or &oamSub

Returns:
the number of fragments.

void oamDisable (OamState * *oam*)
Disables sprite rendering.

Parameters:

oam	must be: &oamMain or &oamSub

void oamEnable (OamState * *oam*)
Enables sprite rendering.

Parameters:

oam	must be: &oamMain or &oamSub

void oamFreeGfx (OamState * *oam*,
const void * *gfxOffset*
)
free vram memory obtained with oamAllocateGfx.

Parameters:

oam	must be: &oamMain or &oamSub
gfxOffset	a vram offset obtained from oamAllocateGfx

Examples:
Graphics/Sprites/allocation_test/source/main.c.

u16* oamGetGfxPtr (OamState * *oam*,
int *gfxOffsetIndex*
)
translates an oam offset into a video ram address

Parameters:

oam	must be: &oamMain or &oamSub
gfxOffsetIndex	the index to compute

Returns:
the address in vram corresponding to the supplied offset

void oamInit (OamState * *oam*,
SpriteMapping *mapping*,
bool *extPalette*
)
Initializes the 2D sprite engine In order to mix tiled and bitmap sprites use SpriteMapping_Bmp_1D_128 or SpriteMapping_Bmp_1D_256.

This will set mapping for both to 1D and give same sized boundaries so the sprite gfx allocation will function. VBlank IRQ must be enabled for this function to work.

Parameters:

oam	must be: &oamMain or &oamSub
mapping	the mapping mode
extPalette	if true the engine sets up extended palettes for 8bpp sprites

Examples:
audio/maxmod/song_events_example/source/template.c,

audio/maxmod/song_events_example2/source/template.c, Graphics/3D/3D_Both_Screens/source/template.c, Graphics/Sprites/allocation_test/source/main.c, Graphics/Sprites/animate_simple/source/template.c, Graphics/Sprites/bitmap_sprites/source/main.cpp, Graphics/Sprites/simple/source/template.c, Graphics/Sprites/sprite_extended_palettes/source/template.c, and Graphics/Sprites/sprite_rotate/source/template.c.

```
void oamRotateScale  ( OamState *  oam,
                       int         rotId,
                       int         angle,
                       int         sx,
                       int         sy
                     )
```

sets the specified rotation scale entry

Parameters:

oam	must be: &oamMain or &oamSub
rotId	the rotation entry to set
angle	the ccw angle to rotate [-32768 - 32767]
sx	the inverse scale factor in the x direction
sy	the inverse scale factor in the y direction

Examples:

Graphics/Sprites/bitmap_sprites/source/main.cpp, and Graphics/Sprites/sprite_rotate/source/template.c.

```
void oamSet  ( OamState *          oam,
               int                 id,
               int                 x,
               int                 y,
               int                 priority,
               int                 palette_alpha,
               SpriteSize          size,
               SpriteColorFormat   format,
               const void *        gfxOffset,
               int                 affineIndex,
               bool                sizeDouble,
               bool                hide,
               bool                hflip,
               bool                vflip,
               bool                mosaic
             )
```

sets an oam entry to the supplied values

Parameters:

oam	must be: &oamMain or &oamSub

id	the oam number to be set [0 - 127]
x	the x location of the sprite in pixels
y	the y location of the sprite in pixels
priority	The sprite priority (0 to 3)
palette_alpha	the palette number for 4bpp and 8bpp (extended palette mode), or the alpha value for bitmap sprites (bitmap sprites must specify a value > 0 to display) [0-15]
size	the size of the sprite
format	the color format of the sprite
gfxOffset	the video memory address of the sprite graphics (not an offset)
affineIndex	affine index to use (if < 0 or > 31 the sprite will be unrotated)
sizeDouble	if affineIndex >= 0 this will be used to double the sprite size for rotation
hide	if non zero (true) the sprite will be hidden
vflip	flip the sprite vertically
hflip	flip the sprite horizontally
mosaic	if true mosaic will be applied to the sprite

Examples:

audio/maxmod/song_events_example/source/template.c, audio/maxmod/song_events_example2/source/template.c, Graphics/Sprites/allocation_test/source/main.c, Graphics/Sprites/animate_simple/source/template.c, Graphics/Sprites/bitmap_sprites/source/main.cpp, Graphics/Sprites/simple/source/template.c, Graphics/Sprites/sprite_extended_palettes/source/template.c, and Graphics/Sprites/sprite_rotate/source/template.c.

```
static void oamSetMosaic ( unsigned int dx,
                           unsigned int dy
                         )              [inline, static]
```

sets engine A global sprite mosaic

Parameters:

dx	(0-15) horizontal mosaic value
dy	(0-15) horizontal mosaic value

```
static void oamSetMosaicSub ( unsigned int dx,
                              unsigned int dy
                            )              [inline, static]
```

sets engine B global sprite mosaic

Parameters:

dx	(0-15) horizontal mosaic value
dy	(0-15) horizontal mosaic value

```
void oamUpdate ( OamState * oam )
```

causes oam memory to be updated...must be called during vblank if using oam api

Parameters:

oam	must be: &oamMain or &oamSub

Examples:

audio/maxmod/song_events_example/source/template.c,
audio/maxmod/song_events_example2/source/template.c,
Graphics/3D/3D_Both_Screens/source/template.c,
Graphics/Sprites/allocation_test/source/main.c,
Graphics/Sprites/animate_simple/source/template.c,
Graphics/Sprites/bitmap_sprites/source/main.cpp,
Graphics/Sprites/simple/source/template.c,
Graphics/Sprites/sprite_extended_palettes/source/template.c, and
Graphics/Sprites/sprite_rotate/source/template.c.

3D engine API

videoGL.h File Reference

openGL (ish) interface to DS 3D hardware. More...

```
#include "nds/dma.h"
#include "nds/ndstypes.h"
#include "nds/arm9/sassert.h"
#include "nds/arm9/video.h"
#include "nds/arm9/cache.h"
#include "nds/arm9/trig_lut.h"
#include "nds/arm9/math.h"
#include "nds/arm9/dynamicArray.h"
```

Data Structures

struct GLvector
: Holds a Vector
  related functions: glScalev(), glTranslatev() More...

struct m3x3
: Holds a Matrix of 3x3. More...

struct m4x3
: Holds a Matrix of 4x3. More...

struct m4x4
: Holds a Matrix of 4x4. More...

Defines

#define f32tofloat(n) (((float)(n)) / (float)(1<<12))
: convert f32 to float

#define f32toint(n) ((n) >> 12)
: convert f32 to int

#define f32tot16(n) ((t16)(n >> 8))
: convert f32 to t16

#define f32tov10(n) ((v10)(n >> 3))
: convert f32 to v10

#define f32tov16(n) (n)
: f32 to v16

#define FIFO_BEGIN REG2ID(GFX_BEGIN)
: packed command that starts a polygon vertex list
  GBATEK
  http://nocash.emubase.de/gbatek.htm#ds3dpolygondefinitionsbyvertices

#define FIFO_CLEAR_COLOR REG2ID(GFX_CLEAR_COLOR)
: packed command for clear color of the rear plane
  GBATEK http://nocash.emubase.de/gbatek.htm#ds3drearplane

#define FIFO_CLEAR_DEPTH   REG2ID(GFX_CLEAR_DEPTH)

sets depth of the rear plane
GBATEK http://nocash.emubase.de/gbatek.htm#ds3drearplane

#define FIFO_COLOR   REG2ID(GFX_COLOR)

packed command for vertex color directly
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dpolygonattributes

#define FIFO_COMMAND_PACK(c1, c2, c3, c4)   (((c4) << 24) | ((c3) << 16) | ((c2) << 8) | (c1))

packs four packed commands into a 32bit command for sending to the GFX FIFO

#define FIFO_DIFFUSE_AMBIENT   REG2ID(GFX_DIFFUSE_AMBIENT)

packed command for setting diffuse and ambient material properties for the following vertices
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dpolygonlightparameters

#define FIFO_END   REG2ID(GFX_END)

packed command that has no discernable effect, it's probably best to never use it since it bloats the size of the list.
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dpolygondefinitionsbyvertices

#define FIFO_FLUSH   REG2ID(GFX_FLUSH)

packed command that has the same effect as swiWaitForVBlank()
GBATEK http://nocash.emubase.de/gbatek.htm#ds3ddisplaycontrol

#define FIFO_LIGHT_COLOR   REG2ID(GFX_LIGHT_COLOR)

packed command for color for a light
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dpolygonlightparameters

#define FIFO_LIGHT_VECTOR   REG2ID(GFX_LIGHT_VECTOR)

packed command for direction of a light source
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dpolygonlightparameters

#define FIFO_NOP   REG2ID(GFX_FIFO)

packed command for nothing, just here to pad your command lists

#define FIFO_NORMAL   REG2ID(GFX_NORMAL)

packed command for normal for following vertices
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dpolygonlightparameters

#define FIFO_PAL_FORMAT   REG2ID(GFX_PAL_FORMAT)

packed command for texture palette attributes
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dtextureattributes

#define FIFO_POLY_FORMAT   REG2ID(GFX_POLY_FORMAT)

packed command for setting polygon attributes
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dpolygonattributes

#define FIFO_SHININESS   REG2ID(GFX_SHININESS)

packed command for setting the shininess table to be used for the following vertices
GBATEK

http://nocash.emubase.de/gbatek.htm#ds3dpolygonlightparameters

#define FIFO_SPECULAR_EMISSION   REG2ID(GFX_SPECULAR_EMISSION)

packed command for setting specular and emmissive material properties for the following vertices
GBATEK
http://nocash.emubase.de/gbatek.htm#ds3dpolygonlightparameters

#define FIFO_STATUS   REG2ID(GFX_STATUS)

packed command for geometry engine status register
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dstatus

#define FIFO_TEX_COORD   REG2ID(GFX_TEX_COORD)

packed command for a texture coordinate
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dtexturecoordinates

#define FIFO_TEX_FORMAT   REG2ID(GFX_TEX_FORMAT)

packed command for texture format
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dtextureformats

#define FIFO_VERTEX10   REG2ID(GFX_VERTEX10)

packed command for a vertex with 3 10bit paramaters (and 2bits of padding)
GBATEK
http://nocash.emubase.de/gbatek.htm#ds3dpolygondefinitionsbyvertices

#define FIFO_VERTEX16   REG2ID(GFX_VERTEX16)

packed command for a vertex with 3 16bit paramaters (and 16bits of padding)
GBATEK
http://nocash.emubase.de/gbatek.htm#ds3dpolygondefinitionsbyvertices

#define FIFO_VERTEX_XY   REG2ID(GFX_VERTEX_XY)

packed command for a vertex with 2 16bit paramaters (reusing current last-set vertex z value)
GBATEK
http://nocash.emubase.de/gbatek.htm#ds3dpolygondefinitionsbyvertices

#define FIFO_VERTEX_XZ   REG2ID(GFX_VERTEX_XZ)

packed command for a vertex with 2 16bit paramaters (reusing current last-set vertex y value)
GBATEK
http://nocash.emubase.de/gbatek.htm#ds3dpolygondefinitionsbyvertices

#define FIFO_VERTEX_YZ   REG2ID(GFX_VERTEX_YZ)

packed command for a vertex with 2 16bit paramaters (reusing current last-set vertex x value)
GBATEK
http://nocash.emubase.de/gbatek.htm#ds3dpolygondefinitionsbyvertices

#define FIFO_VIEWPORT   REG2ID(GFX_VIEWPORT)

packed command for setting viewport
GBATEK http://nocash.emubase.de/gbatek.htm#ds3ddisplaycontrol

#define floatto12d3(n)   ((fixed12d3)((n) * (1 << 3)))

convert float to fixed12d3

#define floattof32(n)   ((int)((n) * (1 << 12)))

convert float to f32

#define floattot16(n)   ((t16)((n) * (1 << 4))) convert float to t16

#define	floattov10(n) ((n>.998) ? 0x1FF : ((v10)((n)*(1<<9))))
	convert float to v10
#define	floattov16(n) ((v16)((n) * (1 << 12)))
	convert float to v16
#define	GL_MAX_DEPTH 0x7FFF
	the maximum value for type fixed12d3
#define	intto12d3(n) ((n) << 3)
	convert int to fixed12d3
#define	inttof32(n) ((n) << 12)
	convert int to f32
#define	inttot16(n) ((n) << 4)
	convert int to t16
#define	inttov10(n) ((n) << 9)
	convert int to v10
#define	inttov16(n) ((n) << 12)
	convert int to v16
#define	NORMAL_PACK(x, y, z) (((x) & 0x3FF) | (((y) & 0x3FF) << 10) | ((z) << 20))
	Pack 3 v10 normals into a 32bit value.
#define	REG2ID(r) (u8)((((u32)(&(r)))-0x04000400) >> 2)
	converts a GFX command for use in a packed command list
#define	t16toint(n) ((n) >> 4)
	convert t16 to int
#define	TEXTURE_PACK(u, v) (((u) & 0xFFFF) | ((v) << 16))
	Pack 2 t16 texture coordinate values into a 32bit value.
#define	v10toint(n) ((n) >> 9)
	convert v10 to int
#define	v16toint(n) ((n) >> 12)
	convert v16 to int
#define	VERTEX_PACK(x, y) (((x) & 0xFFFF) | ((y) << 16))
	Pack to v16 values into one 32bit value.

Typedefs

typedef uint16	fixed12d3
	Used for depth (glClearDepth, glCutoffDepth)
typedef struct GLvector	GLvector
	Holds a Vector related functions: glScalev(), glTranslatev()
typedef struct m3x3	m3x3
	Holds a Matrix of 3x3.

typedef struct m4x3	m4x3
	Holds a Matrix of 4x3.
typedef struct m4x4	m4x4
	Holds a Matrix of 4x4.
typedef unsigned short	rgb
	Holds a color value. 1bit alpha, 5bits red, 5bits green, 5bits blue.
typedef short	t16
	text coordinate 12.4 fixed point
typedef short int	v10
	normal .10 fixed point, NOT USED FOR 10bit VERTEXES!!!
typedef short int	v16
	vertex 4.12 fixed format

Enumerations

enum DISP3DCNT_ENUM {
  GL_TEXTURE_2D = (1<<0),
  GL_TOON_HIGHLIGHT = (1<<1),
  GL_ALPHA_TEST = (1<<2),
  GL_BLEND = (1<<3),
  GL_ANTIALIAS = (1<<4),
  GL_OUTLINE = (1<<5),
  GL_FOG_ONLY_ALPHA = (1<<6),
  GL_FOG = (1<<7),
  GL_COLOR_UNDERFLOW = (1<<12),
  GL_POLY_OVERFLOW = (1<<13),
  GL_CLEAR_BMP = (1<<14)
}

3D Display Control Register Enums
GBATEK http://nocash.emubase.de/gbatek.htm#ds3ddisplaycontrol
related functions: glEnable(), glDisable(), glInit()

More...

enum GL_GET_ENUM {
  GL_GET_VERTEX_RAM_COUNT,
  GL_GET_POLYGON_RAM_COUNT,
  GL_GET_MATRIX_VECTOR,
  GL_GET_MATRIX_POSITION,
  GL_GET_MATRIX_PROJECTION,
  GL_GET_MATRIX_CLIP,
  GL_GET_TEXTURE_WIDTH,
  GL_GET_TEXTURE_HEIGHT

}
Enums for reading stuff from the geometry engine
http://nocash.emubase.de/gbatek.htm#ds3diomap
related functions: glGetInt(), glGetFixed()

More...

```
enum  GL_GLBEGIN_ENUM {
        GL_TRIANGLES = 0,
        GL_QUADS = 1,
        GL_TRIANGLE_STRIP = 2,
        GL_QUAD_STRIP = 3,
        GL_TRIANGLE = 0,
        GL_QUAD = 1
      }
```

Enums selecting polygon draw mode
GBATEK
http://nocash.emubase.de/gbatek.htm#ds3dpolygondefinitionsbyvertices
related functions: glBegin()

More...

```
enum  GL_MATERIALS_ENUM {
        GL_AMBIENT = 0x01,
        GL_DIFFUSE = 0x02,
        GL_AMBIENT_AND_DIFFUSE = 0x03,
        GL_SPECULAR = 0x04,
        GL_SHININESS = 0x08,
        GL_EMISSION = 0x10
      }
```

Enums for setting up materials
GBATEK
http://nocash.emubase.de/gbatek.htm#ds3dpolygonlightparameters
related functions: glMaterialf()

More...

```
enum  GL_MATRIX_MODE_ENUM {
        GL_PROJECTION = 0,
        GL_POSITION = 1,
        GL_MODELVIEW = 2,
        GL_TEXTURE = 3
      }
```

Enums selecting matrix mode
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dmatrixloadmultiply
related functions: glMatrixMode()

More...

```
enum  GL_POLY_FORMAT_ENUM {
        POLY_FORMAT_LIGHT0 = (1<<0),
        POLY_FORMAT_LIGHT1 = (1<<1),
        POLY_FORMAT_LIGHT2 = (1<<2),
        POLY_FORMAT_LIGHT3 = (1<<3),
  POLY_MODULATION = (0<<4),
  POLY_DECAL = (1<<4),
  POLY_TOON_HIGHLIGHT = (2<<4),
  POLY_SHADOW = (3<<4),
  POLY_CULL_FRONT = (1<<6),
  POLY_CULL_BACK = (2<<6),
  POLY_CULL_NONE = (3<<6),
  POLY_FOG = (1<<15)
}
```

Enums for setting how polygons will be displayed
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dpolygonattributes
related functions: glPolyFmt(), glInit(), POLY_ALPHA(), POLY_ID()

More...

```
enum GL_TEXTURE_PARAM_ENUM {
  GL_TEXTURE_WRAP_S = (1 << 16),
  GL_TEXTURE_WRAP_T = (1 << 17),
  GL_TEXTURE_FLIP_S = (1 << 18),
  GL_TEXTURE_FLIP_T = (1 << 19),
  GL_TEXTURE_COLOR0_TRANSPARENT = (1<<29),
  TEXGEN_OFF = (0<<30),
  TEXGEN_TEXCOORD = (1<<30),
  TEXGEN_NORMAL = (2<<30),
  TEXGEN_POSITION = (3<<30)
}
```

Enums for texture parameters, such as texture wrapping and texture coord stuff
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dtextureattributes
related functions: glTexImage2d(), glTexParameter()

More...

```
enum GL_TEXTURE_SIZE_ENUM {
  TEXTURE_SIZE_8 = 0,
  TEXTURE_SIZE_16 = 1,
  TEXTURE_SIZE_32 = 2,
  TEXTURE_SIZE_64 = 3,
  TEXTURE_SIZE_128 = 4,
  TEXTURE_SIZE_256 = 5,
  TEXTURE_SIZE_512 = 6,
  TEXTURE_SIZE_1024 = 7
}
```

Enums for size of a texture, specify one for horizontal and one for vertical
related functions: glTexImage2d(), glTexParameter()

More...

```
enum GL_TEXTURE_TYPE_ENUM {
  GL_RGB32_A3 = 1,
  GL_RGB4 = 2,
  GL_RGB16 = 3,
  GL_RGB256 = 4,
  GL_COMPRESSED = 5,
```

  GL_RGB8_A5 = 6,
  GL_RGBA = 7,
  GL_RGB = 8
}
Enums for texture formats
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dtextureformats
related functions: glTexImage2d(), glTexParameter()

More...

enum GLFLUSH_ENUM {
  GL_TRANS_MANUALSORT = (1<<0),
  GL_WBUFFERING = (1<<1)
}
Enums for glFlush()
GBATEK http://nocash.emubase.de/gbatek.htm#ds3ddisplaycontrol
related functions: glEnable(), glDisable(), glInit()

More...

Functions

void glAlphaFunc (int alphaThreshold)

set the minimum alpha value that will be used
GBATEK http://nocash.emubase.de/gbatek.htm#ds3ddisplaycontrol

void glAssignColorTable (int target, int name)

nglAssignColorTable sets the active texture with a palette set with another texture

void glBegin (GL_GLBEGIN_ENUM mode)

Starts a polygon group.

void glBindTexture (int target, int name)

glBindTexure sets the current named texture to the active texture. Target is ignored as all DS textures are 2D

void glCallList (const u32 *list)

throws a packed list of commands into the graphics FIFO via asyncronous DMA
The first 32bits is the length of the packed command list, followed by a the packed list.
If you want to do this really fast then write your own code that that does this synchronously and only flushes the cache when the list is changed
There is sometimes a problem when you pack the GFX_END command into a list, so don't. GFX_END is a dummy command and never needs called
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dgeometrycommands

void glClearColor (uint8 red, uint8 green, uint8 blue, uint8 alpha)

sets the color of the rear-plane(a.k.a Clear Color/Plane)

void glClearDepth (fixed12d3 depth)

reset the depth buffer to this value; generally set this to GL_MAX_DEPTH.
GBATEK http://nocash.emubase.de/gbatek.htm#ds3drearplane

void glClearPolyID (uint8 ID)

sets the polygon ID of the rear-plane(a.k.a. Clear/Color Plane), useful for antialiasing and edge coloring

void glColor (rgb color)

Set the color for following vertices.

void glColor3b (uint8 red, uint8 green, uint8 blue)

Set the color for following vertices.

void glColor3f (float r, float g, float b)

specify a color for following vertices

void glColorTableEXT (int target, int empty1, uint16 width, int empty2, int empty3, const uint16 *table)

nglColorTableEXT loads a 15-bit color format palette into palette memory, and sets it to the currently bound texture (can be used to remove also)

void glCutoffDepth (fixed12d3 wVal)

Stop the drawing of polygons that are a certain distance from the camera.
GBATEK http://nocash.emubase.de/gbatek.htm#ds3ddisplaycontrol

int glDeleteTextures (int n, int *names)

Deletes the specified number of textures (and associated palettes)

void glDisable (int bits)

Disables various gl states (blend, alpha test, etc..)

void glEnable (int bits)

Enables various gl states (blend, alpha test, etc..)

void glEnd (void)

Ends a polygon group, this seems to be a dummy function that does absolutely nothing, feel free to never use it.

void glFlush (u32 mode)

Waits for a Vblank and swaps the buffers(like swiWaitForVBlank), but lets you specify some 3D options
GBATEK http://nocash.emubase.de/gbatek.htm#ds3ddisplaycontrol

void glFogColor (uint8 red, uint8 green, uint8 blue, uint8 alpha)

sets the fog color

void glFogDensity (int index, int density)

sets the fog density at a given index

void glFogOffset (int offset)

Sets the FOG_OFFSET value.

void glFogShift (int shift)

Sets the FOG_SHIFT value.

void glFrustum (float left, float right, float bottom, float top, float near, float far)

Specifies the viewing frustum for the projection matrix (floating point version)

void glFrustumf32 (int left, int right, int bottom, int top, int near, int far)

Specifies the viewing frustum for the projection matrix (fixed point version)

int glGenTextures (int n, int *names)

Creates room for the specified number of textures.

void glGetFixed (const GL_GET_ENUM param, int *f)

Grabs fixed format of state variables
OpenGL's modelview matrix is handled on the DS with two matrices. The

	combination of the DS's position matrix and directional vector matrix hold the data that is in OpenGL's one modelview matrix. (a.k.a. modelview = postion and vector) http://nocash.emubase.de/gbatek.htm#ds3diomap
void	glGetInt (GL_GET_ENUM param, int *i)
	Grabs integer state variables from openGL.
u32	glGetTexParameter (void)
	Returns the active texture parameter (constructed from internal call to glTexParameter)
void *	glGetTexturePointer (int name)
	returns the address alocated to the texure named by name
void	glInit ()
	Initializes the gl state machine (must be called once before using gl calls)
void	glLight (int id, rgb color, v10 x, v10 y, v10 z)
	set a light up. Only parallel light sources are supported on the DS GBATEK http://nocash.emubase.de/gbatek.htm#ds3dpolygonlightparameters
void	glLoadIdentity (void)
	loads an identity matrix to the current matrix, same as glIdentity(void)
void	glLoadMatrix4x3 (const m4x3 *m)
	Loads a 4x3 matrix into the current matrix.
void	glLoadMatrix4x4 (const m4x4 *m)
	Loads a 4x4 matrix into the current matrix.
void	glMaterialf (GL_MATERIALS_ENUM mode, rgb color)
	specify the material properties to be used in rendering lit polygons
void	glMaterialShinyness (void)
	The DS uses a table for shininess..this generates a half-ass one.
void	glMatrixMode (GL_MATRIX_MODE_ENUM mode)
	change the current matrix mode GBATEK http://nocash.emubase.de/gbatek.htm#ds3dmatrixloadmultiply
void	glMultMatrix3x3 (const m3x3 *m)
	multiplies the current matrix by m
void	glMultMatrix4x3 (const m4x3 *m)
	multiplies the current matrix by
void	glMultMatrix4x4 (const m4x4 *m)
	Multiplies the current matrix by m.
void	glNormal (u32 normal)
	the normal to use for following vertices GBATEK http://nocash.emubase.de/gbatek.htm#ds3dpolygonlightparameters
void	glNormal3f (float x, float y, float z)
	the normal to use for following vertices GBATEK http://nocash.emubase.de/gbatek.htm#ds3dpolygonlightparameters
void	glOrtho (float left, float right, float bottom, float top, float zNear, float zFar)

Multiplies the current matrix into ortho graphic mode.

void glOrthof32 (int left, int right, int bottom, int top, int zNear, int zFar)

Multiplies the current matrix into ortho graphic mode.

void glPolyFmt (u32 params)

Set the parameters for polygons rendered on the current frame
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dpolygonattributes

void glPopMatrix (int num)

Pops num matrices off the stack
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dmatrixstack

void glPushMatrix (void)

Pushes the current matrix onto the stack
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dmatrixstack

void glResetMatrixStack (void)

Resets matrix stack to top level.

void glResetTextures (void)

Resets the gl texture state freeing all texture and texture palette memory.

void glRestoreMatrix (int index)

Restores the current matrix from a location in the stack
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dmatrixstack

void glRotatef (float angle, float x, float y, float z)

Rotate about an arbitrary axis.

void glRotatef32 (float angle, int x, int y, int z)

Rotate on an arbitrary axis.

void glRotatef32i (int angle, int x, int y, int z)

Rotates the model view matrix by angle about the specified unit vector.

void glRotateX (float angle)

Rotates the current modelview matrix by angle degrees about the x axis.

void glRotateXi (int angle)

Rotates the current modelview matrix by angle about the x axis.

void glRotateY (float angle)

Rotates the current modelview matrix by angle degrees about the y axis.

void glRotateYi (int angle)

Rotates the current modelview matrix by angle about the y axis.

void glRotateZ (float angle)

Rotates the current modelview matrix by angle degrees about the z axis.

void glRotateZi (int angle)

Rotates the current modelview matrix by angle about the z axis.

void glScalef (float x, float y, float z)

multiply the current matrix by a scale matrix
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dmatrixloadmultiply

void glScalef32 (int x, int y, int z)

multiply the current matrix by a scale matrix
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dmatrixloadmultiply

void glScalev (const GLvector *v)

multiply the current matrix by a translation matrix
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dmatrixloadmultiply

void glSetOutlineColor (int id, rgb color)
Specifies an edge color for polygons.

void glSetToonTable (const uint16 *table)
Loads a toon table.

void glSetToonTableRange (int start, int end, rgb color)
Sets a range of colors on the toon table.

void glStoreMatrix (int index)
Place the current matrix into the stack at a location
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dmatrixstack

void glTexCoord2f (float s, float t)
Sets texture coordinates for following vertices
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dtextureattributes

void glTexCoord2f32 (int u, int v)
Sets texture coordinates for following vertices
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dtextureattributes

void glTexCoord2t16 (t16 u, t16 v)
Sets texture coordinates for following vertices
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dtextureattributes

int glTexImage2D (int target, int empty1, GL_TEXTURE_TYPE_ENUM type, int sizeX, int sizeY, int empty2, int param, const void *texture)
Loads a 2D texture into texture memory and sets the currently bound texture ID to the attributes specified.

void glTexParameter (int target, int param)
Set parameters for the current texture. Although named the same as its gl counterpart, it is not compatible. Effort may be made in the future to make it so.

void glTranslatef (float x, float y, float z)
multiply the current matrix by a translation matrix
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dmatrixloadmultiply

void glTranslatef32 (int x, int y, int z)
multiply the current matrix by a translation matrix
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dmatrixloadmultiply

void glTranslatev (const GLvector *v)
multiply the current matrix by a translation matrix
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dmatrixloadmultiply

void gluLookAt (float eyex, float eyey, float eyez, float lookAtx, float lookAty, float lookAtz, float upx, float upy, float upz)
Places the camera at the specified location and orientation (floating point version)

void gluLookAtf32 (int eyex, int eyey, int eyez, int lookAtx, int lookAty, int lookAtz, int upx, int upy, int upz)
Places the camera at the specified location and orientation (fixed point version)

void gluPerspective (float fovy, float aspect, float zNear, float zFar)

	Utility function which sets up the projection matrix (floating point version)
void	gluPerspectivef32 (int fovy, int aspect, int zNear, int zFar)
	Utility function which sets up the projection matrix (fixed point version)
void	gluPickMatrix (int x, int y, int width, int height, const int viewport[4])
	Utility function which generates a picking matrix for selection.
void	glVertex3f (float x, float y, float z)
	specifies a vertex location
void	glVertex3v16 (v16 x, v16 y, v16 z)
	specifies a vertex
void	glViewport (uint8 x1, uint8 y1, uint8 x2, uint8 y2)
	specify the viewport for following drawing, can be set several times per frame. GBATEK http://nocash.emubase.de/gbatek.htm#ds3ddisplaycontrol
u32	POLY_ALPHA (int n)
	used in glPolyFmt() to set the alpha level for the following polygons, set to 0 for wireframe mode
u32	POLY_ID (int n)
	used in glPolyFmt() to set the Polygon ID for the following polygons

Detailed Description

openGL (ish) interface to DS 3D hardware.

Enumeration Type Documentation

enum DISP3DCNT_ENUM

3D Display Control Register Enums
GBATEK http://nocash.emubase.de/gbatek.htm#ds3ddisplaycontrol
related functions: glEnable(), glDisable(), glInit()

Enumerator:

GL_TEXTURE_2D	enable/disable textures on the geometry engine
GL_TOON_HIGHLIGHT	enable = Highlight shading; disable = Toon shading
GL_ALPHA_TEST	whether to use the alpha threshold set in glAlphaFunc()
GL_BLEND	enable/disable alpha blending
GL_ANTIALIAS	nable/disable edge antialiasing; polygons must have different polygon IDs for the effect to work and the rear plane must be clear
GL_OUTLINE	enable/disable edge coloring; the high 3bits of the polygon ID determine the color; glSetOutlineColor() sets the available

	colors
GL_FOG_ONLY_ALPHA	enable = fade into background?; disable = don't fade?
GL_FOG	enables/disables fog
GL_COLOR_UNDERFLOW	enable = color buffer underflow, setting resets overflow flag; disable = no color buffer overflow
GL_POLY_OVERFLOW	enable = polygon/vertex buffer overflow, setting resets overflow flag; disable = no polygon/vertex buffer overflow
GL_CLEAR_BMP	rear/clear plane is in BMP mode; disable = rear/color plane is in clear mode

enum GL_GET_ENUM

Enums for reading stuff from the geometry engine
http://nocash.emubase.de/gbatek.htm#ds3diomap
related functions: glGetInt(), glGetFixed()

Enumerator:

GL_GET_VERTEX_RAM_COUNT	returns a count of vertexes currently stored in hardware vertex ram. Use glGetInt() to retrieve
GL_GET_POLYGON_RAM_COUNT	returns a count of polygons currently stored in hardware polygon ram. Use glGetInt() to retrieve
GL_GET_MATRIX_VECTOR	returns the current 3x3 directional vector matrix. Use glGetFixed() to retrieve
GL_GET_MATRIX_POSITION	returns the current 4x4 position matrix. Use glGetFixed() to retrieve
GL_GET_MATRIX_PROJECTION	returns the current 4x4 projection matrix. Use glGetFixed() to retrieve
GL_GET_MATRIX_CLIP	returns the current 4x4 clip matrix. Use glGetFixed() to retrieve
GL_GET_TEXTURE_WIDTH	returns the width of the currently bound texture. Use glGetInt() to retrieve
GL_GET_TEXTURE_HEIGHT	returns the height of the currently bound texture. Use glGetInt() to retrieve

enum GL_GLBEGIN_ENUM

Enums selecting polygon draw mode
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dpolygondefinitionsbyvertices
related functions: glBegin()

Enumerator:

GL_TRIANGLES	draw triangles with each 3 vertices defining a triangle

GL_QUADS	draw quads with each 4 vertices defining a quad
GL_TRIANGLE_STRIP	draw triangles with the first triangle defined by 3 vertices, then each additional triangle being defined by one additional vertex
GL_QUAD_STRIP	draw quads with the first quad being defined by 4 vertices, then each additional triangle being defined by 2 vertices.
GL_TRIANGLE	same as GL_TRIANGLES, old non-OpenGL version
GL_QUAD	same as GL_QUADS, old non-OpenGL version

enum GL_MATERIALS_ENUM
Enums for setting up materials
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dpolygonlightparameters
related functions: glMaterialf()

Enumerator:

GL_AMBIENT	sets the ambient color for the material. The color when the normal is not facing light
GL_DIFFUSE	sets the diffuse color for the material. The color when the normal is facing light
GL_AMBIENT_AND_DIFFUSE	sets the set ambient and diffuse colors for the material; just a two-in-one of the above.
GL_SPECULAR	sets the specular color for the material. The glossy(highlight) color of the polygon
GL_SHININESS	sets the shininess color for the material. The color that shines back to the user. I have shiny pants!
GL_EMISSION	sets the emission color for the material. bright color that is independant of normals and lights

enum GL_MATRIX_MODE_ENUM
Enums selecting matrix mode
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dmatrixloadmultiply
related functions: glMatrixMode()

Enumerator:

GL_PROJECTION	used to set the Projection Matrix
GL_POSITION	used to set the Position Matrix
GL_MODELVIEW	used to set the Modelview Matrix
GL_TEXTURE	used to set the Texture Matrix

enum GL_POLY_FORMAT_ENUM
Enums for setting how polygons will be displayed
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dpolygonattributes
related functions: glPolyFmt(), glInit(), POLY_ALPHA(), POLY_ID()

Enumerator:

POLY_FORMAT_LIGHT0	enable light number 0
POLY_FORMAT_LIGHT1	enable light number 1
POLY_FORMAT_LIGHT2	enable light number 2
POLY_FORMAT_LIGHT3	enable light number 3
POLY_MODULATION	enable modulation shading mode; this is the default
POLY_DECAL	enable decal shading
POLY_TOON_HIGHLIGHT	enable toon/highlight shading mode
POLY_SHADOW	enable shadow shading
POLY_CULL_FRONT	cull front polygons
POLY_CULL_BACK	cull rear polygons
POLY_CULL_NONE	don't cull any polygons
POLY_FOG	enable/disable fog for this polygon

enum GL_TEXTURE_PARAM_ENUM
Enums for texture parameters, such as texture wrapping and texture coord stuff
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dtextureattributes
related functions: glTexImage2d(), glTexParameter()

Enumerator:

GL_TEXTURE_WRAP_S	wrap(repeat) texture on S axis
GL_TEXTURE_WRAP_T	wrap(repeat) texture on T axis
GL_TEXTURE_FLIP_S	flip texture on S axis when wrapping
GL_TEXTURE_FLIP_T	flip texture on T axis when wrapping
GL_TEXTURE_COLOR0_TRANSPARENT	interpret color 0 as clear, same as old GL_TEXTURE_ALPHA_MASK
TEXGEN_OFF	use unmodified texcoord
TEXGEN_TEXCOORD	multiply texcoords by the texture-matrix
TEXGEN_NORMAL	set texcoords equal to normal * texture-matrix, used

	for spherical reflection mapping
TEXGEN_POSITION	set texcoords equal to vertex * texture-matrix

enum GL_TEXTURE_SIZE_ENUM
Enums for size of a texture, specify one for horizontal and one for vertical related functions: glTexImage2d(), glTexParameter()

Enumerator:

TEXTURE_SIZE_8	8 texels
TEXTURE_SIZE_16	16 texels
TEXTURE_SIZE_32	32 texels
TEXTURE_SIZE_64	64 texels
TEXTURE_SIZE_128	128 texels
TEXTURE_SIZE_256	256 texels
TEXTURE_SIZE_512	512 texels
TEXTURE_SIZE_1024	1024 texels

enum GL_TEXTURE_TYPE_ENUM
Enums for texture formats
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dtextureformats
related functions: glTexImage2d(), glTexParameter()

Enumerator:

GL_RGB32_A3	32 color palette, 3 bits of alpha
GL_RGB4	4 color palette
GL_RGB16	16 color palette
GL_RGB256	256 color palette
GL_COMPRESSED	compressed texture
GL_RGB8_A5	8 color palette, 5 bits of alpha
GL_RGBA	15 bit direct color, 1 bit of alpha
GL_RGB	15 bit direct color, manually sets alpha bit to 1

enum GLFLUSH_ENUM
Enums for glFlush()
GBATEK http://nocash.emubase.de/gbatek.htm#ds3ddisplaycontrol
related functions: glEnable(), glDisable(), glInit()

Enumerator:

GL_TRANS_MANUALSORT	enable manual sorting of translucent polygons, otherwise uses Y-sorting
GL_WBUFFERING	enable W depth buffering of vertices, otherwise uses Z depth buffering

---

Function Documentation

static void glAlphaFunc (int *alphaThreshold*) `[inline]`

set the minimum alpha value that will be used
GBATEK http://nocash.emubase.de/gbatek.htm#ds3ddisplaycontrol

Parameters:

alphaThreshold	minimum alpha value that will be used (0-15)

void glAssignColorTable (int *target*,
int *name*
)

nglAssignColorTable sets the active texture with a palette set with another texture

Parameters:

target	ignored, only here for OpenGL compatability (not really, since this isn't in OpenGL)
name	the name(int value) of the texture to load a palette from

static void glBegin (GL_GLBEGIN_ENUM *mode*) `[inline]`

Starts a polygon group.

Parameters:

mode	the draw mode for the polygon

Examples:

Graphics/3D/3D_Both_Screens/source/template.c, Graphics/3D/BoxTest/source/main.cpp, Graphics/3D/nehe/lesson02/source/nehe2.cpp, Graphics/3D/nehe/lesson03/source/nehe3.cpp, Graphics/3D/nehe/lesson04/source/nehe4.cpp, Graphics/3D/nehe/lesson05/source/nehe5.cpp, Graphics/3D/nehe/lesson06/source/nehe6.cpp, Graphics/3D/nehe/lesson07/source/nehe7.cpp, Graphics/3D/nehe/lesson08/source/nehe8.cpp, Graphics/3D/nehe/lesson09/source/nehe9.cpp, Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/3D/nehe/lesson10b/source/nehe10b.cpp, Graphics/3D/nehe/lesson11/source/nehe11.cpp, Graphics/3D/Ortho/source/main.cpp, Graphics/3D/Paletted_Cube/source/main.cpp, Graphics/3D/Simple_Quad/source/main.cpp, Graphics/3D/Simple_Tri/source/main.cpp, Graphics/3D/Textured_Cube/source/main.cpp, Graphics/3D/Textured_Quad/source/main.cpp, input/Touch_Pad/touch_look/source/main.cpp, and time/RealTimeClock/source/main.c.

void glBindTexture (int *target*,
int *name*
)

glBindTexure sets the current named texture to the active texture. Target is ignored as all DS textures are 2D

Parameters:

target	ignored, only here for OpenGL compatability
name	the name(int value) to set to the current texture

Examples:

Graphics/3D/Env_Mapping/source/main.cpp, Graphics/3D/nehe/lesson06/source/nehe6.cpp, Graphics/3D/nehe/lesson07/source/nehe7.cpp, Graphics/3D/nehe/lesson08/source/nehe8.cpp, Graphics/3D/nehe/lesson09/source/nehe9.cpp, Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/3D/nehe/lesson10b/source/nehe10b.cpp, Graphics/3D/nehe/lesson11/source/nehe11.cpp, Graphics/3D/Ortho/source/main.cpp, Graphics/3D/Paletted_Cube/source/main.cpp, Graphics/3D/Textured_Cube/source/main.cpp, Graphics/3D/Textured_Quad/source/main.cpp, and input/Touch_Pad/touch_look/source/main.cpp.

static void glClearColor (uint8 *red*,
uint8 *green*,
uint8 *blue*,
uint8 *alpha*
) `[inline]`

sets the color of the rear-plane(a.k.a Clear Color/Plane)

Parameters:

red	component (0-31)
green	component (0-31)
blue	component (0-31)
alpha	from 0(clear) to 31(opaque)

Examples:

Graphics/3D/3D_Both_Screens/source/template.c, Graphics/3D/BoxTest/source/main.cpp, Graphics/3D/Display_List/source/main.cpp, Graphics/3D/Display_List_2/source/main.cpp, Graphics/3D/Env_Mapping/source/main.cpp, Graphics/3D/nehe/lesson01/source/nehe1.cpp, Graphics/3D/nehe/lesson02/source/nehe2.cpp, Graphics/3D/nehe/lesson03/source/nehe3.cpp, Graphics/3D/nehe/lesson04/source/nehe4.cpp, Graphics/3D/nehe/lesson05/source/nehe5.cpp, Graphics/3D/nehe/lesson06/source/nehe6.cpp, Graphics/3D/nehe/lesson07/source/nehe7.cpp, Graphics/3D/nehe/lesson08/source/nehe8.cpp, Graphics/3D/nehe/lesson09/source/nehe9.cpp, Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/3D/nehe/lesson10b/source/nehe10b.cpp, Graphics/3D/nehe/lesson11/source/nehe11.cpp, Graphics/3D/Ortho/source/main.cpp, Graphics/3D/Paletted_Cube/source/main.cpp, Graphics/3D/Picking/source/main.cpp, Graphics/3D/Simple_Quad/source/main.cpp, Graphics/3D/Simple_Tri/source/main.cpp, Graphics/3D/Textured_Cube/source/main.cpp, Graphics/3D/Textured_Quad/source/main.cpp, Graphics/3D/Toon_Shading/source/main.cpp, input/Touch_Pad/touch_look/source/main.cpp, and time/RealTimeClock/source/main.c.

static void glClearDepth (fixed12d3 *depth*) `[inline]`
reset the depth buffer to this value; generally set this to GL_MAX_DEPTH.
GBATEK http://nocash.emubase.de/gbatek.htm#ds3drearplane

Parameters:

depth Something to do with the depth buffer, generally set to GL_MAX_DEPTH

Examples:

Graphics/3D/3D_Both_Screens/source/template.c, Graphics/3D/BoxTest/source/main.cpp, Graphics/3D/Display_List/source/main.cpp, Graphics/3D/Display_List_2/source/main.cpp, Graphics/3D/Env_Mapping/source/main.cpp, Graphics/3D/nehe/lesson01/source/nehe1.cpp, Graphics/3D/nehe/lesson02/source/nehe2.cpp, Graphics/3D/nehe/lesson03/source/nehe3.cpp, Graphics/3D/nehe/lesson04/source/nehe4.cpp, Graphics/3D/nehe/lesson05/source/nehe5.cpp, Graphics/3D/nehe/lesson06/source/nehe6.cpp, Graphics/3D/nehe/lesson07/source/nehe7.cpp, Graphics/3D/nehe/lesson08/source/nehe8.cpp, Graphics/3D/nehe/lesson09/source/nehe9.cpp, Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/3D/nehe/lesson10b/source/nehe10b.cpp, Graphics/3D/nehe/lesson11/source/nehe11.cpp, Graphics/3D/Ortho/source/main.cpp, Graphics/3D/Paletted_Cube/source/main.cpp, Graphics/3D/Picking/source/main.cpp, Graphics/3D/Simple_Quad/source/main.cpp, Graphics/3D/Simple_Tri/source/main.cpp, Graphics/3D/Textured_Cube/source/main.cpp, Graphics/3D/Textured_Quad/source/main.cpp, Graphics/3D/Toon_Shading/source/main.cpp, input/Touch_Pad/touch_look/source/main.cpp, and time/RealTimeClock/source/main.c.

static void glClearPolyID (uint8 *ID*) `[inline]`
sets the polygon ID of the rear-plane(a.k.a. Clear/Color Plane), useful for antialiasing and edge coloring

Parameters:

ID the polygon ID to give the rear-plane

Examples:

Graphics/3D/3D_Both_Screens/source/template.c, Graphics/3D/BoxTest/source/main.cpp, Graphics/3D/Display_List/source/main.cpp, Graphics/3D/Display_List_2/source/main.cpp, Graphics/3D/Env_Mapping/source/main.cpp, Graphics/3D/nehe/lesson01/source/nehe1.cpp, Graphics/3D/nehe/lesson02/source/nehe2.cpp, Graphics/3D/nehe/lesson03/source/nehe3.cpp, Graphics/3D/nehe/lesson04/source/nehe4.cpp, Graphics/3D/nehe/lesson05/source/nehe5.cpp, Graphics/3D/nehe/lesson06/source/nehe6.cpp, Graphics/3D/nehe/lesson07/source/nehe7.cpp, Graphics/3D/nehe/lesson08/source/nehe8.cpp, Graphics/3D/nehe/lesson09/source/nehe9.cpp, Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/3D/nehe/lesson10b/source/nehe10b.cpp, Graphics/3D/nehe/lesson11/source/nehe11.cpp, Graphics/3D/Ortho/source/main.cpp, Graphics/3D/Paletted_Cube/source/main.cpp, Graphics/3D/Picking/source/main.cpp, Graphics/3D/Simple_Quad/source/main.cpp, Graphics/3D/Simple_Tri/source/main.cpp, Graphics/3D/Textured_Cube/source/main.cpp, Graphics/3D/Textured_Quad/source/main.cpp, and Graphics/3D/Toon_Shading/source/main.cpp.

static void glColor (rgb *color*) `[inline]`

Set the color for following vertices.

Parameters:

color	the 15bit color value

Examples:

Graphics/3D/nehe/lesson10/source/nehe10.cpp.

```
static void glColor3b ( uint8 red,
                        uint8 green,
                        uint8 blue
                      )       [inline]
```

Set the color for following vertices.

Parameters:

red	the red component (0-255) Bottom 3 bits ignored
green	the green component (0-255) Bottom 3 bits ignored
blue	the blue component (0-255) Bottom 3 bits ignored

Examples:

Graphics/3D/3D_Both_Screens/source/template.c, Graphics/3D/nehe/lesson09/source/nehe9.cpp, Graphics/3D/nehe/lesson11/source/nehe11.cpp, Graphics/3D/Simple_Quad/source/main.cpp, and Graphics/3D/Simple_Tri/source/main.cpp.

```
static void glColor3f ( float r,
                        float g,
                        float b
                      )       [inline]
```

specify a color for following vertices

Warning:

FLOAT VERSION!!!! please use glColor3b()

Parameters:

r	the red component of the color
g	the green component of the color
b	the blue component of the color

Examples:

Graphics/3D/BoxTest/source/main.cpp, Graphics/3D/nehe/lesson01/source/nehe1.cpp, Graphics/3D/nehe/lesson02/source/nehe2.cpp, Graphics/3D/nehe/lesson03/source/nehe3.cpp, Graphics/3D/nehe/lesson04/source/nehe4.cpp, Graphics/3D/nehe/lesson05/source/nehe5.cpp, Graphics/3D/nehe/lesson06/source/nehe6.cpp, Graphics/3D/nehe/lesson07/source/nehe7.cpp, Graphics/3D/nehe/lesson08/source/nehe8.cpp, Graphics/3D/nehe/lesson09/source/nehe9.cpp, Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/3D/nehe/lesson10b/source/nehe10b.cpp, Graphics/3D/Ortho/source/main.cpp, Graphics/3D/Paletted_Cube/source/main.cpp, input/Touch_Pad/touch_look/source/main.cpp, and time/RealTimeClock/source/main.c.

```
void glColorTableEXT ( int              target,
                       int              empty1,
                       uint16           width,
                       int              empty2,
                       int              empty3,
                       const uint16 *   table
                     )
```

nglColorTableEXT loads a 15-bit color format palette into palette memory, and sets it to the currently bound texture (can be used to remove also)

Parameters:

target	ignored, only here for OpenGL compatability
empty1	ignored, only here for OpenGL compatability
width	the length of the palette (if 0, then palette is removed from currently bound texture)
empty2	ignored, only here for OpenGL compatability
empty3	ignored, only here for OpenGL compatability
table	pointer to the palette data to load (if NULL, then palette is removed from currently bound texture)

Examples:

Graphics/3D/Paletted_Cube/source/main.cpp.

```
static void glCutoffDepth ( fixed12d3 wVal ) [inline]
```

Stop the drawing of polygons that are a certain distance from the camera.
GBATEK http://nocash.emubase.de/gbatek.htm#ds3ddisplaycontrol

Parameters:

wVal	polygons that are beyond this W-value(distance from camera) will not be drawn; 15bit value.

int glDeleteTextures (int *n*,
int * *names*
)

Deletes the specified number of textures (and associated palettes)

Parameters:

n	the number of textures to delete
names	pointer to the names array to empty

Examples:

Graphics/3D/Paletted_Cube/source/main.cpp.

static void glDisable (int *bits*) `[inline]`

Disables various gl states (blend, alpha test, etc..)

Parameters:

bits bit mask of desired attributes, attributes are enumerated in DISP3DCNT_ENUM

static void glEnable (int *bits*) `[inline]`

Enables various gl states (blend, alpha test, etc..)

Parameters:

bits bit mask of desired attributes, attributes are enumerated in DISP3DCNT_ENUM

Examples:

Graphics/3D/3D_Both_Screens/source/template.c,
Graphics/3D/BoxTest/source/main.cpp, Graphics/3D/Display_List/source/main.cpp,
Graphics/3D/Display_List_2/source/main.cpp,
Graphics/3D/Env_Mapping/source/main.cpp,
Graphics/3D/nehe/lesson01/source/nehe1.cpp,
Graphics/3D/nehe/lesson02/source/nehe2.cpp,
Graphics/3D/nehe/lesson03/source/nehe3.cpp,
Graphics/3D/nehe/lesson04/source/nehe4.cpp,
Graphics/3D/nehe/lesson05/source/nehe5.cpp,
Graphics/3D/nehe/lesson06/source/nehe6.cpp,
Graphics/3D/nehe/lesson07/source/nehe7.cpp,
Graphics/3D/nehe/lesson08/source/nehe8.cpp,
Graphics/3D/nehe/lesson09/source/nehe9.cpp,
Graphics/3D/nehe/lesson10/source/nehe10.cpp,
Graphics/3D/nehe/lesson10b/source/nehe10b.cpp,
Graphics/3D/nehe/lesson11/source/nehe11.cpp, Graphics/3D/Ortho/source/main.cpp,
Graphics/3D/Paletted_Cube/source/main.cpp, Graphics/3D/Picking/source/main.cpp,
Graphics/3D/Simple_Quad/source/main.cpp,
Graphics/3D/Simple_Tri/source/main.cpp,
Graphics/3D/Textured_Cube/source/main.cpp, Graphics/3D/Textured_Quad/source/main.cpp, Graphics/3D/Toon_Shading/source/main.cpp, and
input/Touch_Pad/touch_look/source/main.cpp.

static void glFlush (u32 *mode*) `[inline]`

Waits for a Vblank and swaps the buffers(like swiWaitForVBlank), but lets you specify some 3D options

GBATEK http://nocash.emubase.de/gbatek.htm#ds3ddisplaycontrol

Parameters:

mode flags from GLFLUSH_ENUM for enabling Y-sorting of translucent polygons and W-Buffering of all vertices

Examples:
Graphics/3D/3D_Both_Screens/source/template.c, Graphics/3D/BoxTest/source/main.cpp, Graphics/3D/Display_List/source/main.cpp, Graphics/3D/Display_List_2/source/main.cpp, Graphics/3D/Env_Mapping/source/main.cpp, Graphics/3D/nehe/lesson01/source/nehe1.cpp, Graphics/3D/nehe/lesson02/source/nehe2.cpp, Graphics/3D/nehe/lesson03/source/nehe3.cpp, Graphics/3D/nehe/lesson04/source/nehe4.cpp, Graphics/3D/nehe/lesson05/source/nehe5.cpp, Graphics/3D/nehe/lesson06/source/nehe6.cpp, Graphics/3D/nehe/lesson07/source/nehe7.cpp, Graphics/3D/nehe/lesson08/source/nehe8.cpp, Graphics/3D/nehe/lesson09/source/nehe9.cpp, Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/3D/nehe/lesson10b/source/nehe10b.cpp, Graphics/3D/nehe/lesson11/source/nehe11.cpp, Graphics/3D/Ortho/source/main.cpp, Graphics/3D/Paletted_Cube/source/main.cpp, Graphics/3D/Picking/source/main.cpp, Graphics/3D/Simple_Quad/source/main.cpp, Graphics/3D/Simple_Tri/source/main.cpp, Graphics/3D/Textured_Cube/source/main.cpp, Graphics/3D/Textured_Quad/source/main.cpp, Graphics/3D/Toon_Shading/source/main.cpp, input/Touch_Pad/touch_look/source/main.cpp, and time/RealTimeClock/source/main.c.

```
static void glFogColor  ( uint8  red,
                          uint8  green,
                          uint8  blue,
                          uint8  alpha
                        )        [inline]
```

sets the fog color

Parameters:

red	component (0-31)
green	component (0-31)
blue	component (0-31)
alpha	from 0(clear) to 31(opaque)

Examples:
Graphics/3D/nehe/lesson10/source/nehe10.cpp.

```
static void glFogDensity  ( int  index,
                            int  density
                          )     [inline]
```

sets the fog density at a given index

Parameters:

index	fog table index to operate on (0 to 31)
density	fog density from 0 (none) to 127 (opaque)

Examples:

Graphics/3D/nehe/lesson10/source/nehe10.cpp.

static void glFogOffset (int *shift*) `[inline]`

Sets the FOG_OFFSET value.

Parameters:

shift	FOG_OFFSET value; fogging begins at this depth with a density of FOG_TABLE[0]

Examples:

Graphics/3D/nehe/lesson10/source/nehe10.cpp.

static void glFogShift (int *shift*) `[inline]`

Sets the FOG_SHIFT value.

Parameters:

shift	FOG_SHIFT value; each entry of the fog table covers 0x400 >> FOG_SHIFT depth values

Examples:

Graphics/3D/nehe/lesson10/source/nehe10.cpp.

static void glFrustum (float *left*,
float *right*,
float *bottom*,
float *top*,
float *near*,
float *far*
) `[inline]`

Specifies the viewing frustum for the projection matrix (floating point version)

Warning:

FLOAT VERSION!!!! please use glFrustumf32()

Parameters:

left	left right top and bottom describe a rectangle located at the near clipping plane
right	left right top and bottom describe a rectangle located at the near clipping plane
top	left right top and bottom describe a rectangle located at the near clipping plane
bottom	left right top and bottom describe a rectangle located at the near clipping plane
near	Location of a the near clipping plane (parallel to viewing window)
far	Location of a the far clipping plane (parallel to viewing window)

<convert float to f32

<convert float to f32

<convert float to f32

<convert float to f32

<convert float to f32

<convert float to f32

```
static void glFrustumf32 ( int left,
                           int right,
                           int bottom,
                           int top,
                           int near,
                           int far
                         )    [inline]
```

Specifies the viewing frustum for the projection matrix (fixed point version)

Parameters:

left	left right top and bottom describe a rectangle located at the near clipping plane
right	left right top and bottom describe a rectangle located at the near clipping plane
top	left right top and bottom describe a rectangle located at the near clipping plane
bottom	left right top and bottom describe a rectangle located at the near clipping plane
near	Location of a the near clipping plane (parallel to viewing window)
far	Location of a the far clipping plane (parallel to viewing window)

<convert float to f32

```
int glGenTextures ( int   n,
                    int * names
                  )
```

Creates room for the specified number of textures.

Parameters:

n	the number of textures to generate
names	pointer to the names array to fill

Examples:

Graphics/3D/Env_Mapping/source/main.cpp, Graphics/3D/nehe/lesson06/source/nehe6.cpp, Graphics/3D/nehe/lesson07/source/nehe7.cpp, Graphics/3D/nehe/lesson08/source/nehe8.cpp, Graphics/3D/nehe/lesson09/source/nehe9.cpp, Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/3D/nehe/lesson10b/source/nehe10b.cpp, Graphics/3D/nehe/lesson11/source/nehe11.cpp, Graphics/3D/Ortho/source/main.cpp, Graphics/3D/Paletted_Cube/source/main.cpp, Graphics/3D/Textured_Cube/source/main.cpp, Graphics/3D/Textured_Quad/source/main.cpp, and input/Touch_Pad/touch_look/source/main.cpp.

```
static void glGetFixed ( const GL_GET_ENUM param,
                         int *             f
                       )                   [inline]
```

Grabs fixed format of state variables

OpenGL's modelview matrix is handled on the DS with two matrices. The combination of

the DS's position matrix and directional vector matrix hold the data that is in OpenGL's one modelview matrix. (a.k.a. modelview = postion and vector)
http://nocash.emubase.de/gbatek.htm#ds3diomap

Parameters:

param	The state variable to retrieve
f	pointer with room to hold the requested data

```
static void glGetInt ( GL_GET_ENUM  param,
                       int *        i
                     )              [inline]
```

Grabs integer state variables from openGL.

Parameters:

param	The state variable to retrieve
i	pointer with room to hold the requested data

Examples:

Graphics/3D/BoxTest/source/main.cpp.

```
void* glGetTexturePointer ( int name )
```

returns the address alocated to the texure named by name

Parameters:

name	the name of the texture to get a pointer to

```
static void glLight ( int  id,
                      rgb  color,
                      v10  x,
                      v10  y,
                      v10  z
                    )      [inline]
```

set a light up. Only parallel light sources are supported on the DS
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dpolygonlightparameters

Parameters:

id	the number of the light to setup
color	the color of the light
x	the x component of the lights directional vector. Direction must be normalized
y	the y component of the lights directional vector. Direction must be normalized
z	the z component of the lights directional vector. Direction must be normalized

Examples:

Graphics/3D/Display_List_2/source/main.cpp, Graphics/3D/nehe/lesson06/source/nehe6.cpp, Graphics/3D/nehe/lesson07/source/nehe7.cpp, Graphics/3D/nehe/lesson08/source/nehe8.cpp, Graphics/3D/nehe/lesson09/source/nehe9.cpp, Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/3D/nehe/lesson10b/source/nehe10b.cpp, Graphics/3D/Ortho/source/main.cpp, Graphics/3D/Paletted_Cube/source/main.cpp, Graphics/3D/Picking/source/main.cpp, Graphics/3D/Textured_Cube/source/main.cpp, Graphics/3D/Toon_Shading/source/main.cpp, and input/Touch_Pad/touch_look/source/main.cpp.

static void glLoadMatrix4x3 (const m4x3 * *m*) [inline]

Loads a 4x3 matrix into the current matrix.

Parameters:

m pointer to a 4x4 matrix

static void glLoadMatrix4x4 (const m4x4 * *m*) [inline]

Loads a 4x4 matrix into the current matrix.

Parameters:

m pointer to a 4x4 matrix

void glMaterialf (GL_MATERIALS_ENUM *mode*,
rgb *color*
)

specify the material properties to be used in rendering lit polygons

Parameters:

mode	which material property to change
color	the color to set for that material property

Examples:

Graphics/3D/Env_Mapping/source/main.cpp, Graphics/3D/nehe/lesson06/source/nehe6.cpp, Graphics/3D/nehe/lesson07/source/nehe7.cpp, Graphics/3D/nehe/lesson08/source/nehe8.cpp, Graphics/3D/nehe/lesson09/source/nehe9.cpp, Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/3D/nehe/lesson10b/source/nehe10b.cpp, Graphics/3D/nehe/lesson11/source/nehe11.cpp, Graphics/3D/Ortho/source/main.cpp, Graphics/3D/Paletted_Cube/source/main.cpp, Graphics/3D/Textured_Cube/source/main.cpp, Graphics/3D/Textured_Quad/source/main.cpp, Graphics/3D/Toon_Shading/source/main.cpp, and input/Touch_Pad/touch_look/source/main.cpp.

static void glMatrixMode (GL_MATRIX_MODE_ENUM *mode*) [inline]

change the current matrix mode

GBATEK http://nocash.emubase.de/gbatek.htm#ds3dmatrixloadmultiply

Parameters:

mode the mode for the matrix

Examples:

Graphics/3D/3D_Both_Screens/source/template.c, Graphics/3D/BoxTest/source/main.cpp, Graphics/3D/Display_List/source/main.cpp, Graphics/3D/Display_List_2/source/main.cpp, Graphics/3D/Env_Mapping/source/main.cpp, Graphics/3D/nehe/lesson01/source/nehe1.cpp, Graphics/3D/nehe/lesson02/source/nehe2.cpp, Graphics/3D/nehe/lesson03/source/nehe3.cpp, Graphics/3D/nehe/lesson04/source/nehe4.cpp, Graphics/3D/nehe/lesson05/source/nehe5.cpp, Graphics/3D/nehe/lesson06/source/nehe6.cpp, Graphics/3D/nehe/lesson07/source/nehe7.cpp, Graphics/3D/nehe/lesson08/source/nehe8.cpp, Graphics/3D/nehe/lesson09/source/nehe9.cpp, Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/3D/nehe/lesson10b/source/nehe10b.cpp, Graphics/3D/nehe/lesson11/source/nehe11.cpp, Graphics/3D/Ortho/source/main.cpp, Graphics/3D/Paletted_Cube/source/main.cpp, Graphics/3D/Picking/source/main.cpp, Graphics/3D/Simple_Quad/source/main.cpp, Graphics/3D/Simple_Tri/source/main.cpp, Graphics/3D/Textured_Cube/source/main.cpp, Graphics/3D/Textured_Quad/source/main.cpp, Graphics/3D/Toon_Shading/source/main.cpp, input/Touch_Pad/touch_look/source/main.cpp, and time/RealTimeClock/source/main.c.

static void glMultMatrix3x3 (const m3x3 * *m*) `[inline]`
multiplies the current matrix by m

Parameters:
m pointer to a 3x3 matrix

static void glMultMatrix4x3 (const m4x3 * *m*) `[inline]`
multiplies the current matrix by

Parameters:
m pointer to a 4x3 matrix

static void glMultMatrix4x4 (const m4x4 * *m*) `[inline]`
Multiplies the current matrix by m.

Parameters:
m pointer to a 4x4 matrix

static void glNormal (u32 *normal*) `[inline]`
the normal to use for following vertices
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dpolygonlightparameters

Warning:
The nature of the format means that you can't represent the following normals exactly (0,0,1), (0,1,0), or (1,0,0)

Parameters:
normal the packed normal(3 * 10bit x, y, z)

Examples:
Graphics/3D/nehe/lesson10b/source/nehe10b.cpp, Graphics/3D/Paletted_Cube/source/main.cpp, Graphics/3D/Textured_Cube/source/main.cpp, Graphics/3D/Textured_Quad/source/main.cpp, and input/Touch_Pad/touch_look/source/main.cpp.

static void glNormal3f (float *x*,
float *y*,
float *z*
) `[inline]`
the normal to use for following vertices
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dpolygonlightparameters

Warning:
FLOAT VERSION!!!! please use glNormal()

Parameters:

x	x component of the normal, vector must be normalized
y	y component of the normal, vector must be normalized
z	z component of the normal, vector must be normalized

<convert float to v10

<convert float to v10

<convert float to v10

<Pack 3 v10 normals into a 32bit value

Examples:

Graphics/3D/nehe/lesson07/source/nehe7.cpp, Graphics/3D/nehe/lesson08/source/nehe8.cpp, and Graphics/3D/nehe/lesson10/source/nehe10.cpp.

```
static void glOrtho ( float left,
                      float right,
                      float bottom,
                      float top,
                      float zNear,
                      float zFar
                    )       [inline]
```

Multiplies the current matrix into ortho graphic mode.

Warning:

FLOAT VERSION!!!! please use glOrthof32()

Parameters:

left	left vertical clipping plane
right	right vertical clipping plane
bottom	bottom vertical clipping plane
top	top vertical clipping plane
zNear	near clipping plane
zFar	far clipping plane

<convert float to f32

<convert float to f32

<convert float to f32

<convert float to f32

<convert float to f32

<convert float to f32

Examples:

Graphics/3D/BoxTest/source/main.cpp, and Graphics/3D/Ortho/source/main.cpp.

```
static void glOrthof32 ( int left,
                         int right,
                         int bottom,
```

int *top*,
int *zNear*,
int *zFar*
) `[inline]`

Multiplies the current matrix into ortho graphic mode.

Parameters:

left	left vertical clipping plane
right	right vertical clipping plane
bottom	bottom vertical clipping plane
top	top vertical clipping plane
zNear	near clipping plane
zFar	far clipping plane

<convert int to f32

<convert int to f32

<convert int to f32

<convert float to f32

static void glPolyFmt (u32 *params*) `[inline]`

Set the parameters for polygons rendered on the current frame
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dpolygonattributes

Parameters:

params	the paramters to set for the polygons for the current frame. valid paramters are enumerated in GL_POLY_FORMAT_ENUM and in the functions POLY_ALPHA() and POLY_ID()

Examples:

Graphics/3D/3D_Both_Screens/source/template.c, Graphics/3D/BoxTest/source/main.cpp, Graphics/3D/Display_List/source/main.cpp, Graphics/3D/Display_List_2/source/main.cpp, Graphics/3D/Env_Mapping/source/main.cpp, Graphics/3D/nehe/lesson01/source/nehe1.cpp, Graphics/3D/nehe/lesson02/source/nehe2.cpp, Graphics/3D/nehe/lesson03/source/nehe3.cpp, Graphics/3D/nehe/lesson04/source/nehe4.cpp, Graphics/3D/nehe/lesson05/source/nehe5.cpp, Graphics/3D/nehe/lesson06/source/nehe6.cpp, Graphics/3D/nehe/lesson07/source/nehe7.cpp, Graphics/3D/nehe/lesson08/source/nehe8.cpp, Graphics/3D/nehe/lesson09/source/nehe9.cpp, Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/3D/nehe/lesson10b/source/nehe10b.cpp, Graphics/3D/nehe/lesson11/source/nehe11.cpp, Graphics/3D/Ortho/source/main.cpp, Graphics/3D/Paletted_Cube/source/main.cpp, Graphics/3D/Picking/source/main.cpp, Graphics/3D/Simple_Quad/source/main.cpp, Graphics/3D/Simple_Tri/source/main.cpp, Graphics/3D/Textured_Cube/source/main.cpp, Graphics/3D/Textured_Quad/source/main.cpp, Graphics/3D/Toon_Shading/source/main.cpp, input/Touch_Pad/touch_look/source/main.cpp, and time/RealTimeClock/source/main.c.

static void glPopMatrix (int *num*) `[inline]`

Pops num matrices off the stack

GBATEK http://nocash.emubase.de/gbatek.htm#ds3dmatrixstack

Parameters:

num	the number to pop down the stack

Examples:

Graphics/3D/3D_Both_Screens/source/template.c, Graphics/3D/Display_List/source/main.cpp, Graphics/3D/Display_List_2/source/main.cpp, Graphics/3D/nehe/lesson01/source/nehe1.cpp, Graphics/3D/nehe/lesson02/source/nehe2.cpp, Graphics/3D/nehe/lesson05/source/nehe5.cpp, Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/3D/Ortho/source/main.cpp, Graphics/3D/Paletted_Cube/source/main.cpp, Graphics/3D/Picking/source/main.cpp, Graphics/3D/Simple_Quad/source/main.cpp, Graphics/3D/Simple_Tri/source/main.cpp, Graphics/3D/Textured_Cube/source/main.cpp, Graphics/3D/Textured_Quad/source/main.cpp, Graphics/3D/Toon_Shading/source/main.cpp, input/Touch_Pad/touch_look/source/main.cpp, and time/RealTimeClock/source/main.c.

static void glRestoreMatrix (int *index*) `[inline]`

Restores the current matrix from a location in the stack

GBATEK http://nocash.emubase.de/gbatek.htm#ds3dmatrixstack

Parameters:

index	the place in the stack to restore to

static void glRotatef (float *angle*,
float *x*,
float *y*,
float *z*
) `[inline]`

Rotate about an arbitrary axis.

Warning:

FLOAT VERSION!!!! please use glRotatef32i()

Parameters:

angle	the angle to rotate by
x	the x component of the axis to rotate on
y	the y component of the axis to rotate on
z	the z component of the axis to rotate on

<convert float to f32

<convert float to f32

<convert float to f32

Examples:

Graphics/3D/nehe/lesson04/source/nehe4.cpp, Graphics/3D/nehe/lesson05/source/nehe5.cpp, Graphics/3D/nehe/lesson06/source/nehe6.cpp, Graphics/3D/nehe/lesson07/source/nehe7.cpp, Graphics/3D/nehe/lesson08/source/nehe8.cpp, Graphics/3D/nehe/lesson09/source/nehe9.cpp, Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/3D/nehe/lesson11/source/nehe11.cpp, and Graphics/3D/Ortho/source/main.cpp.

```
static void glRotatef32 ( float angle,
                          int   x,
                          int   y,
                          int   z
                        )       [inline]
```

Rotate on an arbitrary axis.

Warning:

FLOAT VERSION!!!! please use glRotatef32i()

Parameters:

angle	the angle to rotate by
x	the x component of the axis to rotate on
y	the y component of the axis to rotate on
z	the z component of the axis to rotate on

```
void glRotatef32i ( int angle,
                    int x,
                    int y,
                    int z
                  )
```

Rotates the model view matrix by angle about the specified unit vector.

Parameters:

angle	The angle to rotate by
x	X component of the unit vector axis.
y	Y component of the unit vector axis.
z	Z component of the unit vector axis.

Examples:

Graphics/3D/3D_Both_Screens/source/template.c, Graphics/3D/nehe/lesson10b/source/nehe10b.cpp, and input/Touch_Pad/touch_look/source/main.cpp.

```
static void glRotateX ( float angle ) [inline]
```

Rotates the current modelview matrix by angle degrees about the x axis.

Warning:

FLOAT VERSION!!!! please use glRotateXi()

Parameters:

angle	The angle to rotate by

Examples:

Graphics/3D/BoxTest/source/main.cpp, Graphics/3D/Display_List/source/main.cpp, Graphics/3D/Display_List_2/source/main.cpp, Graphics/3D/Paletted_Cube/source/main.cpp,

Graphics/3D/Simple_Quad/source/main.cpp,
Graphics/3D/Simple_Tri/source/main.cpp,
Graphics/3D/Textured_Cube/source/main.cpp, and
Graphics/3D/Textured_Quad/source/main.cpp.

static void glRotateXi (int *angle*) `[inline]`

Rotates the current modelview matrix by angle about the x axis.

Parameters:

angle The angle to rotate by (angle is -32768 to 32767)

<convert int to f32

Examples:

Graphics/3D/Env_Mapping/source/main.cpp, Graphics/3D/Picking/source/main.cpp, and Graphics/3D/Toon_Shading/source/main.cpp.

static void glRotateY (float *angle*) `[inline]`

Rotates the current modelview matrix by angle degrees about the y axis.

Warning:

FLOAT VERSION!!!! please use glRotateYi()

Parameters:

angle The angle to rotate by

Examples:

Graphics/3D/BoxTest/source/main.cpp, Graphics/3D/Display_List/source/main.cpp,
Graphics/3D/Display_List_2/source/main.cpp,
Graphics/3D/Paletted_Cube/source/main.cpp,
Graphics/3D/Simple_Quad/source/main.cpp,
Graphics/3D/Simple_Tri/source/main.cpp,
Graphics/3D/Textured_Cube/source/main.cpp, and
Graphics/3D/Textured_Quad/source/main.cpp.

static void glRotateYi (int *angle*) `[inline]`

Rotates the current modelview matrix by angle about the y axis.

Parameters:

angle The angle to rotate by (angle is -32768 to 32767)

<convert int to f32

Examples:

Graphics/3D/Env_Mapping/source/main.cpp, Graphics/3D/Picking/source/main.cpp, and Graphics/3D/Toon_Shading/source/main.cpp.

static void glRotateZ (float *angle*) `[inline]`

Rotates the current modelview matrix by angle degrees about the z axis.

Warning:

FLOAT VERSION!!!! please use glRotateZi()

Parameters:

angle The angle to rotate by

Examples:

time/RealTimeClock/source/main.c.

static void glRotateZi (int *angle*) `[inline]`

Rotates the current modelview matrix by angle about the z axis.

Parameters:

angle The angle to rotate by (angle is -32768 to 32767)

<convert int to f32

static void glScalef (float *x*,
float *y*,
float *z*
) `[inline]`

multiply the current matrix by a scale matrix
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dmatrixloadmultiply

Warning:

FLOAT VERSION!!!! please use glScalev() or glScalef32()

Parameters:

x	scaling on the x axis
y	scaling on the y axis
z	scaling on the z axis

<convert float to f32

<convert float to f32

<convert float to f32

Examples:

Graphics/3D/nehe/lesson10/source/nehe10.cpp.

static void glScalef32 (int *x*,
int *y*,
int *z*
) `[inline]`

multiply the current matrix by a scale matrix
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dmatrixloadmultiply

Parameters:

x	scaling on the x axis
y	scaling on the y axis
z	scaling on the z axis

static void glScalev (const GLvector * *v*) `[inline]`

multiply the current matrix by a translation matrix
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dmatrixloadmultiply

Parameters:

v the vector to translate by

Examples:

Graphics/3D/Env_Mapping/source/main.cpp.

static void glSetOutlineColor (int *id*,
rgb *color*
) `[inline]`

Specifies an edge color for polygons.

Parameters:

id	which outline color to set (0-7)
color	the 15bit color to set

Examples:

Graphics/3D/Picking/source/main.cpp.

static void glSetToonTable (const uint16 * *table*) `[inline]`

Loads a toon table.

Parameters:

table	pointer to the 32 color palette to load into the toon table

static void glSetToonTableRange (int *start*,
int *end*,
rgb *color*
) `[inline]`

Sets a range of colors on the toon table.

Parameters:

start	the start of the range
end	the end of the range
color	the color to set for that range

Examples:

Graphics/3D/Toon_Shading/source/main.cpp.

static void glStoreMatrix (int *index*) `[inline]`

Place the current matrix into the stack at a location
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dmatrixstack

Parameters:

index	the place in the stack to put the current matrix

static void glTexCoord2f (float *s*,
float *t*
) `[inline]`

Sets texture coordinates for following vertices
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dtextureattributes

Warning:

FLOAT VERSION!!!! please use glTexCoord2t16()

Parameters:

s	S(a.k.a. U) texture coordinate (0.0 - 1.0)
t	T(a.k.a. V) texture coordinate (0.0 - 1.0)

<convert float to t16

<convert float to t16

Examples:

Graphics/3D/nehe/lesson06/source/nehe6.cpp,
Graphics/3D/nehe/lesson07/source/nehe7.cpp,
Graphics/3D/nehe/lesson08/source/nehe8.cpp,
Graphics/3D/nehe/lesson09/source/nehe9.cpp,
Graphics/3D/nehe/lesson10/source/nehe10.cpp, and
Graphics/3D/Ortho/source/main.cpp.

```
void glTexCoord2f32  ( int  u,
                       int  v
                     )
```

Sets texture coordinates for following vertices
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dtextureattributes

Parameters:

u	U(a.k.a. S) texture coordinate (0.0 - 1.0)
v	V(a.k.a. T) texture coordinate (0.0 - 1.0)

```
static void glTexCoord2t16  ( t16  u,
                              t16  v
                            )      [inline]
```

Sets texture coordinates for following vertices
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dtextureattributes

Parameters:

u	U(a.k.a. S) texture coordinate in texels
v	V(a.k.a. T) texture coordinate in texels

<Pack 2 t16 texture coordinate values into a 32bit value

Examples:

Graphics/3D/nehe/lesson10b/source/nehe10b.cpp,
Graphics/3D/nehe/lesson11/source/nehe11.cpp, and
input/Touch_Pad/touch_look/source/main.cpp.

```
int glTexImage2D  ( int                    target,
                    int                    empty1,
                    GL_TEXTURE_TYPE_ENUM   type,
                    int                    sizeX,
                    int                    sizeY,
                    int                    empty2,
                    int                    param,
                    const void *           texture
                  )
```

Loads a 2D texture into texture memory and sets the currently bound texture ID to the attributes specified.

Parameters:

target	not used, just here for OpenGL compatibility
empty1	not used, just here for OpenGL compatibility
type	The format of the texture

sizeX	the horizontal size of the texture; valid sizes are enumerated in GL_TEXTURE_TYPE_ENUM
sizeY	the vertical size of the texture; valid sizes are enumerated in GL_TEXTURE_TYPE_ENUM
empty2	not used, just here for OpenGL compatibility
param	parameters for the texture
texture	pointer to the texture data to load

Examples:

Graphics/3D/Env_Mapping/source/main.cpp, Graphics/3D/nehe/lesson06/source/nehe6.cpp, Graphics/3D/nehe/lesson07/source/nehe7.cpp, Graphics/3D/nehe/lesson08/source/nehe8.cpp, Graphics/3D/nehe/lesson09/source/nehe9.cpp, Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/3D/nehe/lesson10b/source/nehe10b.cpp, Graphics/3D/nehe/lesson11/source/nehe11.cpp, Graphics/3D/Ortho/source/main.cpp, Graphics/3D/Paletted_Cube/source/main.cpp, Graphics/3D/Textured_Cube/source/main.cpp, Graphics/3D/Textured_Quad/source/main.cpp, and input/Touch_Pad/touch_look/source/main.cpp.

```
void glTexParameter  ( int  target,
                       int  param
                     )
```

Set parameters for the current texture. Although named the same as its gl counterpart, it is not compatible. Effort may be made in the future to make it so.

Parameters:

target	not used, just here for OpenGL compatibility
param	paramaters for the texture

```
static void glTranslatef  ( float  x,
                            float  y,
                            float  z
                          )        [inline]
```

multiply the current matrix by a translation matrix
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dmatrixloadmultiply

Warning:

FLOAT VERSION!!!! please use glTranslatef32()

Parameters:

x	translation on the x axis
y	translation on the y axis
z	translation on the z axis

<convert float to f32

<convert float to f32

<convert float to f32

Examples:

Graphics/3D/3D_Both_Screens/source/template.c, Graphics/3D/BoxTest/source/main.cpp, Graphics/3D/nehe/lesson02/source/nehe2.cpp, Graphics/3D/nehe/lesson03/source/nehe3.cpp, Graphics/3D/nehe/lesson04/source/nehe4.cpp, Graphics/3D/nehe/lesson05/source/nehe5.cpp, Graphics/3D/nehe/lesson06/source/nehe6.cpp, Graphics/3D/nehe/lesson07/source/nehe7.cpp, Graphics/3D/nehe/lesson08/source/nehe8.cpp, Graphics/3D/nehe/lesson09/source/nehe9.cpp, Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/3D/nehe/lesson11/source/nehe11.cpp, Graphics/3D/Ortho/source/main.cpp, and time/RealTimeClock/source/main.c.

static void glTranslatef32 (int *x*,
int *y*,
int *z*
) `[inline]`

multiply the current matrix by a translation matrix
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dmatrixloadmultiply

Parameters:

x	translation on the x axis
y	translation on the y axis
z	translation on the z axis

Examples:

Graphics/3D/Display_List/source/main.cpp, Graphics/3D/Env_Mapping/source/main.cpp, Graphics/3D/nehe/lesson10b/source/nehe10b.cpp, Graphics/3D/Picking/source/main.cpp, Graphics/3D/Simple_Quad/source/main.cpp, Graphics/3D/Simple_Tri/source/main.cpp, Graphics/3D/Textured_Cube/source/main.cpp, Graphics/3D/Textured_Quad/source/main.cpp, and input/Touch_Pad/touch_look/source/main.cpp.

static void glTranslatev (const GLvector * *v*) `[inline]`

multiply the current matrix by a translation matrix
GBATEK http://nocash.emubase.de/gbatek.htm#ds3dmatrixloadmultiply

Parameters:

v	the vector to translate by

static void gluLookAt (float *eyex*,
float *eyey*,
float *eyez*,
float *lookAtx*,
float *lookAty*,
float *lookAtz*,
float *upx*,
float *upy*,
float *upz*
) `[inline]`

Places the camera at the specified location and orientation (floating point version)

Warning:

FLOAT VERSION!!!! please use gluLookAtf32()

Parameters:

eyex	(eyex, eyey, eyez) Location of the camera.
eyey	(eyex, eyey, eyez) Location of the camera.
eyez	(eyex, eyey, eyez) Location of the camera.
lookAtx	(lookAtx, lookAty, lookAtz) Where the camera is looking.
lookAty	(lookAtx, lookAty, lookAtz) Where the camera is looking.
lookAtz	(lookAtx, lookAty, lookAtz) Where the camera is looking.
upx	<upx, upy, upz> Unit vector describing which direction is up for the camera.
upy	<upx, upy, upz> Unit vector describing which direction is up for the camera.
upz	<upx, upy, upz> Unit vector describing which direction is up for the camera.

<convert float to f32

<convert float to f32

<convert float to f32

<convert float to f32

<convert float to f32

<convert float to f32

<convert float to f32

<convert float to f32

<convert float to f32

Examples:

Graphics/3D/Display_List/source/main.cpp, Graphics/3D/Display_List_2/source/main.cpp, Graphics/3D/Paletted_Cube/source/main.cpp, Graphics/3D/Picking/source/main.cpp, Graphics/3D/Simple_Quad/source/main.cpp, Graphics/3D/Simple_Tri/source/main.cpp, Graphics/3D/Textured_Cube/source/main.cpp, Graphics/3D/Textured_Quad/source/main.cpp, Graphics/3D/Toon_Shading/source/main.cpp, and time/RealTimeClock/source/main.c.

```
static void gluLookAtf32  ( int  eyex,
                            int  eyey,
                            int  eyez,
                            int  lookAtx,
                            int  lookAty,
                            int  lookAtz,
                            int  upx,
                            int  upy,
                            int  upz
                          )     [inline]
```

Places the camera at the specified location and orientation (fixed point version)

Parameters:

eyex	(eyex, eyey, eyez) Location of the camera.
eyey	(eyex, eyey, eyez) Location of the camera.
eyez	(eyex, eyey, eyez) Location of the camera.
lookAtx	(lookAtx, lookAty, lookAtz) Where the camera is looking.
lookAty	(lookAtx, lookAty, lookAtz) Where the camera is looking.
lookAtz	(lookAtx, lookAty, lookAtz) Where the camera is looking.
upx	<upx, upy, upz> Unit vector describing which direction is up for the camera.
upy	<upx, upy, upz> Unit vector describing which direction is up for the camera.
upz	<upx, upy, upz> Unit vector describing which direction is up for the camera.

```
static void gluPerspective  ( float  fovy,
                              float  aspect,
                              float  zNear,
                              float  zFar
                            )        [inline]
```

Utility function which sets up the projection matrix (floating point version)

Warning:

FLOAT VERSION!!!! please use gluPerspectivef32()

Parameters:

fovy	Specifies the field of view in degrees
aspect	Specifies the aspect ratio of the screen (normally screen width/screen height)
zNear	Specifies the near clipping plane
zFar	Specifies the far clipping plane

<convert float to f32

<convert float to f32

<convert float to f32

Examples:

Graphics/3D/3D_Both_Screens/source/template.c, Graphics/3D/BoxTest/source/main.cpp, Graphics/3D/Display_List/source/main.cpp, Graphics/3D/Display_List_2/source/main.cpp, Graphics/3D/Env_Mapping/source/main.cpp, Graphics/3D/nehe/lesson01/source/nehe1.cpp, Graphics/3D/nehe/lesson02/source/nehe2.cpp, Graphics/3D/nehe/lesson03/source/nehe3.cpp, Graphics/3D/nehe/lesson04/source/nehe4.cpp, Graphics/3D/nehe/lesson05/source/nehe5.cpp, Graphics/3D/nehe/lesson06/source/nehe6.cpp, Graphics/3D/nehe/lesson07/source/nehe7.cpp, Graphics/3D/nehe/lesson08/source/nehe8.cpp, Graphics/3D/nehe/lesson09/source/nehe9.cpp, Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/3D/nehe/lesson10b/source/nehe10b.cpp, Graphics/3D/nehe/lesson11/source/nehe11.cpp, Graphics/3D/Ortho/source/main.cpp, Graphics/3D/Paletted_Cube/source/main.cpp, Graphics/3D/Picking/source/main.cpp, Graphics/3D/Simple_Quad/source/main.cpp, Graphics/3D/Simple_Tri/source/main.cpp, Graphics/3D/Textured_Cube/source/main.cpp, Graphics/3D/Textured_Quad/source/main.cpp, Graphics/3D/Toon_Shading/source/main.cpp, input/Touch_Pad/touch_look/source/main.cpp, and time/RealTimeClock/source/main.c.

static void gluPerspectivef32 (int *fovy*,
int *aspect*,
int *zNear*,
int *zFar*
) [inline]

Utility function which sets up the projection matrix (fixed point version)

Parameters:

fovy	Specifies the field of view in degrees (-32768 to 32767)
aspect	Specifies the aspect ratio of the screen (normally screen width/screen height)
zNear	Specifies the near clipping plane
zFar	Specifies the far clipping plane

static void gluPickMatrix (int *x*,
int *y*,
int *width*,
int *height*,
const int *viewport*[4]
) [inline]

Utility function which generates a picking matrix for selection.

Parameters:

x	2D x of center (touch x normally)
y	2D y of center (touch y normally)
width	width in pixels of the window (3 or 4 is a good number)
height	height in pixels of the window (3 or 4 is a good number)
viewport	the current viewport (normally {0, 0, 255, 191})

<convert int to f32

<convert int to f32

<convert int to f32

<convert int to f32

<convert int to f32

<convert int to f32

Examples:

Graphics/3D/Picking/source/main.cpp.

static void glVertex3f (float *x*,
float *y*,
float *z*

```
                         )      [inline]
```
specifies a vertex location

Warning:

FLOAT VERSION!!!! please use glVertex3v16()

Parameters:

x the x component of the vertex

y the y component of the vertex

z the z component of the vertex

<convert float to v16

<convert float to v16

<convert float to v16

Examples:

Graphics/3D/3D_Both_Screens/source/template.c, Graphics/3D/BoxTest/source/main.cpp, Graphics/3D/nehe/lesson02/source/nehe2.cpp, Graphics/3D/nehe/lesson03/source/nehe3.cpp, Graphics/3D/nehe/lesson04/source/nehe4.cpp, Graphics/3D/nehe/lesson05/source/nehe5.cpp, Graphics/3D/nehe/lesson06/source/nehe6.cpp, Graphics/3D/nehe/lesson07/source/nehe7.cpp, Graphics/3D/nehe/lesson08/source/nehe8.cpp, Graphics/3D/nehe/lesson09/source/nehe9.cpp, Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/3D/Ortho/source/main.cpp, and time/RealTimeClock/source/main.c.

```
static void glVertex3v16  ( v16  x,
                            v16  y,
                            v16  z
                          )      [inline]
```
specifies a vertex

Parameters:

x the x component for the vertex

y the y component for the vertex

z the z component for the vertex

Examples:

Graphics/3D/nehe/lesson10b/source/nehe10b.cpp, Graphics/3D/nehe/lesson11/source/nehe11.cpp, Graphics/3D/Paletted_Cube/source/main.cpp, Graphics/3D/Simple_Quad/source/main.cpp, Graphics/3D/Simple_Tri/source/main.cpp, Graphics/3D/Textured_Cube/source/main.cpp, Graphics/3D/Textured_Quad/source/main.cpp, and input/Touch_Pad/touch_look/source/main.cpp.

```
static void glViewport  ( uint8  x1,
                          uint8  y1,
                          uint8  x2,
                          uint8  y2
```

) [inline]

specify the viewport for following drawing, can be set several times per frame. GBATEK http://nocash.emubase.de/gbatek.htm#ds3ddisplaycontrol

Parameters:

x1	the left of the viewport
y1	the bottom of the viewport
x2	the right of the viewport
y2	the top of the viewport

Examples:

Graphics/3D/3D_Both_Screens/source/template.c, Graphics/3D/BoxTest/source/main.cpp, Graphics/3D/Display_List/source/main.cpp, Graphics/3D/Display_List_2/source/main.cpp, Graphics/3D/Env_Mapping/source/main.cpp, Graphics/3D/nehe/lesson01/source/nehe1.cpp, Graphics/3D/nehe/lesson02/source/nehe2.cpp, Graphics/3D/nehe/lesson03/source/nehe3.cpp, Graphics/3D/nehe/lesson04/source/nehe4.cpp, Graphics/3D/nehe/lesson05/source/nehe5.cpp, Graphics/3D/nehe/lesson06/source/nehe6.cpp, Graphics/3D/nehe/lesson07/source/nehe7.cpp, Graphics/3D/nehe/lesson08/source/nehe8.cpp, Graphics/3D/nehe/lesson09/source/nehe9.cpp, Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/3D/nehe/lesson10b/source/nehe10b.cpp, Graphics/3D/nehe/lesson11/source/nehe11.cpp, Graphics/3D/Ortho/source/main.cpp, Graphics/3D/Paletted_Cube/source/main.cpp, Graphics/3D/Picking/source/main.cpp, Graphics/3D/Simple_Quad/source/main.cpp, Graphics/3D/Simple_Tri/source/main.cpp, Graphics/3D/Textured_Cube/source/main.cpp, Graphics/3D/Textured_Quad/source/main.cpp, Graphics/3D/Toon_Shading/source/main.cpp, input/Touch_Pad/touch_look/source/main.cpp, and time/RealTimeClock/source/main.c.

static u32 POLY_ALPHA (int *n*) [inline]

used in glPolyFmt() to set the alpha level for the following polygons, set to 0 for wireframe mode

Parameters:

n	the level of alpha (0-31)

Examples:

Graphics/3D/3D_Both_Screens/source/template.c, Graphics/3D/BoxTest/source/main.cpp, Graphics/3D/Display_List/source/main.cpp, Graphics/3D/Display_List_2/source/main.cpp, Graphics/3D/Env_Mapping/source/main.cpp, Graphics/3D/nehe/lesson01/source/nehe1.cpp, Graphics/3D/nehe/lesson02/source/nehe2.cpp, Graphics/3D/nehe/lesson03/source/nehe3.cpp, Graphics/3D/nehe/lesson04/source/nehe4.cpp, Graphics/3D/nehe/lesson05/source/nehe5.cpp, Graphics/3D/nehe/lesson06/source/nehe6.cpp, Graphics/3D/nehe/lesson07/source/nehe7.cpp, Graphics/3D/nehe/lesson08/source/nehe8.cpp, Graphics/3D/nehe/lesson09/source/nehe9.cpp, Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/3D/nehe/lesson10b/source/nehe10b.cpp, Graphics/3D/nehe/lesson11/source/nehe11.cpp, Graphics/3D/Ortho/source/main.cpp, Graphics/3D/Paletted_Cube/source/main.cpp, Graphics/3D/Picking/source/main.cpp, Graphics/3D/Simple_Quad/source/main.cpp, Graphics/3D/Simple_Tri/source/main.cpp, Graphics/3D/Textured_Cube/source/main.cpp, Graphics/3D/Textured_Quad/source/main.cpp, Graphics/3D/Toon_Shading/source/main.cpp, input/Touch_Pad/touch_look/source/main.cpp, and time/RealTimeClock/source/main.c.

static int POLY_ID (int *n*) `[inline]`

used in glPolyFmt() to set the Polygon ID for the following polygons

Parameters:

n the ID to set for following polygons (0-63)

Examples:

Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/3D/Paletted_Cube/source/main.cpp, and Graphics/3D/Picking/source/main.cpp.

boxtest.h File Reference

Box Test Functions. More...

```
#include "nds/arm9/video.h"
#include "nds/arm9/videoGL.h"
```

Functions

int BoxTest (v16 x, v16 y, v16 z, v16 width, v16 height, v16 depth)
Performs a test to determine if the provided box is in the view frustrum.

void BoxTest_Asynch (v16 x, v16 y, v16 z, v16 height, v16 width, v16 depth)
Performs a test to determine if the provided box is in the view frustum. Performs a test to determine if the provided box is in the view frustum. BoxTestResult must be called to get the result of this operation.

int BoxTestf (float x, float y, float z, float width, float height, float depth)
Performs a test to determine if the provided box is in the view frustum.

void BoxTestf_Asynch (float x, float y, float z, float width, float height, float depth)
Performs a test to determine if the provided box is in the view frustum. Performs a test to determine if the provided box is in the view frustum. BoxTestResult must be called to get the result of this operation.

int BoxTestResult (void)
Gets the result of the last box test. Needed for asynch box test calls.

Detailed Description

Box Test Functions.

Function Documentation

```
int BoxTest ( v16 x,
              v16 y,
              v16 z,
              v16 width,
              v16 height,
              v16 depth
            )
```

Performs a test to determine if the provided box is in the view frustrum.

Parameters:

x	(x, y, z) point of a vertex on the box
y	(x, y, z) point of a vertex on the box
z	(x, y, z) point of a vertex on the box
height	(height, width, depth) describe the size of the box referenced from (x, y,

	z)
width	(height, width, depth) describe the size of the box referenced from (x, y, z)
depth	(height, width, depth) describe the size of the box referenced from (x, y, z)

Returns:

non zero if any or all of the box is in the view frustum.

Examples:

Graphics/3D/BoxTest/source/main.cpp.

```
void BoxTest_Asynch  ( v16  x,
                       v16  y,
                       v16  z,
                       v16  height,
                       v16  width,
                       v16  depth
                     )
```

Performs a test to determine if the provided box is in the view frustum. Performs a test to determine if the provided box is in the view frustum. BoxTestResult must be called to get the result of this operation.

Parameters:

x	(x, y, z) point of a vertex on the box
y	(x, y, z) point of a vertex on the box
z	(x, y, z) point of a vertex on the box
width	(width, height, depth) describe the size of the box referenced from (x, y, z)
height	(width, height, depth) describe the size of the box referenced from (x, y, z)
depth	(width, height, depth) describe the size of the box referenced from (x, y, z)

```
int BoxTestf  ( float  x,
                float  y,
                float  z,
                float  width,
                float  height,
                float  depth
              )
```

Performs a test to determine if the provided box is in the view frustum.

Parameters:

x	(x, y, z) point of a vertex on the box
y	(x, y, z) point of a vertex on the box
z	(x, y, z) point of a vertex on the box
width	(width, height, depth) describe the size of the box referenced from (x, y, z)

height	(width, height, depth) describe the size of the box referenced from (x, y, z)
depth	(width, height, depth) describe the size of the box referenced from (x, y, z)

Returns:

non zero if any or all of the box is in the view frustum.

Examples:

Graphics/3D/BoxTest/source/main.cpp.

```
void BoxTestf_Asynch  (  float  x,
                         float  y,
                         float  z,
                         float  width,
                         float  height,
                         float  depth
                      )
```

Performs a test to determine if the provided box is in the view frustum. Performs a test to determine if the provided box is in the view frustum. BoxTestResult must be called to get the result of this operation.

Parameters:

x	(x, y, z) point of a vertex on the box
y	(x, y, z) point of a vertex on the box
z	(x, y, z) point of a vertex on the box
width	(width, height, depth) describe the size of the box referenced from (x, y, z)
height	(width, height, depth) describe the size of the box referenced from (x, y, z)
depth	(width, height, depth) describe the size of the box referenced from (x, y, z)

```
int BoxTestResult  (  void  )
```

Gets the result of the last box test. Needed for asynch box test calls.

Returns:

non zero if any or all of the box is in the view frustum.

postest.h File Reference

Position Test Functions.
More...

```
#include <nds/arm9/video.h>
#include <nds/arm9/videoGL.h>
```

Functions

GL_STATIC_INL void PosTest (v16 x, v16 y, v16 z)
Performs a position test.

GL_STATIC_INL void PosTest_Asynch (v16 x, v16 y, v16 z)
Starts a position test asynchronously.

GL_STATIC_INL bool PosTestBusy ()
true if the hardware is currently performing a position/vertex/box test.

GL_STATIC_INL int32 PosTestWresult ()
Returns the distance from the camera of the last position test, pretty darn useful.

GL_STATIC_INL int32 PosTestXresult ()
Returns absolute X position of the last position test (location if the modelview matrix was identity)

GL_STATIC_INL int32 PosTestYresult ()
Returns absolute Y position of the last position test (location if the modelview matrix was identity)

GL_STATIC_INL int32 PosTestZresult ()
Returns absolute Z position of the last position test (location if the modelview matrix was identity)

Detailed Description

Position Test Functions.

The position test multiplies a given vector by the position matrix and returns the coords(x,y,z,w). The position test is really quick, about 10x faster than a box test.

Function Documentation

GL_STATIC_INL void PosTest (v16 *x*,

v16 *y*,
v16 *z*
)

Performs a position test.

Parameters:

x specifies x offset from the current modelview matrix
y specifies y offset from the current modelview matrix
z specifies z offset from the current modelview matrix

GL_STATIC_INL void PosTest_Asynch (v16 *x*,
v16 *y*,
v16 *z*
)

Starts a position test asynchronously.

Parameters:

x specifies x offset from the current modelview matrix
y specifies y offset from the current modelview matrix
z specifies z offset from the current modelview matrix

Examples:

Graphics/3D/Picking/source/main.cpp.

GL_STATIC_INL bool PosTestBusy ( )

true if the hardware is currently performing a position/vertex/box test.

Returns:

whether a test is being performed

Examples:

Graphics/3D/Picking/source/main.cpp.

GL_STATIC_INL int32 PosTestWresult ( )

Returns the distance from the camera of the last position test, pretty darn useful.

Returns:

W magnitude

Examples:

Graphics/3D/Picking/source/main.cpp.

GL_STATIC_INL int32 PosTestXresult ( )

Returns absolute X position of the last position test (location if the modelview matrix was identity)

Returns:

Absolute X position

GL_STATIC_INL int32 PosTestYresult ( )

Returns absolute Y position of the last position test (location if the modelview matrix was identity)

Returns:

Absolute Y position

GL_STATIC_INL int32 PosTestZresult ( )

Returns absolute Z position of the last position test (location if the modelview matrix was identity)

Returns:

Absolute Z position

Audio API

sound.h File Reference

A simple sound playback library for the DS. Provides functionality for starting and stopping sound effects from the ARM9 side as well as access to PSG and noise hardware. Maxmod should be used in most music and sound effect situations. More...

```
#include <nds/ndstypes.h>
```

Enumerations

```
enum DutyCycle {
    DutyCycle_0 = 7,
    DutyCycle_12 = 0,
    DutyCycle_25 = 1,
    DutyCycle_37 = 2,
    DutyCycle_50 = 3,
    DutyCycle_62 = 4,
    DutyCycle_75 = 5,
    DutyCycle_87 = 6
  }
```

PSG Duty cycles used by the PSG hardware.

More...

```
enum MicFormat {
    MicFormat_8Bit = 1,
    MicFormat_12Bit = 0
  }
```

Microphone recording formats DS.

More...

```
enum SoundFormat {
    SoundFormat_16Bit = 1,
    SoundFormat_8Bit = 0,
    SoundFormat_PSG = 3,
    SoundFormat_ADPCM = 2
  }
```

Sound formats used by the DS.

More...

Functions

void soundDisable (void)

Disables Sound on the DS.

void soundEnable (void)

Enables Sound on the DS. Should be called prior to attempting sound playback.

void	soundKill (int soundId)
	Stops the sound specified by soundId and frees any resources allocated.
void	soundMicOff (void)
	Stops the microphone from recording.
int	soundMicRecord (void *buffer, u32 bufferLength, MicFormat format, int freq, MicCallback callback)
	Starts a microphone recording to a double buffer specified by buffer.
void	soundPause (int soundId)
	Pause the sound specified by soundId.
int	soundPlayNoise (u16 freq, u8 volume, u8 pan)
	Plays white noise with the specified parameters.
int	soundPlayPSG (DutyCycle cycle, u16 freq, u8 volume, u8 pan)
	Pause a tone with the specified properties.
int	soundPlaySample (const void *data, SoundFormat format, u32 dataSize, u16 freq, u8 volume, u8 pan, bool loop, u16 loopPoint)
	Plays a sound in the specified format at the specified frequency.
void	soundResume (int soundId)
	Resumes a paused sound.
void	soundSetFreq (int soundId, u16 freq)
	Sets the sound frequency.
void	soundSetPan (int soundId, u8 pan)
	Sets the sound pan.
void	soundSetVolume (int soundId, u8 volume)
	Sets the sound volume.
void	soundSetWaveDuty (int soundId, DutyCycle cycle)
	Sets the Wave Duty of a PSG sound.

Detailed Description

A simple sound playback library for the DS. Provides functionality for starting and stopping sound effects from the ARM9 side as well as access to PSG and noise hardware. Maxmod should be used in most music and sound effect situations.

Enumeration Type Documentation

enum DutyCycle

PSG Duty cycles used by the PSG hardware.

Enumerator:

DutyCycle_0	0.0% duty cycle
DutyCycle_12	12.5% duty cycle
DutyCycle_25	25.0% duty cycle

DutyCycle_37	37.5% duty cycle
DutyCycle_50	50.0% duty cycle
DutyCycle_62	62.5% duty cycle
DutyCycle_75	75.0% duty cycle
DutyCycle_87	87.5% duty cycle

enum MicFormat
Microphone recording formats DS.

Enumerator:

MicFormat_8Bit	8-bit PCM
MicFormat_12Bit	12-bit PCM

enum SoundFormat
Sound formats used by the DS.

Enumerator:

SoundFormat_16Bit	16-bit PCM
SoundFormat_8Bit	8-bit PCM
SoundFormat_PSG	PSG (programmable sound generator?)
SoundFormat_ADPCM	IMA ADPCM compressed audio

Function Documentation

void soundKill (int *soundId*)
Stops the sound specified by soundId and frees any resources allocated.

Parameters:

soundId	The sound ID returned by play sound

```
int soundMicRecord ( void *       buffer,
                     u32          bufferLength,
                     MicFormat    format,
                     int          freq,
                     MicCallback  callback
                   )
```

Starts a microphone recording to a double buffer specified by buffer.

Parameters:

buffer	A pointer to the start of the double buffer

bufferLength	The length of the buffer in bytes (both halfs of the double buffer)
format	Microphone can record in 8 or 12 bit format. 12 bit is shifted up to 16 bit pcm
freq	The sample frequency
callback	This will be called every time the buffer is full or half full

Returns:

Returns non zero for success.

Examples:

audio/micrecord/source/micrecord.c.

void soundPause (int *soundId*)

Pause the sound specified by soundId.

Parameters:

soundId	The sound ID returned by play sound

Examples:

time/timercallback/source/main.c.

```
int soundPlayNoise  ( u16  freq,
                      u8   volume,
                      u8   pan
                    )
```

Plays white noise with the specified parameters.

Parameters:

freq	The frequency in Hz of the sample
volume	The channel volume. 0 to 127 (min to max)
pan	The channel pan 0 to 127 (left to right with 64 being centered)

Returns:

An integer id coresponding to the channel of playback. This value can be used to pause, resume, or kill the sound as well as adjust volume, pan, and frequency

```
int soundPlayPSG  ( DutyCycle  cycle,
                    u16        freq,
                    u8         volume,
                    u8         pan
                  )
```

Pause a tone with the specified properties.

Parameters:

cycle	The DutyCycle of the sound wave
freq	The frequency in Hz of the sample
volume	The channel volume. 0 to 127 (min to max)
pan	The channel pan 0 to 127 (left to right with 64 being centered)

Returns:

An integer id coresponding to the channel of playback. This value can be used to pause, resume, or kill the sound as well as adjust volume, pan, and frequency

Examples:

time/timercallback/source/main.c.

```
int soundPlaySample ( const void *   data,
                      SoundFormat   format,
                      u32           dataSize,
                      u16           freq,
                      u8            volume,
                      u8            pan,
                      bool          loop,
                      u16           loopPoint
                    )
```

Plays a sound in the specified format at the specified frequency.

Parameters:

data	A pointer to the sound data
format	The format of the data (only 16-bit and 8-bit pcm and ADPCM formats are supported by this function)
dataSize	The size in bytes of the sound data
freq	The frequency in Hz of the sample
volume	The channel volume. 0 to 127 (min to max)
pan	The channel pan 0 to 127 (left to right with 64 being centered)
loop	If true, the sample will loop playing once then repeating starting at the offset stored in loopPoint
loopPoint	The offset for the sample loop to restart when repeating

Returns:

An integer id coresponding to the channel of playback. This value can be used to pause, resume, or kill the sound as well as adjust volume, pan, and frequency

Examples:

audio/micrecord/source/micrecord.c.

```
void soundResume ( int soundId )
```

Resumes a paused sound.

Parameters:

soundId	The sound ID returned by play sound

Examples:

time/timercallback/source/main.c.

```
void soundSetFreq ( int  soundId,
                    u16  freq
                  )
```

Sets the sound frequency.

Parameters:

soundId	The sound ID returned by play sound
freq	The frequency in Hz

void soundSetPan (int *soundId*,
u8 *pan*
)

Sets the sound pan.

Parameters:

soundId	The sound ID returned by play sound
pan	The new pan value (0 to 127 left to right (64 = center))

void soundSetVolume (int *soundId*,
u8 *volume*
)

Sets the sound volume.

Parameters:

soundId	The sound ID returned by play sound
volume	The new volume (0 to 127 min to max)

void soundSetWaveDuty (int *soundId*,
DutyCycle *cycle*
)

Sets the Wave Duty of a PSG sound.

Parameters:

soundId	The sound ID returned by play sound
cycle	The DutyCycle of the sound wave

math.h File Reference

hardware coprocessor math instructions. More...

```
#include "nds/ndstypes.h"
```

Functions

static void	crossf32 (int32 *a, int32 *b, int32 *result) 1.19.12 fixed point cross product function result = AxB
static int32	div32 (int32 num, int32 den) integer divide
static int32	div64 (int64 num, int32 den) integer 64 bit divide
static int32	divf32 (int32 num, int32 den) Fixed point divide.
static int32	dotf32 (int32 *a, int32 *b) 1.19.12 fixed point dot product function result = A dot B
static int32	mod32 (int32 num, int32 den) integer modulous
static int32	mod64 (int64 num, int32 den) integer 64 bit modulous
static int32	mulf32 (int32 a, int32 b) Fixed point multiply.
static void	normalizef32 (int32 *a) 1.19.12 fixed point normalize function A = A / |A|
static u32	sqrt32 (int a) integer sqrt
static u32	sqrt64 (long long a) integer sqrt
static int32	sqrtf32 (int32 a) Fixed point sqrt.

Detailed Description

hardware coprocessor math instructions.

Function Documentation

static void crossf32 (int32 * *a*,

int32 * *b*,
int32 * *result*
) [inline, static]

1.19.12 fixed point cross product function result = AxB

Parameters:

a	pointer to fixed 3x3 matrix
b	pointer to fixed 3x3 matrix
result	pointer to fixed 3x3 matrix Cross product x = Ay * Bz - By * Az y = Az * Bx - Bz * Ax z = Ax * By - Bx * Ay

static int32 div32 (int32 *num*,
int32 *den*
) [inline, static]

integer divide

Parameters:

num	numerator
den	denominator

Returns:

returns 32 bit integer result

static int32 div64 (int64 *num*,
int32 *den*
) [inline, static]

integer 64 bit divide

Parameters:

num	64 bit numerator
den	32 bit denominator

Returns:

returns 32 bit integer result

static int32 divf32 (int32 *num*,
int32 *den*
) [inline, static]

Fixed point divide.

Parameters:

num	Takes 20.12 numerator.
den	Takes 20.12 denominator.

Returns:

returns 20.12 result.

static int32 dotf32 (int32 * *a*,
int32 * *b*
) [inline, static]

1.19.12 fixed point dot product function result = A dot B

Parameters:

a pointer to fixed 3x3 matrix

b pointer to fixed 3x3 matrix

Returns:

32 bit integer result Dot Product result = Ax * Bx + Ay * By + Az * Bz

```
static int32 mod32 ( int32 num,
                     int32 den
                   )      [inline, static]
```

integer modulous

Parameters:

num numerator

den denominator

Returns:

returns 32 bit integer remainder

```
static int32 mod64 ( int64 num,
                     int32 den
                   )      [inline, static]
```

integer 64 bit modulous

Parameters:

num 64 bit numerator

den 32 bit denominator

Returns:

returns 32 bit integer remainder

```
static int32 mulf32 ( int32 a,
                      int32 b
                    )      [inline, static]
```

Fixed point multiply.

Parameters:

a Takes 20.12

b Takes 20.12

Returns:

returns 20.12 result

```
static void normalizef32 ( int32 * a ) [inline, static]
```

1.19.12 fixed point normalize function A = A / |A|

Parameters:

a pointer to fixed 3x3 matrix Normalize Ax = Ax / mag Ay = Ay / mag Az = Az / mag

```
static u32 sqrt32 ( int a ) [inline, static]
```

integer sqrt

Parameters:

a 32 bit integer argument

Returns:

returns 32 bit integer result

static u32 sqrt64 (long long *a*) `[inline, static]`
integer sqrt

Parameters:
a 64 bit integer argument

Returns:
returns 32 bit integer result

static int32 sqrtf32 (int32 *a*) `[inline, static]`
Fixed point sqrt.

Parameters:
a Takes 20.12

Returns:
returns 20.12 result

trig_lut.h File Reference

fixed point trig functions. Angle can be in the range of -32768 to 32767. There are 32768 degrees in the unit circle used by nds. To convert between standard degrees (360 per circle): More...

```
#include <nds/ndstypes.h>
```

Defines

#define angleToDegrees(angle)   ((angle) * 360 / DEGREES_IN_CIRCLE)
  converts an angle in the format used by libnds in the 360 degree format.

#define DEGREES_IN_CIRCLE   (1 << 15)
  number of degrees in a circle.

#define degreesToAngle(degrees)   ((degrees) * DEGREES_IN_CIRCLE / 360)
  convert an angle in 360 degree format to the format used by libnds.

#define fixedToFloat(n, bits)   (((float)(n)) / (float)(1<<(bits)))
  converts a fixed point number to a floating point number.

#define fixedToInt(n, bits)   ((int)((n)>>(bits)))
  convert a fixed point number to an integer.

#define floatToFixed(n, bits)   ((int)((n) * (float)(1<<(bits))))
  converts a floating point number to a fixed point number.

#define floorFixed(n, bits)   ((int)((n) & ~(((1 << (bits)) - 1))))
  removes the decimal part of a fixed point number.

#define intToFixed(n, bits)   ((int)((n)<<(bits)))
  converts an integer to a fixed point number.

Functions

s16 acosLerp (s16 par)
  fixed point arccos

s16 asinLerp (s16 par)
  fixed point arcsin

s16 cosLerp (s16 angle)
  fixed point cosine

s16 sinLerp (s16 angle)
  fixed point sine

s32 tanLerp (s16 angle)
  fixed point tangent

Detailed Description

fixed point trig functions. Angle can be in the range of -32768 to 32767. There are 32768 degrees in the unit circle used by nds. To convert between standard degrees (360 per circle):

angle = degreesToAngle(angleInDegrees);

or

angle = angleInDegrees * 32768 / 360;

This unit of measure is sometimes refered to as a binary radian (brad) or binary degree. It allows for more precise representation of angle and faster calculation as the DS has no floating point processor.

Define Documentation

```
#define fixedToFloat (  n,
                        bits
                     )    (((float)(n)) / (float)(1<<(bits)))
```

converts a fixed point number to a floating point number.

Parameters:

n	the fixed point number to convert.
bits	the number of bits used for the decimal part.

Returns:

the floating point number.

```
#define fixedToInt (  n,
                      bits
                   )    ((int)((n)>>(bits)))
```

convert a fixed point number to an integer.

Parameters:

n	the number the number to convert.
bits	the number of bits used for the decimal part.

Returns:

the integer part.

```
#define floatToFixed (  n,
                        bits
                     )    ((int)((n) * (float)(1<<(bits))))
```

converts a floating point number to a fixed point number.

Parameters:

n	the floating point number to convert.
bits	the number of bits used for the decimal part.

Returns:

the fixed point number.

```
#define floorFixed (  n,
                      bits
                   )    ((int)((n) & ~(((1 << (bits)) - 1))))
```

removes the decimal part of a fixed point number.

Parameters:

n the fixed point number.

bits the number of bits used for the decimal part.

Returns:

a fixed point number with 0 as a decimal part.

```
#define intToFixed (    n,
                        bits
                    )       ((int)((n)<<(bits)))
```

converts an integer to a fixed point number.

Parameters:

n the integer to convert.

bits the number of bits used for the decimal part.

Returns:

the fixed point number.

Examples:

Graphics/Printing/rotscale_text/source/main.c, and
Graphics/Sprites/sprite_rotate/source/template.c.

Function Documentation

s16 acosLerp (s16 *par*)

fixed point arccos

Parameters:

par 4.12 fixed point number with the range [-1, 1]

Returns:

s16 angle (-32768 to 32767)

s16 asinLerp (s16 *par*)

fixed point arcsin

Parameters:

par 4.12 fixed point number with the range [-1, 1]

Returns:

s16 angle (-32768 to 32767)

s16 cosLerp (s16 *angle*)

fixed point cosine

Parameters:

angle (-32768 to 32767)

Returns:

4.12 fixed point number with the range [-1, 1]

Examples:

capture/ScreenShot/source/main.cpp,

Graphics/3D/nehe/lesson10/source/nehe10.cpp,
Graphics/3D/nehe/lesson10b/source/nehe10b.cpp,
Graphics/3D/nehe/lesson11/source/nehe11.cpp, and
input/Touch_Pad/touch_look/source/main.cpp.

s16 sinLerp (s16 *angle*)

fixed point sine

Parameters:

angle (-32768 to 32767)

Returns:

4.12 fixed point number with the range [-1, 1]

Examples:

audio/maxmod/streaming/source/main.c, capture/ScreenShot/source/main.cpp,
Graphics/3D/nehe/lesson10/source/nehe10.cpp,
Graphics/3D/nehe/lesson10b/source/nehe10b.cpp,
Graphics/3D/nehe/lesson11/source/nehe11.cpp, and
input/Touch_Pad/touch_look/source/main.cpp.

s32 tanLerp (s16 *angle*)

fixed point tangent

Parameters:

angle (-32768 to 32767)

Returns:

20.12 fixed point number with the range [-81.483, 524287.999]

memory.h File Reference

Defines for many of the regions of memory on the DS as well as a few control functions for memory bus access. More...

```
#include "ndstypes.h"
```

Data Structures

struct sGBAHeader
the GBA file header format. See gbatek for more info. More...

struct sNDSBanner
the NDS banner format. See gbatek for more information. More...

struct sNDSHeader
the NDS file header format See gbatek for more info. More...

Defines

#define ALLRAM ((u8*)0x00000000)
8 bit pointer to the start of all the ram.

#define GBA_BUS ((vu16 *)(0x08000000))
16 bit volatile pointer to the GBA slot bus.

#define GBAROM ((u16*)0x08000000)
16 bit pointer to the GBA slot ROM.

#define MAINRAM16 ((u16*)0x02000000)
16 bit pointer to main ram.

#define MAINRAM32 ((u32*)0x02000000)
32 bit pointer to main ram.

#define MAINRAM8 ((u8*)0x02000000)
8 bit pointer to main ram.

#define SRAM ((u8*)0x0A000000)
8 bit pointer to GBA slot Save ram.

Functions

struct sGBAHeader __attribute__ ((__packed__)) tGBAHeader
the GBA file header format. See gbatek for more info.

void sysSetBusOwners (bool arm9rom, bool arm9card)
Sets the owner of the DS card bus (slot 1) and gba cart bus (slot 2). Only one cpu may access the device at a time.

void sysSetCardOwner (bool arm9)
Sets the owner of the DS card bus. Both CPUs cannot have access to

	the DS card bus (slot 1).
void	sysSetCartOwner (bool arm9)
	Sets the owner of the GBA cart. Both CPUs cannot have access to the gba cart (slot 2).

Variables

u32	arm7binarySize
	size of the arm7 binary.
u32	arm7destination
	destination address to where the arm7 binary should be copied.
u32	arm7executeAddress
	adress that should be executed after the binary has been copied.
u32	arm7overlaySize
	File arm7 overlay size.
u32	arm7overlaySource
	File arm7 overlay offset.
u32	arm7romOffset
	offset of the arm7 binary in the nds file.
u32	arm9binarySize
	size of the arm9 binary.
u32	arm9destination
	destination address to where the arm9 binary should be copied.
u32	arm9executeAddress
	adress that should be executed after the binary has been copied.
u32	arm9overlaySize
	File arm9 overlay size.
u32	arm9overlaySource
	File arm9 overlay offset.
u32	arm9romOffset
	offset of the arm9 binary in the nds file.
u32	bannerOffset
	offset to the banner with icon and titles etc.
u32	bfPrime1
	Secure Area Disable part 1.
u32	bfPrime2
	Secure Area Disable part 2.
u32	cardControl13
	Port 40001A4h setting for normal commands (used in modes 1 and 3)
u32	cardControlBF
	Port 40001A4h setting for KEY1 commands (used in mode 2)
u16	checksum
	a 16 bit checksum? (gbatek says its unused/reserved).
u8	complement

complement checksum of the gba header.

u16 crc
: 16 bit crc/checksum of the banner.

u32 debugRomDestination
: debug RAM destination.

u32 debugRomSize
: debug size.

u32 debugRomSource
: debug ROM offset.

u8 devicecode
: used by nintedo's hardware debuggers. normally 0.

u8 deviceSize
: capacity of the device (1 << n Mbit)

u8 deviceType
: type of device in the game card

u32 entryPoint
: 32 bits arm opcode to jump to executable code.

u32 fatOffset
: File Allocation Table (FAT) offset.

u32 fatSize
: File Allocation Table (FAT) size.

u32 filenameOffset
: File Name Table (FNT) offset.

u32 filenameSize
: File Name Table (FNT) size.

u8 flags
: bit 2: auto-boot flag.

char gamecode [0x4]
: 4 characters for the game code. first character is usally A or B, next 2 characters is a short title < and last character is for destination/language.

char gameCode [4]
: 4 characters for the game code.

char gameTitle [12]
: 12 characters for the game title.

u8 gbaLogo [156]
: Nintendo logo needed for booting the game.

u16 headerCRC16
: header checksum, CRC-16.

u32 headerSize
: ROM header size.

u8 icon [512]
: 32*32 icon of the game with 4 bit per pixel.

u8 is96h
: fixed value that is always 96h.

u8	logo [156] nintendo logo needed for booting the game.
u16	logoCRC16 Nintendo Logo Checksum, CRC-16.
u16	makercode identifies the (commercial) developer.
u16	palette [16] the pallete of the icon.
u16	readTimeout Secure Area Loading Timeout.
u32	romSize total size of the ROM.
u8	romversion version of the ROM.
u16	secureCRC16 Secure Area Checksum, CRC-16.
char	title [0xC] 12 characters for the game title.
u16	titles [6][128] title of the game in 6 different languages.
u8	unitcode identifies the required hardware.
u8	unitCode identifies the required hardware.
u32	unknownRAM1 ARM9 Auto Load List RAM Address (?)
u32	unknownRAM2 ARM7 Auto Load List RAM Address (?)
u8	version the version of the game.

Detailed Description

Defines for many of the regions of memory on the DS as well as a few control functions for memory bus access.

Function Documentation

struct sGBAHeader __attribute__ ((__packed__))

the GBA file header format. See gbatek for more info.

the NDS banner format. See gbatek for more information.

the NDS file header format See gbatek for more info.

static void sysSetBusOwners (bool *arm9rom*,
bool *arm9card*
) [inline]

Sets the owner of the DS card bus (slot 1) and gba cart bus (slot 2). Only one cpu may access the device at a time.

Parameters:

arm9rom	if true the arm9 is the owner of slot 2, otherwise the arm7
arm9card	if true the arm9 is the owner of slot 1, otherwise the arm7

Examples:

card/eeprom/source/main.cpp.

static void sysSetCardOwner (bool *arm9*) [inline]

Sets the owner of the DS card bus. Both CPUs cannot have access to the DS card bus (slot 1).

Parameters:

arm9 if true the arm9 is the owner, otherwise the arm7

static void sysSetCartOwner (bool *arm9*) [inline]

Sets the owner of the GBA cart. Both CPUs cannot have access to the gba cart (slot 2).

Parameters:

arm9 if true the arm9 is the owner, otherwise the arm7

Variable Documentation

u16 version

the version of the game.

version of the banner.

---

memory.h File Reference

Defines for many of the regions of memory on the DS as well as a few control functions for memory bus access. More...

```
#include "ndstypes.h"
```

Data Structures

struct	sGBAHeader the GBA file header format. See gbatek for more info. More...
struct	sNDSBanner the NDS banner format. See gbatek for more information. More...
struct	sNDSHeader the NDS file header format See gbatek for more info. More...

Defines

#define	ALLRAM ((u8*)0x00000000) 8 bit pointer to the start of all the ram.
#define	GBA_BUS ((vu16 *)(0x08000000)) 16 bit volatile pointer to the GBA slot bus.
#define	GBAROM ((u16*)0x08000000) 16 bit pointer to the GBA slot ROM.
#define	MAINRAM16 ((u16*)0x02000000) 16 bit pointer to main ram.
#define	MAINRAM32 ((u32*)0x02000000) 32 bit pointer to main ram.
#define	MAINRAM8 ((u8*)0x02000000) 8 bit pointer to main ram.
#define	SRAM ((u8*)0x0A000000) 8 bit pointer to GBA slot Save ram.

Functions

struct sGBAHeader	__attribute__ ((__packed__)) tGBAHeader the GBA file header format. See gbatek for more info.
void	sysSetBusOwners (bool arm9rom, bool arm9card) Sets the owner of the DS card bus (slot 1) and gba cart bus (slot 2). Only one cpu may access the device at a time.
void	sysSetCardOwner (bool arm9) Sets the owner of the DS card bus. Both CPUs cannot have access to the DS card bus (slot 1).
void	sysSetCartOwner (bool arm9) Sets the owner of the GBA cart. Both CPUs cannot have access to the

gba cart (slot 2).

Variables

u32	arm7binarySize
	size of the arm7 binary.
u32	arm7destination
	destination address to where the arm7 binary should be copied.
u32	arm7executeAddress
	adress that should be executed after the binary has been copied.
u32	arm7overlaySize
	File arm7 overlay size.
u32	arm7overlaySource
	File arm7 overlay offset.
u32	arm7romOffset
	offset of the arm7 binary in the nds file.
u32	arm9binarySize
	size of the arm9 binary.
u32	arm9destination
	destination address to where the arm9 binary should be copied.
u32	arm9executeAddress
	adress that should be executed after the binary has been copied.
u32	arm9overlaySize
	File arm9 overlay size.
u32	arm9overlaySource
	File arm9 overlay offset.
u32	arm9romOffset
	offset of the arm9 binary in the nds file.
u32	bannerOffset
	offset to the banner with icon and titles etc.
u32	bfPrime1
	Secure Area Disable part 1.
u32	bfPrime2
	Secure Area Disable part 2.
u32	cardControl13
	Port 40001A4h setting for normal commands (used in modes 1 and 3)
u32	cardControlBF
	Port 40001A4h setting for KEY1 commands (used in mode 2)
u16	checksum
	a 16 bit checksum? (gbatek says its unused/reserved).
u8	complement
	complement checksum of the gba header.
u16	crc
	16 bit crc/checksum of the banner.

u32 debugRomDestination
: debug RAM destination.

u32 debugRomSize
: debug size.

u32 debugRomSource
: debug ROM offset.

u8 devicecode
: used by nintedo's hardware debuggers. normally 0.

u8 deviceSize
: capacity of the device (1 << n Mbit)

u8 deviceType
: type of device in the game card

u32 entryPoint
: 32 bits arm opcode to jump to executable code.

u32 fatOffset
: File Allocation Table (FAT) offset.

u32 fatSize
: File Allocation Table (FAT) size.

u32 filenameOffset
: File Name Table (FNT) offset.

u32 filenameSize
: File Name Table (FNT) size.

u8 flags
: bit 2: auto-boot flag.

char gamecode [0x4]
: 4 characters for the game code. first character is usally A or B, next 2 characters is a short title < and last character is for destination/language.

char gameCode [4]
: 4 characters for the game code.

char gameTitle [12]
: 12 characters for the game title.

u8 gbaLogo [156]
: Nintendo logo needed for booting the game.

u16 headerCRC16
: header checksum, CRC-16.

u32 headerSize
: ROM header size.

u8 icon [512]
: 32*32 icon of the game with 4 bit per pixel.

u8 is96h
: fixed value that is always 96h.

u8 logo [156]
: nintendo logo needed for booting the game.

u16 logoCRC16

	Nintendo Logo Checksum, CRC-16.
u16	makercode
	identifies the (commercial) developer.
u16	palette [16]
	the pallete of the icon.
u16	readTimeout
	Secure Area Loading Timeout.
u32	romSize
	total size of the ROM.
u8	romversion
	version of the ROM.
u16	secureCRC16
	Secure Area Checksum, CRC-16.
char	title [0xC]
	12 characters for the game title.
u16	titles [6][128]
	title of the game in 6 different languages.
u8	unitcode
	identifies the required hardware.
u8	unitCode
	identifies the required hardware.
u32	unknownRAM1
	ARM9 Auto Load List RAM Address (?)
u32	unknownRAM2
	ARM7 Auto Load List RAM Address (?)
u8	version
	the version of the game.

Detailed Description

Defines for many of the regions of memory on the DS as well as a few control functions for memory bus access.

Function Documentation

struct sGBAHeader __attribute__ ((__packed__))

the GBA file header format. See gbatek for more info.

the NDS banner format. See gbatek for more information.

the NDS file header format See gbatek for more info.

static void sysSetBusOwners (bool *arm9rom*,
bool *arm9card*

) [inline]

Sets the owner of the DS card bus (slot 1) and gba cart bus (slot 2). Only one cpu may access the device at a time.

Parameters:

arm9rom	if true the arm9 is the owner of slot 2, otherwise the arm7
arm9card	if true the arm9 is the owner of slot 1, otherwise the arm7

Examples:

card/eeprom/source/main.cpp.

static void sysSetCardOwner (bool *arm9*) [inline]

Sets the owner of the DS card bus. Both CPUs cannot have access to the DS card bus (slot 1).

Parameters:

arm9	if true the arm9 is the owner, otherwise the arm7

static void sysSetCartOwner (bool *arm9*) [inline]

Sets the owner of the GBA cart. Both CPUs cannot have access to the gba cart (slot 2).

Parameters:

arm9	if true the arm9 is the owner, otherwise the arm7

Variable Documentation

u16 version

the version of the game.

version of the banner.

dma.h File Reference

Wrapper functions for direct memory access hardware. More...

```
#include "nds/ndstypes.h"
```

Functions

int dmaBusy (uint8 channel)
  determines if the specified channel is busy

void dmaCopy (const void *source, void *dest, uint32 size)
  copies from source to destination using channel 3 of DMA available channels in half words

void dmaCopyAsynch (const void *source, void *dest, uint32 size)
  copies from source to destination using channel 3 of DMA available channels in half words. This function returns immediately after starting the transfer.

void dmaCopyHalfWords (uint8 channel, const void *src, void *dest, uint32 size)
  copies from source to destination on one of the 4 available channels in half words

void dmaCopyHalfWordsAsynch (uint8 channel, const void *src, void *dest, uint32 size)
  copies from source to destination on one of the 4 available channels in half words. This function returns immediately after starting the transfer.

void dmaCopyWords (uint8 channel, const void *src, void *dest, uint32 size)
  copies from source to destination on one of the 4 available channels in words

void dmaCopyWordsAsynch (uint8 channel, const void *src, void *dest, uint32 size)
  copies from source to destination on one of the 4 available channels in half words. This function returns immediately after starting the transfer.

void dmaFillHalfWords (u16 value, void *dest, uint32 size)
  fills the source with the supplied value using DMA channel 3

void dmaFillWords (u32 value, void *dest, uint32 size)
  fills the source with the supplied value using DMA channel 3

Detailed Description

Wrapper functions for direct memory access hardware.

The DS has 4 hardware direct memory access devices per CPU which can be used to transfer or fill chunks of memeory without CPU intervention. Utilizing DMA is generaly faster than CPU copies (memcpy, swiCopy, for loops, etc..).
DMA has no access to data caches on the DS and as such will give unexpected results when DMAing data from main memory. The cache must be flushed as follows when using DMA to ensure proper opertion on the arm9:

```
DC_FlushRange(source, sizeof(dataToCopy));
dmaCopy(source, destination, sizeof(dataToCopy));
```

Function Documentation

static dmaBusy (uint8 *channel*) [inline]

determines if the specified channel is busy

Parameters:

channel	the dma channel to check (0 - 3).

Returns:

non zero if busy, 0 if channel is free

static void dmaCopy (const void * *source*, void * *dest*, uint32 *size*) [inline]

copies from source to destination using channel 3 of DMA available channels in half words

Parameters:

source	the source to copy from
dest	the destination to copy to
size	the size in bytes of the data to copy. Will be truncated to the nearest half word (2 bytes)

Examples:

audio/maxmod/song_events_example/source/template.c, audio/micrecord/source/micrecord.c, capture/ScreenShot/source/main.cpp, capture/ScreenShot/source/screenshot.cpp, Graphics/Backgrounds/256_color_bmp/source/main.cpp, Graphics/Backgrounds/all_in_one/source/advanced.cpp, Graphics/Backgrounds/all_in_one/source/basic.cpp, Graphics/Backgrounds/all_in_one/source/handmade.cpp, Graphics/Backgrounds/all_in_one/source/scrolling.cpp, Graphics/Backgrounds/rotation/source/main.cpp, and Graphics/Sprites/animate_simple/source/template.c.

static void dmaCopyAsynch (const void * *src*, void * *dest*, uint32 *size*) [inline]

copies from source to destination using channel 3 of DMA available channels in half words. This function returns immediately after starting the transfer.

Parameters:

src	the source to copy from
dest	the destination to copy to
size	the size in bytes of the data to copy. Will be truncated to the nearest half word (2 bytes)

static void dmaCopyHalfWords (uint8 *channel*, const void * *src*, void * *dest*, uint32 *size*) [inline]

copies from source to destination on one of the 4 available channels in half words

Parameters:

channel	the dma channel to use (0 - 3).
src	the source to copy from
dest	the destination to copy to
size	the size in bytes of the data to copy. Will be truncated to the nearest half word (2 bytes)

static void dmaCopyHalfWordsAsynch (uint8 *channel*,
const void * *src*,
void * *dest*,
uint32 *size*
) `[inline]`

copies from source to destination on one of the 4 available channels in half words. This function returns immediately after starting the transfer.

Parameters:

channel	the dma channel to use (0 - 3).
src	the source to copy from
dest	the destination to copy to
size	the size in bytes of the data to copy. Will be truncated to the nearest half word (2 bytes)

static void dmaCopyWords (uint8 *channel*,
const void * *src*,
void * *dest*,
uint32 *size*
) `[inline]`

copies from source to destination on one of the 4 available channels in words

Parameters:

channel	the dma channel to use (0 - 3).
src	the source to copy from
dest	the destination to copy to
size	the size in bytes of the data to copy. Will be truncated to the nearest word (4 bytes)

static void dmaCopyWordsAsynch (uint8 *channel*,
const void * *src*,
void * *dest*,
uint32 *size*
) `[inline]`

copies from source to destination on one of the 4 available channels in half words. This function returns immediately after starting the transfer.

Parameters:

channel	the dma channel to use (0 - 3).
src	the source to copy from
dest	the destination to copy to
size	the size in bytes of the data to copy. Will be truncated to the nearest word (4 bytes)

static void dmaFillHalfWords	(	u16	*value*,
		void *	*dest*,
		uint32	*size*
	)		`[inline]`

fills the source with the supplied value using DMA channel 3

Parameters:

value	the 16 byte value to fill memory with
dest	the destination to copy to
size	the size in bytes of the area to fill. Will be truncated to the nearest half word (2 bytes)

Examples:
audio/maxmod/song_events_example2/source/template.c, and Graphics/Sprites/bitmap_sprites/source/main.cpp.

static void dmaFillWords	(	u32	*value*,
		void *	*dest*,
		uint32	*size*
	)		`[inline]`

fills the source with the supplied value using DMA channel 3

Parameters:

value	the 32 byte value to fill memory with
dest	the destination to copy to
size	the size in bytes of the area to fill. Will be truncated to the nearest word (4 bytes)

System

ndstypes.h File Reference

Custom types employed by libnds. More...

```
#include <stdint.h>
```

Defines

#define	ALIGN(m) __attribute__((aligned (m)))
	aligns a struct (and other types?) to m, making sure that the size of the struct is a multiple of m.
#define	BIT(n) (1 << (n))
	returns a number with the nth bit set.
#define	PACKED __attribute__ ((packed))
	packs a struct (and other types?) so it won't include padding bytes.

Typedefs

typedef uint8_t	byte
	8 bit unsigned integer.
typedef float	float32
	32 bit signed floating point number.
typedef double	float64
	64 bit signed floating point number.
typedef int16_t	int16
	16 bit signed integer.
typedef int32_t	int32
	32 bit signed integer.
typedef int64_t	int64
	64 bit signed integer.
typedef int8_t	int8
	8 bit signed integer.
typedef int16_t	s16
	16 bit signed integer.
typedef int32_t	s32
	32 bit signed integer.
typedef int64_t	s64
	64 bit signed integer.
typedef int8_t	s8
	8 bit signed integer.
typedef uint16_t	u16
	16 bit unsigned integer.

typedef uint32_t	u32
	32 bit unsigned integer.
typedef uint64_t	u64
	64 bit unsigned integer.
typedef uint8_t	u8
	8 bit unsigned integer.
typedef uint16_t	uint16
	16 bit unsigned integer.
typedef uint32_t	uint32
	32 bit unsigned integer.
typedef uint64_t	uint64
	64 bit unsigned integer.
typedef uint8_t	uint8
	8 bit unsigned integer.
typedef volatile float32	vfloat32
	32 bit volatile signed floating point number.
typedef volatile float64	vfloat64
	64 bit volatile signed floating point number.
typedef volatile int16_t	vint16
	16 bit volatile signed integer.
typedef volatile int32_t	vint32
	32 bit volatile signed integer.
typedef volatile int64_t	vint64
	64 bit volatile signed integer.
typedef volatile int8_t	vint8
	8 bit volatile signed integer.
typedef void(*	VoidFn)(void)
	a function pointer that takes no arguments and doesn't return anything.
typedef volatile s16	vs16
	16 bit volatile signed integer.
typedef volatile s32	vs32
	32 bit volatile signed integer.
typedef volatile s64	vs64
	64 bit volatile signed integer.
typedef volatile s8	vs8

	8 bit volatile signed integer.
typedef volatile u16	vu16
	16 bit volatile unsigned integer.
typedef volatile u32	vu32
	32 bit volatile unsigned integer.
typedef volatile u64	vu64
	64 bit volatile unsigned integer.
typedef volatile u8	vu8
	8 bit volatile unsigned integer.
typedef volatile uint16_t	vuint16
	16 bit volatile unsigned integer.
typedef volatile uint32_t	vuint32
	32 bit volatile unsigned integer.
typedef volatile uint64_t	vuint64
	64 bit volatile unsigned integer.
typedef volatile uint8_t	vuint8
	8 bit volatile unsigned integer.

Enumerations

enum	bool
	C++ compatible bool for C.

system.h File Reference

NDS hardware definitions. These definitions are usually only touched during the initialization of the program. More...

```
#include "ndstypes.h"
```

Data Structures

struct	RTCtime
	struct containing time and day of the real time clock. More...
struct	sysVectors_t
	A struct with all the CPU exeption vectors. each member contains an ARM instuction that will be executed when an exeption occured. More...
struct	tPERSONAL_DATA
	User's DS settings. Defines the structure the DS firmware uses for transfer of the user's settings to the booted program. More...

Defines

#define	HALT_CR (*(vu16*)0x04000300)
	Halt control register.
#define	PersonalData ((PERSONAL_DATA*)0x2FFFC80)
	Default location for the user's personal data (see PERSONAL_DATA).
#define	REG_DISPSTAT (*(vu16*)0x04000004)
	LCD status register.
#define	REG_POWERCNT *(vu16*)0x4000304
	Power control register.
#define	REG_VCOUNT (*(vu16*)0x4000006)
	Current display scanline.

Typedefs

typedef struct sysVectors_t	sysVectors
	A struct with all the CPU exeption vectors. each member contains an ARM instuction that will be executed when an exeption occured.

Enumerations

enum ARM7_power {
POWER_SOUND = (1 << (0)),
PM_CONTROL_REG = 0,
PM_BATTERY_REG = 1,
PM_AMPLIFIER_REG = 2,
PM_READ_REGISTER = (1<<7),
PM_AMP_OFFSET = 2,

PM_GAIN_OFFSET = 3,
PM_BACKLIGHT_LEVEL = 4,
PM_GAIN_20 = 0,
PM_GAIN_40 = 1,
PM_GAIN_80 = 2,
PM_GAIN_160 = 3,
PM_AMP_ON = 1,
PM_AMP_OFF = 0
}
Power-controlled hardware devices accessable to the ARM7.

More...

enum BACKLIGHT_LEVELS {
BACKLIGHT_LOW,
BACKLIGHT_MED,
BACKLIGHT_HIGH,
BACKLIGHT_MAX
}
Backlight level settings. Note, these are only available on DS Lite.

More...

enum DISP_BITS {
DISP_IN_VBLANK = (1 << (0)),
DISP_IN_HBLANK = (1 << (1)),
DISP_YTRIGGERED = (1 << (2)),
DISP_VBLANK_IRQ = (1 << (3)),
DISP_HBLANK_IRQ = (1 << (4)),
DISP_YTRIGGER_IRQ = (1 << (5))
}
LCD Status register bitdefines.

More...

enum PM_Bits {
PM_SOUND_AMP = (1 << (0)),
PM_SOUND_MUTE = (1 << (1)),
PM_BACKLIGHT_BOTTOM = (1 << (2)),
PM_BACKLIGHT_TOP = (1 << (3)),
PM_SYSTEM_PWR = (1 << (6)),
POWER_LCD = (1 << (16)) | (1 << (0)),
POWER_2D_A = (1 << (16)) | (1 << (1)),
POWER_MATRIX = (1 << (16)) | (1 << (2)),
POWER_3D_CORE = (1 << (16)) | (1 << (3)),
POWER_2D_B = (1 << (16)) | (1 << (9)),
POWER_SWAP_LCDS = (1 << (16)) | (1 << (15)),
POWER_ALL_2D = (1 << (16)) | POWER_LCD | POWER_2D_A |
POWER_2D_B,
POWER_ALL = (1 << (16)) | POWER_ALL_2D |
POWER_3D_CORE | POWER_MATRIX
}
Power Management control bits.

More...

Functions

struct tPERSONAL_DATA	__attribute__ ((packed)) PERSONAL_DATA
	User's DS settings. Defines the structure the DS firmware uses for transfer of the user's settings to the booted program.
u32	getBatteryLevel ()
	gets the DS Battery level
static void	lcdMainOnBottom (void)
	Forces the main core to display on the bottom.
static void	lcdMainOnTop (void)
	Forces the main core to display on the top.
static void	lcdSwap (void)
	Switches the screens.
void	ledBlink (int bm)
	Set the LED blink mode.
void *	memCached (void *address)
	returns a cached mirror of an address.
void *	memUncached (void *address)
	returns an uncached mirror of an address.
void	powerOff (int bits)
	Turns off specified hardware.
void	powerOn (int bits)
	Turns on specified hardware.
void	setVectorBase (int highVector)
	Set the arm9 vector base.
static void	SetYtrigger (int Yvalue)
	sets the Y trigger(?)
static void	systemShutDown (void)
	Powers down the DS.
void	systemSleep (void)
	Causes the nds to go to sleep. The nds will be reawakened when the lid is opened.

Variables

u8	alarmHour
	What hour the alarm clock is set to (0-23).
u8	alarmMinute
	What minute the alarm clock is set to (0-59).
u8	birthDay
	The user's birth day (1-31).

u8	birthMonth
	The user's birth month (1-12).
u16	calX1
	Touchscreen calibration: first X touch.
u8	calX1px
	Touchscreen calibration: first X touch pixel.
u16	calX2
	Touchscreen calibration: second X touch.
u8	calX2px
	Touchscreen calibration: second X touch pixel.
u16	calY1
	Touchscreen calibration: first Y touch.
u8	calY1px
	Touchscreen calibration: first X touch pixel.
u16	calY2
	Touchscreen calibration: second Y touch.
u8	calY2px
	Touchscreen calibration: second Y touch pixel.
s16	message [26]
	The user's message.
u16	messageLen
	The length of the user's message in characters.
s16	name [10]
	The user's name in UTF-16 format.
u16	nameLen
	The length of the user's name in characters.
u32	rtcOffset
	Real Time Clock offset.
u8	theme
	The user's theme color (0-15).

Detailed Description

NDS hardware definitions. These definitions are usually only touched during the initialization of the program.

Define Documentation

```
#define HALT_CR   (*(vu16*)0x04000300)
```

Halt control register.

Writing 0x40 to HALT_CR activates GBA mode. HALT_CR can only be accessed via the BIOS.

#define REG_POWERCNT   *(vu16*)0x4000304
Power control register.

This register controls what hardware should be turned on or off.

Typedef Documentation

typedef struct sysVectors_t sysVectors
A struct with all the CPU exeption vectors. each member contains an ARM instuction that will be executed when an exeption occured.

See gbatek for more information.

Enumeration Type Documentation

enum ARM7_power
Power-controlled hardware devices accessable to the ARM7.

Note that these should only be used when programming for the ARM7. Trying to boot up these hardware devices via the ARM9 would lead to unexpected results. ARM7 only.

Enumerator:

POWER_SOUND	Controls the power for the sound controller.
PM_CONTROL_REG	Selects the PM control register.
PM_BATTERY_REG	Selects the PM battery register.
PM_AMPLIFIER_REG	Selects the PM amplifier register.
PM_READ_REGISTER	Selects the PM read register.
PM_AMP_OFFSET	Selects the PM amp register.
PM_GAIN_OFFSET	Selects the PM gain register.
PM_BACKLIGHT_LEVEL	Selects the DS Lite backlight register.
PM_GAIN_20	Sets the mic gain to 20db.
PM_GAIN_40	Sets the mic gain to 40db.
PM_GAIN_80	Sets the mic gain to 80db.
PM_GAIN_160	Sets the mic gain to 160db.
PM_AMP_ON	Turns the sound amp on.
PM_AMP_OFF	Turns the sound amp off.

enum BACKLIGHT_LEVELS

Backlight level settings. Note, these are only available on DS Lite.

Enumerator:

BACKLIGHT_LOW	low backlight setting.
BACKLIGHT_MED	medium backlight setting.
BACKLIGHT_HIGH	high backlight setting.
BACKLIGHT_MAX	max backlight setting.

enum DISP_BITS

LCD Status register bitdefines.

Enumerator:

DISP_IN_VBLANK	The display currently in a vertical blank.
DISP_IN_HBLANK	The display currently in a horizontal blank.
DISP_YTRIGGERED	Current scanline and DISP_Y match.
DISP_VBLANK_IRQ	Interrupt on vertical blank.
DISP_HBLANK_IRQ	Interrupt on horizontal blank.
DISP_YTRIGGER_IRQ	Interrupt when current scanline and DISP_Y match.

enum PM_Bits

Power Management control bits.

Enumerator:

PM_SOUND_AMP	Power the sound hardware (needed to hear stuff in GBA mode too).
PM_SOUND_MUTE	Mute the main speakers, headphone output will still work.
PM_BACKLIGHT_BOTTOM	Enable the top backlight if set.
PM_BACKLIGHT_TOP	Enable the bottom backlight if set.
PM_SYSTEM_PWR	Turn the power *off* if set.
POWER_LCD	Controls the power for both LCD screens.
POWER_2D_A	Controls the power for the main 2D core.
POWER_MATRIX	Controls the power for the 3D matrix.
POWER_3D_CORE	Controls the power for the main 3D core.
POWER_2D_B	Controls the power for the sub 2D core.

POWER_SWAP_LCDS	Controls which screen should use the main core.
POWER_ALL_2D	power just 2D hardware.
POWER_ALL	power everything.

Function Documentation

struct tPERSONAL_DATA __attribute__ ((packed))

User's DS settings. Defines the structure the DS firmware uses for transfer of the user's settings to the booted program.

holds a red green blue triplet

Theme/Color values:

- 0 = Gray
- 1 = Brown
- 2 = Red
- 3 = Pink
- 4 = Orange
- 5 = Yellow
- 6 = Yellow/Green-ish
- 7 = Green
- 8 = Dark Green
- 9 = Green/Blue-ish
- 10 = Light Blue
- 11 = Blue
- 12 = Dark Blue
- 13 = Dark Purple
- 14 = Purple
- 15 = Purple/Red-ish

Language values:

- 0 = Japanese
- 1 = English
- 2 = French
- 3 = German
- 4 = Italian
- 5 = Spanish
- 6 = Chinese(?)
- 7 = Unknown/Reserved

< User's language.

< GBA screen selection (lower screen if set, otherwise upper screen).

< Brightness level at power on, dslite.

< The DS should boot from the DS cart or GBA cart automatically if one is inserted.

< User Settings Lost (0=Normal, 1=Prompt/Settings Lost)

void ledBlink (int *bm*)
Set the LED blink mode.

Arm9 only

Parameters:
    bm What to power on.

void* memCached (void * *address*)
returns a cached mirror of an address.

Parameters:
    address an address.
Returns:
    a pointer to the cached mirror of that address.

void* memUncached (void * *address*)
returns an uncached mirror of an address.

Parameters:
    address an address.
Returns:
    a pointer to the uncached mirror of that address.

void powerOff (int *bits*)
Turns off specified hardware.

May be called from arm7 or arm9 (arm9 power bits will be ignored by arm7, arm7 power bits will be passed to the arm7 from the arm9).

Parameters:
    bits What to power on.

static void powerOn (int *bits*) `[inline]`
Turns on specified hardware.

May be called from arm7 or arm9 (arm9 power bits will be ignored by arm7, arm7 power bits will be passed to the arm7 from the arm9).

Parameters:
    bits What to power on.

void setVectorBase (int *highVector*)
Set the arm9 vector base.

Arm9 only

Parameters:
    highVector high vector

static void SetYtrigger (int *Yvalue*) `[inline, static]`
sets the Y trigger(?)

Parameters:
    Yvalue the value for the Y trigger.
Examples:
    audio/maxmod/streaming/source/main.c.

void systemSleep (void)

Causes the nds to go to sleep. The nds will be reawakened when the lid is opened.

Note:

By default, this is automatically called when closing the lid.

bios.h File Reference

Nintendo DS Bios functions. More...

`#include "nds/ndstypes.h"`

Data Structures

struct	DecompressionStream
	A struct that contains callback function pointers used by the decompression functions. More...
struct	UnpackStruct
	A struct and struct pointer with information about unpacking data. More...

Defines

#define	COPY_MODE_COPY (0)
	copy a range of memory to another piece of memory
#define	COPY_MODE_FILL BIT(24)
	fill a piece of memory with a value.
#define	COPY_MODE_HWORD (0)
	copy in chunks of halfword size.
#define	COPY_MODE_WORD BIT(26)
	copy in chunks of word size.

Typedefs

typedef u8(*	getByteCallback)(u8 *source)
	Should returns a raw byte of the stream.
typedef int(*	getHeaderCallback)(u8 *source, u16 *dest, u32 arg)
	Should return the header of a compressed stream of bytes.
typedef int(*	getResultCallback)(u8 *source)
	Should verify the result after data got decompressed.
typedef struct DecompressionStream	TDecompressionStream
	A struct that contains callback function pointers used by the decompression functions.
typedef struct UnpackStruct	TUnpackStruct
	A struct and struct pointer with information about unpacking data.

Functions

void	swiChangeSoundBias (int enabled, int delay)
	increments or decrements the sound bias once per delay.

void	swiCopy (const void *source, void *dest, int flags)
	copies or fills some memory.
uint16	swiCRC16 (uint16 crc, void *data, uint32 size)
	calculates a CRC-16 checksum.
void	swiDecodeDelta16 (void *source, void *destination)
	Decodes a stream of bytes based on the difference of the bytes.
void	swiDecodeDelta8 (void *source, void *destination)
	Decodes a stream of bytes based on the difference of the bytes.
int	swiDecompressHuffman (void *source, void *destination, uint32 toGetSize, TDecompressionStream *stream)
	Decompresses Huffman compressed data.
int	swiDecompressLZSSVram (void *source, void *destination, uint32 toGetSize, TDecompressionStream *stream)
	Decompresses LZSS compressed data vram safe.
void	swiDecompressLZSSWram (void *source, void *destination)
	Decompresses LZSS compressed data.
int	swiDecompressRLEVram (void *source, void *destination, uint32 toGetSize, TDecompressionStream *stream)
	Decompresses RLE compressed data vram safe.
void	swiDecompressRLEWram (void *source, void *destination)
	Decompresses RLE compressed data.
void	swiDelay (uint32 duration)
	delays the code.
int	swiDivide (int numerator, int divisor)
	divides 2 numbers.
void	swiDivMod (int numerator, int divisor, int *result, int *remainder)
	divides 2 numbers and stores both the result and the remainder.
void	swiFastCopy (const void *source, void *dest, int flags)
	copies or fills some memory. can only copy in word chunks.
uint16	swiGetPitchTable (int index)
	Returns an entry in the pitch table.
uint16	swiGetSineTable (int index)
	Returns an entry in the sine table.
uint8	swiGetVolumeTable (int index)
	Returns an entry in the volume table.
void	swiHalt (void)
	Halts the CPU untill an interupt occures.
int	swiIsDebugger (void)
	returns 0 if running on a nintendo hardware debugger.
int	swiRemainder (int numerator, int divisor)
	calculate the remainder of an division.

void swiSetHaltCR (uint32 data)
Writes a word of the data to 0x04000300:32.

void swiSetHaltCR (uint8 data)
Writes a byte of the data to 0x04000301:8.

void swiSleep (void)
Halts the CPU and most of the hardware untill an interupt occures.

void swiSoftReset (void)
resets the DS.

int swiSqrt (int value)
calculates the square root.

void swiSwitchToGBAMode (void)
Switches the DS to GBA mode.

void swiUnpackBits (uint8 *source, uint32 *destination, PUnpackStruct params)
Unpack data stored in multiple elements in a byte to a larger space.

void swiWaitForIRQ (void)
wait for any interrupt.

Detailed Description

Nintendo DS Bios functions.

See gbatek for more information.

Typedef Documentation

typedef u8(* getByteCallback)(u8 *source)

Should returns a raw byte of the stream.

Parameters:

source	A pointer to the byte.

Returns:

A byte.

typedef int(* getHeaderCallback)(u8 *source, u16 *dest, u32 arg)

Should return the header of a compressed stream of bytes.

The result is a word, with the size of decompressed data in bits 8-31, and bits 0-7 are ignored. This value is also returned by the bios function, unless getResult is non-NULL and returns a negative value. This useally returns the 4 bytes that source points to.

Parameters:

source	A pointer to the compressed data.
dest	A pointer to the space where the decompressed data should be copied to.

arg	A callback value that gets passed to the bios function.

Returns:

The header of the compressed data containing the length of the data and the compression type.

typedef int(* getResultCallback)(u8 *source)

Should verify the result after data got decompressed.

getResult is used to provide a result for the bios function, given the source pointer after all data has been read (or if getSize < 0). Its value is only returned if negative, otherwise the typical result is used, so it is likely some sort of error-checking procedure.

Parameters:

source	The current source address.

Returns:

0 if it went right, or a negative number if something went wrong. value will be returned from bios function if value is negative.

Function Documentation

void swiChangeSoundBias (int *enabled*,
int *delay*
)

increments or decrements the sound bias once per delay.

Parameters:

enabled	0 to decrement it until it reaches 0x000, 1 to increment it until it reaches 0x200.
delay	Is in the same units of time as swiDelay.

Note:

ARM7 exclusive.

void swiCopy (const void * *source*,
void * *dest*,
int *flags*
)

copies or fills some memory.

Parameters:

source	pointer to transfer source or pointer to value to fill the memory with.
dest	pointer to transfer destination.
flags	bits(0-20): size of data to copy/fill in words, or'd with the copy mode size (word or halfword) and type (copy or fill).

Examples:

Graphics/3D/Paletted_Cube/source/main.cpp, and Graphics/Sprites/allocation_test/source/main.c.

uint16 swiCRC16 (uint16 *crc*,

```
                          void *   data,
                          uint32   size
                        )
```
calculates a CRC-16 checksum.

Parameters:

crc	starting CRC-16 value.
data	pointer to data (processed nibble by nibble)
size	size in bytes.

Returns:
the CRC-16 after the data has been processed.

```
void swiDecodeDelta16 ( void *  source,
                        void *  destination
                      )
```
Decodes a stream of bytes based on the difference of the bytes.

Parameters:

source	Pointer to a header word, followed by encoded data. word(31..8) = size of data (in bytes). word(7..0) = ignored.
destination	Destination address.

Note:
Writes data a halfword at a time.
ARM9 exclusive.

```
void swiDecodeDelta8 ( void *  source,
                       void *  destination
                     )
```
Decodes a stream of bytes based on the difference of the bytes.

Parameters:

source	Pointer to a header word, followed by encoded data. word(31..8) = size of data (in bytes). word(7..0) = ignored.
destination	Destination address.

Note:
Writes data a byte at a time.
ARM9 exclusive.

```
int swiDecompressHuffman ( void *                     source,
                           void *                     destination,
                           uint32                     toGetSize,
                           TDecompressionStream *     stream
                         )
```
Decompresses Huffman compressed data.

Parameters:

source	Pointer to source data (always goes through the function pointers, so could just be an offset).
destination	Pointer to destination.

toGetSize	Callback value that is passed to getHeaderCallback function pointer.
stream	Pointer to struct with callback function pointers.

Returns:

The length of the decompressed data, or a signed errorcode from the Open/Close functions.

See also:

decompress.h

int swiDecompressLZSSVram (void * *source*, void * *destination*, uint32 *toGetSize*, TDecompressionStream * *stream*)

Decompresses LZSS compressed data vram safe.

Parameters:

source	Pointer to source data (always goes through the function pointers, so could just be an offset).
destination	Pointer to destination.
toGetSize	Callback value that is passed to getHeaderCallback function pointer.
stream	Pointer to struct with callback function pointers.

Returns:

The length of the decompressed data, or a signed errorcode from the Open/Close functions.

Note:

Writes data a halfword at a time.

See also:

decompress.h

void swiDecompressLZSSWram (void * *source*, void * *destination*)

Decompresses LZSS compressed data.

Parameters:

source	pointer to a header word, followed by compressed data. bit 0-7 of header is ignored. bit 8-31 of header is size of uncompressed data in bytes.
destination	destination address.

Note:

Writes data a byte at a time.

See also:

decompress.h

int swiDecompressRLEVram (void * *source*, void * *destination*,

uint32 *toGetSize*,
TDecompressionStream * *stream*
)

Decompresses RLE compressed data vram safe.

compressed data format: bit(7): 0= uncompressed, 1= compressed. bit(0-6) when uncompressed: run length - 1, followed by run_length bytes of true data. bit(0-6) when compressed: run length - 3, followed by one byte of true data, to be repeated.

Parameters:

source	Pointer to source data (always goes through the function pointers, so could just be an offset).
destination	Pointer to destination.
toGetSize	Callback value that is passed to getHeaderCallback function pointer.
stream	Pointer to struct with callback function pointers.

Returns:

The length of the decompressed data, or a signed errorcode from the Open/Close functions.

Note:

Writes data a halfword at a time.

See also:

decompress.h

void swiDecompressRLEWram (void * *source*,
void * *destination*
)

Decompresses RLE compressed data.

compressed data format: bit(7): 0= uncompressed, 1= compressed. bit(0-6) when uncompressed: run length - 1, followed by run_length bytes of true data. bit(0-6) when compressed: run length - 3, followed by one byte of true data, to be repeated.

Parameters:

source	pointer to a header word, followed by compressed data. bit 0-7 of header is ignored. bit 8-31 of header is size of uncompressed data in bytes.
destination	destination address.

Note:

Writes data a byte at a time.

See also:

decompress.h

void swiDelay (uint32 *duration*)

delays the code.

Delays for for a period X + Y*duration where X is the swi overhead and Y is a cycle of

```
loop:
  sub r0, #1
  bgt loop
```

of thumb fetches in BIOS memory

Parameters:

duration	length of delay

Note:
Duration should be 1 or more, a duration of 0 is a huge delay.

```
int swiDivide ( int numerator,
                int divisor
              )
```

divides 2 numbers.

Parameters:

numerator	signed integer to divide
divisor	signed integer to divide by

Returns:
numerator / divisor

```
void swiDivMod ( int    numerator,
                 int    divisor,
                 int *  result,
                 int *  remainder
               )
```

divides 2 numbers and stores both the result and the remainder.

Parameters:

numerator	signed integer to divide
divisor	signed integer to divide by
result	pointer to integer set to numerator / divisor
remainder	pointer to integer set to numerator % divisor

```
void swiFastCopy ( const void *  source,
                   void *        dest,
                   int           flags
                 )
```

copies or fills some memory. can only copy in word chunks.

Parameters:

source	pointer to transfer source or pointer to value to fill the memory with.
dest	pointer to transfer destination.
flags	bits(0-20): size of data to copy/fill in words, or'd with the type (copy or fill).

Note:
Transfers more quickly than swiCopy, but has higher interrupt latency.

```
uint16 swiGetPitchTable ( int index )
```

Returns an entry in the pitch table.

Parameters:

index	The index of the pitch table (0-767).

Returns:

The entry.

Note:
ARM7 exclusive.

uint16 swiGetSineTable (int *index*)
Returns an entry in the sine table.

Parameters:
index The index of the sine table (0-63).

Returns:
The entry.

Note:
ARM7 exclusive.

uint8 swiGetVolumeTable (int *index*)
Returns an entry in the volume table.

Parameters:
index The index of the volume table (0-723).

Returns:
The entry.

Note:
ARM7 exclusive.

void swiHalt (void)
Halts the CPU untill an interupt occures.

Note:
ARM7 exclusive.

int swiIsDebugger (void)
returns 0 if running on a nintendo hardware debugger.

Returns:
0 if running on a debugger (8 MB of ram instead of 4 MB), else some other number.

int swiRemainder (int *numerator*,
int *divisor*
)
calculate the remainder of an division.

Parameters:

numerator	signed integer to divide
divisor	signed integer to divide by

Returns:
numerator % divisor

void swiSetHaltCR (uint32 *data*)

Writes a word of the data to 0x04000300:32.

Parameters:

data	the word to write.

Note:

This is on the ARM9, but works differently then the ARM7 function!

void swiSetHaltCR (uint8 *data*)

Writes a byte of the data to 0x04000301:8.

Parameters:

data	The byte to write.

Note:

ARM7 exclusive.

void swiSleep (void)

Halts the CPU and most of the hardware untill an interupt occures.

Note:

ARM7 exclusive.

int swiSqrt (int *value*)

calculates the square root.

Parameters:

value	the value to calculate.

Returns:

the square root of the value as an integer.

Note:

use fixed point math if you want more accuracy.

void swiSwitchToGBAMode (void)

Switches the DS to GBA mode.

Note:

ARM7 exclusive.

void swiUnpackBits (uint8 * *source*,
uint32 * *destination*,
PUnpackStruct *params*
)

Unpack data stored in multiple elements in a byte to a larger space.

i.e. 8 elements per byte (i.e. b/w font), into 1 element per byte.

Parameters:

source	Source address.
destination	destination address (word aligned).
params	pointer to an UnpackStruct.

void swiWaitForIRQ (void)

wait for any interrupt.

Note:

ARM9 exclusive.

cache.h File Reference

ARM9 cache control functions. More...

```
#include "nds/ndstypes.h"
```

Functions

void DC_FlushAll ()
  flush the entire data cache to memory.
void DC_FlushRange (const void *base, u32 size)
  flush the data cache for a range of addresses to memory.
void DC_InvalidateAll ()
  invalidate the entire data cache.
void DC_InvalidateRange (const void *base, u32 size)
  invalidate the data cache for a range of addresses.
void IC_InvalidateAll ()
  invalidate entire instruction cache.
void IC_InvalidateRange (const void *base, u32 size)
  invalidate the instruction cache for a range of addresses.

Detailed Description

ARM9 cache control functions.

Function Documentation

DC_FlushRange (const void * *base*,
u32 *size*
)

flush the data cache for a range of addresses to memory.

Parameters:

base	base address of the region to flush.
size	size of the region to flush.

Examples:
capture/ScreenShot/source/main.cpp.

DC_InvalidateRange (const void * *base*,
u32 *size*
)

invalidate the data cache for a range of addresses.

Parameters:

base	base address of the region to invalidate

size	size of the region to invalidate.

Examples:

audio/micrecord/source/micrecord.c.

IC_InvalidateRange	(	const void *	*base*,
		u32	*size*
	)		

invalidate the instruction cache for a range of addresses.

Parameters:

base	base address of the region to invalidate
size	size of the region to invalidate.

interrupts.h File Reference

nds interrupt support. More...

`#include <nds/ndstypes.h>`

Defines

#define	IRQ_TIMER(n) (1 << ((n) + 3))
	returns the mask for a given timer.
#define	MAX_INTERRUPTS 25
	maximum number of interrupts.
#define	REG_IE (*(vuint32*)0x04000210)
	Interrupt Enable Register.
#define	REG_IF (*(vuint32*)0x04000214)
	Interrupt Flag Register.
#define	REG_IME (*(vuint32*)0x04000208)
	Interrupt Master Enable Register.

Enumerations

```
enum  IME_VALUE {
        IME_DISABLE = 0,
        IME_ENABLE = 1
      }
      values allowed for REG_IME

      More...
enum  IRQ_MASKS {
        IRQ_VBLANK = BIT(0),
        IRQ_HBLANK = BIT(1),
        IRQ_VCOUNT = BIT(2),
        IRQ_TIMER0 = BIT(3),
        IRQ_TIMER1 = BIT(4),
        IRQ_TIMER2 = BIT(5),
        IRQ_TIMER3 = BIT(6),
        IRQ_NETWORK = BIT(7),
        IRQ_DMA0 = BIT(8),
        IRQ_DMA1 = BIT(9),
        IRQ_DMA2 = BIT(10),
        IRQ_DMA3 = BIT(11),
        IRQ_KEYS = BIT(12),
        IRQ_CART = BIT(13),
        IRQ_IPC_SYNC = BIT(16),
        IRQ_FIFO_EMPTY = BIT(17),
        IRQ_FIFO_NOT_EMPTY = BIT(18),
        IRQ_CARD = BIT(19),
        IRQ_CARD_LINE = BIT(20),
        IRQ_GEOMETRY_FIFO = BIT(21),
```

IRQ_LID = BIT(22),
IRQ_SPI = BIT(23),
IRQ_WIFI = BIT(24),
IRQ_ALL = (~0)
}
values allowed for REG_IE and REG_IF

More...

enum	IRQ_MASKSAUX { IRQ_I2C = BIT(6) } values allowed for REG_AUXIE and REG_AUXIF More...

Functions

void	irqClear (u32 irq) remove the handler associated with the interrupt mask irq.
void	irqDisable (u32 irq) Prevent the given interrupt from occuring.
void	irqEnable (u32 irq) Allow the given interrupt to occur.
void	irqInit () Initialise the libnds interrupt system.
void	irqInitHandler (VoidFn handler) Install a user interrupt dispatcher.
void	irqSet (u32 irq, VoidFn handler) Add a handler for the given interrupt mask.
VoidFn	setPowerButtonCB (VoidFn CB) set callback for DSi Powerbutton press
void	swiIntrWait (u32 waitForSet, uint32 flags) wait for interrupt(s) to occur
void	swiWaitForVBlank (void) Wait for vblank interrupt.

Detailed Description

nds interrupt support.

Define Documentation

#define IRQ_TIMER (n) (1 << ((n) + 3))

returns the mask for a given timer.

Parameters:

n the timer.

Returns:
the bitmask.

#define REG_IE   (*(vuint32*)0x04000210)
Interrupt Enable Register.

This is the activation mask for the internal interrupts. Unless the corresponding bit is set, the IRQ will be masked out.

#define REG_IF   (*(vuint32*)0x04000214)
Interrupt Flag Register.

Since there is only one hardware interrupt vector, the IF register contains flags to indicate when a particular of interrupt has occured. To acknowledge processing interrupts, set IF to the value of the interrupt handled.

#define REG_IME   (*(vuint32*)0x04000208)
Interrupt Master Enable Register.

When bit 0 is clear, all interrupts are masked. When it is 1, interrupts will occur if not masked out in REG_IE.

Enumeration Type Documentation

enum IME_VALUE
values allowed for REG_IME

Enumerator:

IME_DISABLE	Disable all interrupts.
IME_ENABLE	Enable all interrupts not masked out in REG_IE

enum IRQ_MASKS
values allowed for REG_IE and REG_IF

Enumerator:

IRQ_VBLANK	vertical blank interrupt mask
IRQ_HBLANK	horizontal blank interrupt mask
IRQ_VCOUNT	vcount match interrupt mask
IRQ_TIMER0	timer 0 interrupt mask
IRQ_TIMER1	timer 1 interrupt mask
IRQ_TIMER2	timer 2 interrupt mask
IRQ_TIMER3	timer 3 interrupt mask
IRQ_NETWORK	serial interrupt mask

IRQ_DMA0	DMA 0 interrupt mask
IRQ_DMA1	DMA 1 interrupt mask
IRQ_DMA2	DMA 2 interrupt mask
IRQ_DMA3	DMA 3 interrupt mask
IRQ_KEYS	Keypad interrupt mask
IRQ_CART	GBA cartridge interrupt mask
IRQ_IPC_SYNC	IPC sync interrupt mask
IRQ_FIFO_EMPTY	Send FIFO empty interrupt mask
IRQ_FIFO_NOT_EMPTY	Receive FIFO not empty interrupt mask
IRQ_CARD	interrupt mask DS Card Slot
IRQ_CARD_LINE	interrupt mask
IRQ_GEOMETRY_FIFO	geometry FIFO interrupt mask
IRQ_LID	interrupt mask DS hinge
IRQ_SPI	SPI interrupt mask
IRQ_WIFI	WIFI interrupt mask (ARM7)
IRQ_ALL	'mask' for all interrupt

enum IRQ_MASKSAUX
values allowed for REG_AUXIE and REG_AUXIF

Enumerator:

IRQ_I2C	I2C interrupt mask (DSi ARM7)

Function Documentation

irqClear (u32 *irq*)
remove the handler associated with the interrupt mask irq.

Parameters:

irq Mask associated with the interrupt.

irqDisable (u32 *irq*)
Prevent the given interrupt from occuring.

Parameters:

irq The set of interrupt masks to disable.

Note:

Specify multiple interrupts to disable by ORing several IRQ_MASKS.

irqEnable (u32 *irq*)

Allow the given interrupt to occur.

Parameters:

irq The set of interrupt masks to enable.

Note:

Specify multiple interrupts to enable by ORing several IRQ_MASKS.

Examples:

audio/maxmod/streaming/source/main.c.

irqInit ( )

Initialise the libnds interrupt system.

This function is called internally (prior to main()) to set up irqs on the ARM9. It must be called on the ARM7 prior to installing irq handlers.

irqInitHandler (VoidFn *handler*)

Install a user interrupt dispatcher.

This function installs the main interrupt function, all interrupts are serviced through this routine. For most purposes the libnds interrupt dispacther should be used in preference to user code unless you know *exactly* what you're doing.

Parameters:

handler Address of the function to use as an interrupt dispatcher

Note:

the function *must* be ARM code

irqSet (u32 *irq*,
VoidFn *handler*
)

Add a handler for the given interrupt mask.

Specify the handler to use for the given interrupt. This only works with the default interrupt handler, do not mix the use of this routine with a user-installed IRQ handler.

Parameters:

irq	Mask associated with the interrupt.
handler	Address of the function to use as an interrupt service routine

Note:

When any handler specifies using IRQ_VBLANK or IRQ_HBLANK, DISP_SR is automatically updated to include the corresponding DISP_VBLANK_IRQ or DISP_HBLANK_IRQ.

Warning:

Only one IRQ_MASK can be specified with this function.

Examples:

hello_world/source/main.cpp.

VoidFn setPowerButtonCB (VoidFn *CB*)
set callback for DSi Powerbutton press

Parameters:

CB	function to call when power button pressed

Returns:
the previously set callback

swiIntrWait (u32 *waitForSet*,
uint32 *flags*
)
wait for interrupt(s) to occur

Parameters:

waitForSet	0: Return if the interrupt has already occured 1: Wait until the interrupt has been set since the call
flags	interrupt mask to wait for

Examples:
audio/maxmod/streaming/source/main.c.

swiWaitForVBlank (void)
Wait for vblank interrupt.

Waits for a vertical blank interrupt

Note:
Identical to calling swiIntrWait(1, 1)

Examples:
audio/maxmod/audio_modes/source/main.c, audio/maxmod/basic_sound/source/MaxModExample.c, audio/maxmod/reverb/source/main.c, audio/maxmod/song_events_example/source/template.c, audio/maxmod/song_events_example2/source/template.c, audio/maxmod/streaming/source/main.c, audio/micrecord/source/micrecord.c, capture/ScreenShot/source/main.cpp, card/eeprom/source/main.cpp, debugging/exceptionTest/source/exceptionTest.c, ds_motion/source/main.c, dswifi/ap_search/source/template.c, dswifi/autoconnect/source/autoconnect.c, dswifi/httpget/source/httpget.c, filesystem/libfat/libfatdir/source/directory.c, filesystem/nitrofs/nitrodir/source/directory.c, Graphics/3D/3D_Both_Screens/source/template.c, Graphics/3D/BoxTest/source/main.cpp, Graphics/3D/nehe/lesson01/source/nehe1.cpp, Graphics/3D/nehe/lesson02/source/nehe2.cpp, Graphics/3D/nehe/lesson03/source/nehe3.cpp, Graphics/3D/nehe/lesson04/source/nehe4.cpp, Graphics/3D/nehe/lesson05/source/nehe5.cpp, Graphics/3D/nehe/lesson06/source/nehe6.cpp, Graphics/3D/nehe/lesson07/source/nehe7.cpp, Graphics/3D/nehe/lesson08/source/nehe8.cpp, Graphics/3D/nehe/lesson09/source/nehe9.cpp, Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/3D/nehe/lesson10b/source/nehe10b.cpp,

Graphics/3D/nehe/lesson11/source/nehe11.cpp, Graphics/3D/Paletted_Cube/source/main.cpp, Graphics/3D/Simple_Quad/source/main.cpp, Graphics/3D/Simple_Tri/source/main.cpp, Graphics/3D/Textured_Cube/source/main.cpp, Graphics/3D/Textured_Quad/source/main.cpp, Graphics/3D/Toon_Shading/source/main.cpp, Graphics/Backgrounds/16bit_color_bmp/source/template.cpp, Graphics/Backgrounds/256_color_bmp/source/main.cpp, Graphics/Backgrounds/all_in_one/source/advanced.cpp, Graphics/Backgrounds/all_in_one/source/main.cpp, Graphics/Backgrounds/all_in_one/source/scrolling.cpp, Graphics/Backgrounds/Double_Buffer/source/main.cpp, Graphics/Backgrounds/rotation/source/main.cpp, Graphics/Printing/ansi_console/source/main.c, Graphics/Printing/console_windows/source/main.c, Graphics/Printing/custom_font/source/main.c, Graphics/Printing/print_both_screens/source/template.c, Graphics/Printing/rotscale_text/source/main.c, Graphics/Sprites/allocation_test/source/main.c, Graphics/Sprites/animate_simple/source/template.c, Graphics/Sprites/bitmap_sprites/source/main.cpp, Graphics/Sprites/fire_and_sprites/source/main.cpp, Graphics/Sprites/simple/source/template.c, Graphics/Sprites/sprite_extended_palettes/source/template.c, Graphics/Sprites/sprite_rotate/source/template.c, hello_world/source/main.cpp, input/keyboard/keyboard_async/source/template.c, input/keyboard/keyboard_stdin/source/keymain.c, input/Touch_Pad/touch_area/source/template.c, input/Touch_Pad/touch_test/source/main.c, time/RealTimeClock/source/main.c, time/stopwatch/source/main.c, and time/timercallback/source/main.c.

fifocommon.h File Reference

low level FIFO API. More...

```
#include "ndstypes.h"
```

Typedefs

typedef void(* FifoAddressHandlerFunc)(void *address, void *userdata)
: fifo callback function pointer with the sent address and the callback's user data.

typedef void(* FifoDatamsgHandlerFunc)(int num_bytes, void *userdata)
: fifo callback function pointer with the number of bytes sent and the callback's user data

typedef void(* FifoValue32HandlerFunc)(u32 value32, void *userdata)
: fifo callback function pointer with the sent value and the callback's user data.

Enumerations

```
enum  FifoChannels {
        FIFO_PM = 0,
        FIFO_SOUND = 1,
        FIFO_SYSTEM = 2,
        FIFO_MAXMOD = 3,
        FIFO_DSWIFI = 4,
        FIFO_RSVD_01 = 5,
        FIFO_RSVD_02 = 6,
        FIFO_RSVD_03 = 7,
        FIFO_USER_01 = 8,
        FIFO_USER_02 = 9,
        FIFO_USER_03 = 10,
        FIFO_USER_04 = 11,
        FIFO_USER_05 = 12,
        FIFO_USER_06 = 13,
        FIFO_USER_07 = 14,
        FIFO_USER_08 = 15
      }
```
Enum values for the different fifo channels.

More...

enum FifoPMCommands
: Enum values for the fifo power management commands.

enum FifoSoundCommand

Enum values for the fifo sound commands.

enum FifoSystemCommands
Enum values for the fifo system commands.

enum FifoWifiCommands
Enum values for the fifo wifi commands.

```
enum PM_LedBlinkMode {
  PM_LED_ON = (0<<4),
  PM_LED_SLEEP = (1<<4),
  PM_LED_BLINK = (3<<4)
}
```
Power Management LED blink mode control bits.

More...

Functions

bool	fifoCheckAddress (int channel) checks if there is any addresses in the fifo queue.
bool	fifoCheckDatamsg (int channel) checks if there is any data messages in the fifo queue.
int	fifoCheckDatamsgLength (int channel) gets the number of bytes in the queue for the first data entry.
bool	fifoCheckValue32 (int channel) checks if there is any values in the fifo queue.
void *	fifoGetAddress (int channel) Get the first address in queue for a specific channel.
int	fifoGetDatamsg (int channel, int buffersize, u8 *destbuffer) Reads a data message in a given buffer and returns the number of bytes written.
u32	fifoGetValue32 (int channel) Get the first value32 in queue for a specific channel.
bool	fifoInit () Initializes the fifo system.
bool	fifoSendAddress (int channel, void *address) Send an address to an channel.
bool	fifoSendDatamsg (int channel, int num_bytes, u8 *data_array) Send a sequence of bytes to the other CPU.
bool	fifoSendValue32 (int channel, u32 value32) Send a 32bit value.
bool	fifoSetAddressHandler (int channel, FifoAddressHandlerFunc newhandler,

void *userdata)

Set user address message callback.

bool fifoSetDatamsgHandler (int channel, FifoDatamsgHandlerFunc newhandler, void *userdata)

Set user data message callback.

bool fifoSetValue32Handler (int channel, FifoValue32HandlerFunc newhandler, void *userdata)

Set user value32 message callback.

Detailed Description

low level FIFO API.

Typedef Documentation

typedef void(* FifoAddressHandlerFunc)(void *address, void *userdata)

fifo callback function pointer with the sent address and the callback's user data.

The handler is called when new data arrives.

Note:

callback functions are called from interrupt level, but are well secured. not too much caution is necessary, but don't call alloc, free or printf from within them, just to be safe.

typedef void(* FifoDatamsgHandlerFunc)(int num_bytes, void *userdata)

fifo callback function pointer with the number of bytes sent and the callback's user data

The handler is called when new data arrives. This callback must call fifoGetData to actually retrieve the data. If it doesn't, the data will be destroyed on return.

Note:

callback functions are called from interrupt level, but are well secured. not too much caution is necessary, but don't call alloc, free or printf from within them, just to be safe.

typedef void(* FifoValue32HandlerFunc)(u32 value32, void *userdata)

fifo callback function pointer with the sent value and the callback's user data.

The handler is called when new data arrives.

Note:

callback functions are called from interrupt level, but are well secured. not too much caution is necessary, but don't call alloc, free or printf from within them, just to be safe.

Enumeration Type Documentation

enum FifoChannels

Enum values for the different fifo channels.

Enumerator:

FIFO_PM	fifo channel reserved for power management.
FIFO_SOUND	fifo channel reserved for sound access.
FIFO_SYSTEM	fifo channel reserved for system functions.
FIFO_MAXMOD	fifo channel reserved for the maxmod library.
FIFO_DSWIFI	fifo channel reserved for the dswifi library.
FIFO_RSVD_01	fifo channel reserved for future use.
FIFO_RSVD_02	fifo channel reserved for future use.
FIFO_RSVD_03	fifo channel reserved for future use.
FIFO_USER_01	fifo channel available for users.
FIFO_USER_02	fifo channel available for users.
FIFO_USER_03	fifo channel available for users.
FIFO_USER_04	fifo channel available for users.
FIFO_USER_05	fifo channel available for users.
FIFO_USER_06	fifo channel available for users.
FIFO_USER_07	fifo channel available for users.
FIFO_USER_08	fifo channel available for users.

enum PM_LedBlinkMode

Power Management LED blink mode control bits.

Enumerator:

PM_LED_ON	Steady on.
PM_LED_SLEEP	Blinking, mostly off.
PM_LED_BLINK	Blinking, mostly on.

Function Documentation

bool fifoCheckAddress (int *channel*)
checks if there is any addresses in the fifo queue.

Parameters:
channel the channel to check.

Returns:
true if there is any addresses in the queue and if there isn't an address handler in place for the channel.

bool fifoCheckDatamsg (int *channel*)
checks if there is any data messages in the fifo queue.

Parameters:
channel the channel to check.

Returns:
true if there is any data messages in the queue and if there isn't a data message handler in place for the channel.

int fifoCheckDatamsgLength (int *channel*)
gets the number of bytes in the queue for the first data entry.

Parameters:
channel the channel to check.

Returns:
the number of bytes in the queue for the first data entry, or -1 if there are no entries.

bool fifoCheckValue32 (int *channel*)
checks if there is any values in the fifo queue.

Parameters:
channel the channel to check.

Returns:
true if there is any values in the queue and if there isn't a value handler in place for the channel.

void* fifoGetAddress (int *channel*)
Get the first address in queue for a specific channel.

Parameters:
channel the channel to check.

Returns:
the first address in queue, or NULL if there is none.

```
int fifoGetDatamsg ( int   channel,
                     int   buffersize,
                     u8 *  destbuffer
                   )
```
Reads a data message in a given buffer and returns the number of bytes written.

Parameters:
channel the channel to check.

buffersize	the size of the buffer where the message will be copied to.
destbuffer	a pointer to the buffer where the message will be copied to.

Returns:
the number of bytes written, or -1 if there is no message.

Warning:
If your buffer is not big enough, you may lose data! Check the data length first if you're not sure what the size is.

u32 fifoGetValue32 (int *channel*)

Get the first value32 in queue for a specific channel.

Parameters:

channel	the channel to check.

Returns:
the first value32 in queue, or 0 if there is no message.

bool fifoInit ( )

Initializes the fifo system.

Attempts to sync with the other CPU, if it fails, fifo services won't be provided.

Note:
call irqInit() before calling this function.

Returns:
true if syncing worked, false if something went wrong.

```
bool fifoSendAddress  ( int      channel,
                        void *   address
                      )
```

Send an address to an channel.

Transmits an address in the range 0x02000000-0x023FFFFF to the other CPU.

Parameters:

channel	channel number to send to.
address	address to send.

Returns:
true if the address has been send, false if something went wrong.

```
bool fifoSendDatamsg  ( int    channel,
                        int    num_bytes,
                        u8 *   data_array
                      )
```

Send a sequence of bytes to the other CPU.

num_bytes can be between 0 and FIFO_MAX_DATA_BYTES - sending 0 bytes can be useful sometimes...

Parameters:

channel	channel number to send to
num_bytes	number of bytes to send
data_array	pointer to data array

Returns:

true if the data message has been send, false if something went wrong.

bool fifoSendValue32 (int *channel*,
u32 *value32*
)

Send a 32bit value.

Transmits a 32bit value to the other CPU.

Parameters:

channel	channel number to send to
value32	32bit value to send

Returns:

true if the value has been send, false if something went wrong.

Note:

Transfer is more efficient if the top 8 bits are zero. So sending smaller values or bitmasks that don't include the top bits is preferred.

bool fifoSetAddressHandler (int *channel*,
FifoAddressHandlerFunc *newhandler*,
void * *userdata*
)

Set user address message callback.

Set a callback to receive incoming address messages on a specific channel.

Parameters:

channel	channel number to send to.
newhandler	a function pointer to the new handler function.
userdata	a pointer that will be passed on to the handler when it will be called.

Returns:

true if the handler has been set, false if something went wrong.

Note:

Setting the handler for a channel feeds the queue of buffered messages to the new handler, if there are any unread messages.

bool fifoSetDatamsgHandler (int *channel*,
FifoDatamsgHandlerFunc *newhandler*,
void * *userdata*
)

Set user data message callback.

Set a callback to receive incoming data messages on a specific channel.

Parameters:

channel	channel number to send to.
newhandler	a function pointer to the new handler function.
userdata	a pointer that will be passed on to the handler when it will be called.

Returns:

true if the handler has been set, false if something went wrong.

Note:

Setting the handler for a channel feeds the queue of buffered messages to the new handler, if there are any unread messages.

bool fifoSetValue32Handler	(	int	*channel*,
		FifoValue32HandlerFunc	*newhandler*,
		void *	*userdata*
	)		

Set user value32 message callback.

Set a callback to receive incoming value32 messages on a specific channel.

Parameters:

channel	channel number to send to.
newhandler	a function pointer to the new handler function.
userdata	a pointer that will be passed on to the handler when it will be called.

Returns:

true if the handler has been set, false if something went wrong.

Note:

Setting the handler for a channel feeds the queue of buffered messages to the new handler, if there are any unread messages.

timers.h File Reference

Contains defines, macros and functions for ARM7 and ARM9 timer operation. It also contains a simplified API for timer use and some cpu timing functions. More...

```
#include <nds/ndstypes.h>
```

Defines

#define BUS_CLOCK (33513982)
the speed in which the timer ticks in hertz.

#define TIMER0_CR (*(vu16*)0x04000102)
Same as TIMER_CR(0).

#define TIMER0_DATA (*(vu16*)0x04000100)
Same as TIMER_DATA(0).

#define TIMER1_CR (*(vu16*)0x04000106)
Same as TIMER_CR(1).

#define TIMER1_DATA (*(vu16*)0x04000104)
Same as TIMER_DATA(1).

#define TIMER2_CR (*(vu16*)0x0400010A)
Same as TIMER_CR(2).

#define TIMER2_DATA (*(vu16*)0x04000108)
Same as TIMER_DATA(2).

#define TIMER3_CR (*(vu16*)0x0400010E)
Same as TIMER_CR(3).

#define TIMER3_DATA (*(vu16*)0x0400010C)
Same as TIMER_DATA(3).

#define TIMER_CASCADE (1<<2)
When set will cause the timer to count when the timer below overflows (unavailable for timer 0).

#define TIMER_CR(n) (*(vu16*)(0x04000102+((n)<<2)))
Returns a dereferenced pointer to the data register for timer control Register.

#define TIMER_DATA(n) (*(vu16*)(0x04000100+((n)<<2)))
Returns a dereferenced pointer to the data register for timer number "n".

#define TIMER_ENABLE (1<<7)
Enables the timer.

#define TIMER_FREQ(n) (-BUS_CLOCK/(n))
A macro that calculates TIMER_DATA(n) settings for a given frequency of n. will calculate the correct value for TIMER_DATA(n) given the frequency in hertz (number of times the timer should overflow per second).

#define TIMER_FREQ_1024(n) (-(BUS_CLOCK>>10)/(n))
A macro that calculates TIMER_DATA(n) settings for a given frequency of n. will calculate the correct value for TIMER_DATA(n) given the frequency in hertz (number of times the timer should overflow per second).

#define TIMER_FREQ_256(n) (-(BUS_CLOCK>>8)/(n))

A macro that calculates TIMER_DATA(n) settings for a given frequency of n. will calculate the correct value for TIMER_DATA(n) given the frequency in hertz (number of times the timer should overflow per second).

#define TIMER_FREQ_64(n) (-(BUS_CLOCK>>6)/(n))

A macro that calculates TIMER_DATA(n) settings for a given frequency of n. will calculate the correct value for TIMER_DATA(n) given the frequency in hertz (number of times the timer should overflow per second).

#define TIMER_IRQ_REQ (1<<6)

Causes the timer to request an Interupt on overflow.

Enumerations

```
enum ClockDivider {
  ClockDivider_1 = 0,
  ClockDivider_64 = 1,
  ClockDivider_256 = 2,
  ClockDivider_1024 = 3
}
```

allowable timer clock dividers.

More...

Functions

u32 cpuEndTiming ()

ends cpu Timing.

u32 cpuGetTiming ()

returns the number of ticks which have elapsed since cpuStartTiming.

void cpuStartTiming (int timer)

begins cpu Timing using two timers for 32bit resolution.

u16 timerElapsed (int timer)

returns the ticks elapsed since the last call to timerElapsed().

u16 timerPause (int timer)

pauses the specified timer.

void timerStart (int timer, ClockDivider divider, u16 ticks, VoidFn callback)

start a hardware timer. Callback is tied directly to interupt table and called directly resulting in less latency than the attached timer.

u16 timerStop (int timer)

Stops the specified timer.

static u16 timerTick (int timer)

returns the raw ticks of the specified timer.

static void timerUnpause (int timer)

unpauses the specified timer.

Detailed Description

Contains defines, macros and functions for ARM7 and ARM9 timer operation. It also contains a simplified API for timer use and some cpu timing functions.

The timers are fed with a 33.513982 MHz source on the ARM9 and ARM7.

Note:
 that dswifi will use timer 3 on the arm9, so don't use that if you use dswifi.

Define Documentation

#define TIMER_CR (n) (*(vu16*)(0x04000102+((n)<<2)))
Returns a dereferenced pointer to the data register for timer control Register.

Example Usage: TIMER_CR(x) = TIMER_ENABLE | ClockDivider_64;

Possible bit defines:

See also:
 TIMER_ENABLE
 TIMER_IRQ_REQ
 TIMER_CASCADE
 ClockDivider

Examples:
 Graphics/3D/BoxTest/source/main.cpp.

#define TIMER_DATA (n) (*(vu16*)(0x04000100+((n)<<2)))
Returns a dereferenced pointer to the data register for timer number "n".

See also:
 TIMER_CR(n)
 TIMER_FREQ(n)

TIMER_DATA(n) when set will latch that value into the counter. Everytime the counter rolls over TIMER_DATA(0) will return to the latched value. This allows you to control the frequency of the timer using the following formula:
TIMER_DATA(x) = -(BUS_CLOCK/(freq * divider));

Example Usage: TIMER_DATA(0) = value; were 0 can be 0 through 3 and value is 16 bits.

Examples:
 Graphics/3D/BoxTest/source/main.cpp.

#define TIMER_FREQ (n) (-BUS_CLOCK/(n))
A macro that calculates TIMER_DATA(n) settings for a given frequency of n. will calculate the correct value for TIMER_DATA(n) given the frequency in hertz (number of times the timer should overflow per second).

Example Usage:

```
calls the timerCallBack function 5 times per second.
                 timerStart(0, ClockDivider_1024, TIMER_FREQ_1024(5),
timerCallBack);
```

Max frequency is: 33554432Hz Min frequency is: 512Hz

Note:

Use the appropriate macro depending on the used clock divider.

#define TIMER_FREQ_1024 (n) (-(BUS_CLOCK>>10)/(n))

A macro that calculates TIMER_DATA(n) settings for a given frequency of n. will calculate the correct value for TIMER_DATA(n) given the frequency in hertz (number of times the timer should overflow per second).

Example Usage:

```
calls the timerCallBack function 5 times per second.
                timerStart(0, ClockDivider_1024, TIMER_FREQ_1024(5),
timerCallBack);
```

Max frequency is: 32768Hz Min frequency is: 0.5Hz

Note:

Use the appropriate macro depending on the used clock divider.

Examples:

time/timercallback/source/main.c.

#define TIMER_FREQ_256 (n) (-(BUS_CLOCK>>8)/(n))

A macro that calculates TIMER_DATA(n) settings for a given frequency of n. will calculate the correct value for TIMER_DATA(n) given the frequency in hertz (number of times the timer should overflow per second).

Example Usage:

```
calls the timerCallBack function 5 times per second.
                timerStart(0, ClockDivider_1024, TIMER_FREQ_1024(5),
timerCallBack);
```

Max frequency is: 131072Hz Min frequency is: 2Hz

Note:

Use the appropriate macro depending on the used clock divider.

#define TIMER_FREQ_64 (n) (-(BUS_CLOCK>>6)/(n))

A macro that calculates TIMER_DATA(n) settings for a given frequency of n. will calculate the correct value for TIMER_DATA(n) given the frequency in hertz (number of times the timer should overflow per second).

Example Usage:

```
calls the timerCallBack function 5 times per second.
                timerStart(0, ClockDivider_1024, TIMER_FREQ_1024(5),
timerCallBack);
```

Max frequency is: 524288Hz Min frequency is: 8Hz

Note:

Use the appropriate macro depending on the used clock divider.

Enumeration Type Documentation

enum ClockDivider

allowable timer clock dividers.

Enumerator:

ClockDivider_1	divides the timer clock by 1 (~33513.982 kHz)
ClockDivider_64	divides the timer clock by 64 (~523.657 kHz)
ClockDivider_256	divides the timer clock by 256 (~130.914 kHz)
ClockDivider_1024	divides the timer clock by 1024 (~32.7284 kHz)

Function Documentation

u32 cpuEndTiming ( )

ends cpu Timing.

Returns:

The number of ticks which have elapsed since cpuStartTiming.

u32 cpuGetTiming ( )

returns the number of ticks which have elapsed since cpuStartTiming.

Returns:

The number of ticks which have elapsed since cpuStartTiming.

void cpuStartTiming (int *timer*)

begins cpu Timing using two timers for 32bit resolution.

Parameters:

timer The base hardware timer to use (0 - 2).

u16 timerElapsed (int *timer*)

returns the ticks elapsed since the last call to timerElapsed().

Parameters:

timer The hardware timer to use (0 - 3).

Returns:

The number of ticks which have elapsed since the last callto timerElapsed().

Examples:

time/stopwatch/source/main.c.

u16 timerPause (int *timer*)
pauses the specified timer.

Parameters:

timer	The hardware timer to use (0 - 3).

Returns:
The number of ticks which have elapsed since the last callto timerElapsed().

Examples:
time/stopwatch/source/main.c.

void timerStart (int *timer*, ClockDivider *divider*, u16 *ticks*, VoidFn *callback*)
start a hardware timer. Callback is tied directly to interupt table and called directly resulting in less latency than the attached timer.

Parameters:

timer	The hardware timer to use (0 - 3).
divider	The timer channel clock divider (clock will tick at 33.513982 Mhz / divider)
ticks	The number of ticks which must elapse before the timer overflows
callback	The callback to be called when the timer expires (if null no irq will be generated by the timer)

Examples:
time/stopwatch/source/main.c, and time/timercallback/source/main.c.

u16 timerStop (int *timer*)
Stops the specified timer.

Parameters:

timer	The hardware timer to use (0 - 3).

Returns:
The number of ticks which have elapsed since the last callto timerElapsed().

Examples:
time/stopwatch/source/main.c.

static u16 timerTick (int *timer*) `[inline, static]`
returns the raw ticks of the specified timer.

Parameters:

timer	The hardware timer to use (0 - 3).

Returns:
the raw ticks of the specified timer data register.

static void timerUnpause (int *timer*) `[inline, static]`
unpauses the specified timer.

Parameters:

timer The hardware timer to use (0 - 3).

Examples:

time/stopwatch/source/main.c.

User Input/ouput

arm9/input.h File Reference

NDS button and touchscreen input support. More...

```
#include <nds/touch.h>
#include <nds/input.h>
```

Functions

uint32	keysCurrent (void)
	Obtains the current keypad state. Call this function to get keypad state without affecting state of other key functions (keysUp keysHeld etc...)
uint32	keysDown (void)
	Obtains the current keypad pressed state.
uint32	keysDownRepeat (void)
	Obtains the current keypad pressed or repeating state.
uint32	keysHeld (void)
	Obtains the current keypad held state.
void	keysSetRepeat (u8 setDelay, u8 setRepeat)
	Sets the key repeat parameters.
uint32	keysUp (void)
	Obtains the current keypad released state.
void	scanKeys (void)
	Obtains the current keypad state. Call this function once per main loop in order to use the keypad functions.
void	touchRead (touchPosition *data)
	Obtains the current touchpad state.

Detailed Description

NDS button and touchscreen input support.

Function Documentation

void keysSetRepeat (u8 *setDelay*,
                     u8 *setRepeat*
)

Sets the key repeat parameters.

Parameters:

setDelay	Number of scanKeys calls before keys start to repeat.
setRepeat	Number of scanKeys calls before keys repeat.

void touchRead (touchPosition * *data*)

Obtains the current touchpad state.

Parameters:

data a touchPosition ptr which will be filled by the function.

Examples:

Graphics/3D/BoxTest/source/main.cpp, Graphics/3D/Env_Mapping/source/main.cpp, Graphics/3D/Picking/source/main.cpp, Graphics/3D/Toon_Shading/source/main.cpp, Graphics/Printing/console_windows/source/main.c, Graphics/Printing/print_both_screens/source/template.c, Graphics/Sprites/simple/source/template.c, Graphics/Sprites/sprite_extended_palettes/source/template.c, hello_world/source/main.cpp, input/Touch_Pad/touch_area/source/template.c, input/Touch_Pad/touch_look/source/main.cpp, and input/Touch_Pad/touch_test/source/main.c.

keyboard.h File Reference

nds stdio keyboard integration. More...

```
#include <nds/ndstypes.h>
#include <nds/arm9/background.h>
```

Data Structures

struct	Keyboard describes a keyboard. More...
struct	KeyMap defines a key mapping. More...

Typedefs

typedef struct Keyboard	Keyboard describes a keyboard.
typedef void(*	KeyChangeCallback)(int key) callback function pointer for a key changed.
typedef struct KeyMap	KeyMap defines a key mapping.

Enumerations

```
enum  KeyboardState {
        Lower = 0,
        Upper = 1,
        Numeric = 2,
        Reduced = 3
      }
```

States the keyboard can be in, currently only Lower and Upper supported.

More...

```
enum  Keys {
        NOKEY = -1,
        DVK_FOLD = -23,
        DVK_TAB = 9,
        DVK_BACKSPACE = 8,
        DVK_CAPS = -15,
        DVK_SHIFT = -14,
        DVK_SPACE = 32,
        DVK_MENU = -5,
        DVK_ENTER = 10,
        DVK_CTRL = -16,
        DVK_UP = -17,
```

  DVK_RIGHT = -18,
  DVK_DOWN = -19,
  DVK_LEFT = -20,
  DVK_ALT = -26
}
enum values for the keyboard control keys. negative values are keys with no sensible ascii representation. numbers are chosen to mimic ascii control sequences.

More...

Functions

Keyboard *	keyboardDemoInit (void)
	initializes the keyboard with default options. Same as calling keyboardInit(NULL, 3, BgType_Text4bpp, BgSize_T_256x512, 20, 0, false, true)
int	keyboardGetChar (void)
	Waits for user to press a key and returns the key pressed. Use keyboardUpdate instead for async operation.
Keyboard *	keyboardGetDefault (void)
	Gets the default keyboard.
int	keyboardGetKey (int x, int y)
	returns the ascii code for the key located at the supplied x and y. Will not effect keyboard shift state.
void	keyboardGetString (char *buffer, int maxLen)
	reads the input until a the return key is pressed or the maxLen is exceeded.
void	keyboardHide (void)
	Hides the keyboard.
Keyboard *	keyboardInit (Keyboard *keyboard, int layer, BgType type, BgSize size, int mapBase, int tileBase, bool mainDisplay, bool loadGraphics)
	initializes the keyboard system with the supplied keyboard
void	keyboardShow (void)
	Displays the keyboard.
int	keyboardUpdate (void)
	Processes the keyboard. Should be called once per frame when using the keyboard in an async manner.

Detailed Description

nds stdio keyboard integration.

The keyboard component allows the use of a default keyboard via stdin as well as direct via the functions exposed below. The default behavior is a hidden keyboard that shows on a call to scanf(stdin, ...).
By default the keyboard uses background 3 of the sub display, consumes approximatly

40KiB of background vram begining at tile base 1 and 2KB of map stored at map base 30. The default is designed to function along side a default instance of the console print functionality.
To customize keyboard behavior and resource usage modify the keyboard structure returned by keyboardGetDefault() or create your own keyboard.

Enumeration Type Documentation

enum KeyboardState

States the keyboard can be in, currently only Lower and Upper supported.

Enumerator:

Lower	Normal keyboard display (lowercase letters)
Upper	Caps lock Held
Numeric	Numeric only keypad (not provided by the default keyboard)
Reduced	Reduced footprint keyboard (not provided by the default keyboard)

enum Keys

enum values for the keyboard control keys. negative values are keys with no sensible ascii representation. numbers are chosen to mimic ascii control sequences.

Enumerator:

NOKEY	will be returned if no key was pressed.
DVK_FOLD	will be returned if the fold key was pressed (topleft on the default keyboard).
DVK_TAB	will be returned if the tab key was pressed.
DVK_BACKSPACE	will be returned if the backspace key was pressed.
DVK_CAPS	will be returned if the caps key was pressed.
DVK_SHIFT	will be returned if the shift key was pressed.
DVK_SPACE	will be returned if the space key was pressed.
DVK_MENU	will be returned if the menu key was pressed.
DVK_ENTER	will be returned if the enter key was pressed.
DVK_CTRL	will be returned if the ctrl key was pressed.
DVK_UP	will be returned if the up key was pressed.
DVK_RIGHT	will be returned if the right key was pressed.
DVK_DOWN	will be returned if the down key was pressed.

DVK_LEFT	will be returned if the left key was pressed.
DVK_ALT	will be returned if the alt key was pressed.

Function Documentation

Keyboard* keyboardDemoInit (void)

initializes the keyboard with default options. Same as calling keyboardInit(NULL, 3, BgType_Text4bpp, BgSize_T_256x512, 20, 0, false, true)

Returns:

a pointer to the current keyboard.

Examples:

input/keyboard/keyboard_async/source/template.c, and input/keyboard/keyboard_stdin/source/keymain.c.

int keyboardGetKey (int *x*,
int *y*
)

returns the ascii code for the key located at the supplied x and y. Will not effect keyboard shift state.

Parameters:

x	the pixel x location
y	the pixel y location

Returns:

the key pressed or NOKEY if user pressed outside the keypad

void keyboardGetString (char * *buffer*,
int *maxLen*
)

reads the input until a the return key is pressed or the maxLen is exceeded.

Parameters:

buffer	a buffer to hold the input string
maxLen	the maximum length to read

Keyboard* keyboardInit (Keyboard * *keyboard*,
int *layer*,
BgType *type*,
BgSize *size*,
int *mapBase*,
int *tileBase*,
bool *mainDisplay*,
bool *loadGraphics*

)

initializes the keyboard system with the supplied keyboard

Parameters:

keyboard	the keyboard struct to initialize (can be NULL)
layer	the background layer to use
type	the background type to initialize
size	the background size to initialize
mapBase	the map base to use for the background
tileBase	the graphics tile base to use for the background
mainDisplay	if true the keyboard will render on the main display
loadGraphics	if true the keyboard graphics will be loaded

Returns:

returns the initialized keyboard struct

int keyboardUpdate (void)

Processes the keyboard. Should be called once per frame when using the keyboard in an async manner.

Returns:

the ascii code of the key pressed or -1 if no key was pressed.

Examples:

input/keyboard/keyboard_async/source/template.c.

console.h File Reference

nds stdio support. More...

```
#include <nds/ndstypes.h>
#include <nds/arm9/background.h>
```

Data Structures

struct	ConsoleFont
	a font struct for the console. More...
struct	PrintConsole
	console structure used to store the state of a console render context. More...

Typedefs

typedef struct ConsoleFont	ConsoleFont
	a font struct for the console.
typedef struct PrintConsole	PrintConsole
	console structure used to store the state of a console render context.

Enumerations

enum DebugDevice {
  DebugDevice_NULL = 0x0,
  DebugDevice_NOCASH = 0x1,
  DebugDevice_CONSOLE = 0x02
}
Console debug devices supported by libnds.

More...

Functions

void	consoleClear (void)
	Clears the screan by using iprintf("\x1b[2J");.
void	consoleDebugInit (DebugDevice device)
	Initializes debug console output on stderr to the specified device.
PrintConsole *	consoleDemoInit (void)
	Initialize the console to a default state for prototyping. This function sets the console to use sub display, VRAM_C, and BG0 and enables MODE_0_2D on the sub display. It is intended for use in prototyping applications which need print ability and not actual game use. Print functionality can be utilized with just this call.
PrintConsole *	consoleGetDefault (void)

	Gets a pointer to the console with the default values this should only be used when using a single console or without changing the console that is returned, other wise use consoleInit()
PrintConsole *	consoleInit (PrintConsole *console, int layer, BgType type, BgSize size, int mapBase, int tileBase, bool mainDisplay, bool loadGraphics)
	Initialise the console.
PrintConsole *	consoleSelect (PrintConsole *console)
	Make the specified console the render target.
void	consoleSetFont (PrintConsole *console, ConsoleFont *font)
	Loads the font into the console.
void	consoleSetWindow (PrintConsole *console, int x, int y, int width, int height)
	Sets the print window.

Detailed Description

nds stdio support.

Provides stdio integration for printing to the DS screen as well as debug print functionality provided by stderr.
General usage is to initialize the console by: consoleDemoInit() or to customize the console usage by: consoleInit()
The default instance utilizes the sub display, approximatly 15KiB of vram C starting at tile base 0 and 2KiB of map at map base 30.
Debug printing is performed by initializing the debug console via consoleDebugInit() as follows:

```
consoleDebugInit(DebugDevice_NOCASH);
fprintf(stderr, "debug message in no$gba window %i", stuff);

OR

consoleDebugInit(DebugDevice_CONSOLE);
fprintf(stderr, "debug message on DS console screen");
```

The print console must be initialized to use DB_CONSOLE

Typedef Documentation

typedef struct PrintConsole PrintConsole
console structure used to store the state of a console render context.

Default values from consoleGetDefault();

```
PrintConsole defaultConsole =
{
Font:
        {
                (u16*)default_font_bin, //font gfx
                0, //font palette
                0, //font color count
                4, //bpp
                0, //first ascii character in the set
                128, //number of characters in the font set
                true, //convert to single color
        },
        0, //font background map
        0, //font background gfx
        31, //map base
        0, //char base
        0, //bg layer in use
        -1, //bg id
        0,0, //cursorX cursorY
        0,0, //prevcursorX prevcursorY
        32, //console width
        24, //console height
        0,  //window x
        0,  //window y
        32, //window width
        24, //window height
        3, //tab size
        0, //font character offset
        0, //selected palette
        0,  //print callback
        false, //console initialized
        true, //load graphics
};
```

Enumeration Type Documentation

enum DebugDevice

Console debug devices supported by libnds.

Enumerator:

DebugDevice_NULL	swallows prints to stderr
DebugDevice_NOCASH	Directs stderr debug statements to no$gba debug window.
DebugDevice_CONSOLE	Directs stderr debug statements to DS console window.

Function Documentation

void consoleDebugInit (DebugDevice *device*)

Initializes debug console output on stderr to the specified device.

Parameters:

device The debug device (or devices) to output debug print statements to

PrintConsole* consoleDemoInit (void)

Initialize the console to a default state for prototyping. This function sets the console to use sub display, VRAM_C, and BG0 and enables MODE_0_2D on the sub display. It is intended for use in prototyping applications which need print ability and not actual game use. Print functionality can be utilized with just this call.

Returns:

A pointer to the current PrintConsole.

Examples:

audio/maxmod/audio_modes/source/main.c, audio/maxmod/basic_sound/source/MaxModExample.c, audio/maxmod/reverb/source/main.c, audio/maxmod/streaming/source/main.c, audio/micrecord/source/micrecord.c, card/eeprom/source/main.cpp, ds_motion/source/main.c, dswifi/ap_search/source/template.c, dswifi/autoconnect/source/autoconnect.c, dswifi/httpget/source/httpget.c, filesystem/libfat/libfatdir/source/directory.c, filesystem/nitrofs/nitrodir/source/directory.c, Graphics/3D/BoxTest/source/main.cpp, Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/Backgrounds/16bit_color_bmp/source/template.cpp, Graphics/Backgrounds/256_color_bmp/source/main.cpp, Graphics/Backgrounds/all_in_one/source/main.cpp, Graphics/Backgrounds/Double_Buffer/source/main.cpp, Graphics/Backgrounds/rotation/source/main.cpp, Graphics/Printing/ansi_console/source/main.c, Graphics/Printing/console_windows/source/main.c, Graphics/Sprites/allocation_test/source/main.c, Graphics/Sprites/bitmap_sprites/source/main.cpp, hello_world/source/main.cpp, input/keyboard/keyboard_async/source/template.c, input/keyboard/keyboard_stdin/source/keymain.c, input/Touch_Pad/touch_area/source/template.c, time/RealTimeClock/source/main.c, and time/stopwatch/source/main.c.

```
PrintConsole* consoleGetDefault  ( void  )
```

Gets a pointer to the console with the default values this should only be used when using a single console or without changing the console that is returned, other wise use consoleInit()

Returns:

A pointer to the console with the default values

```
PrintConsole* consoleInit  ( PrintConsole *  console,
                             int             layer,
                             BgType          type,
                             BgSize          size,
                             int             mapBase,
                             int             tileBase,
                             bool            mainDisplay,
                             bool            loadGraphics
                           )
```

Initialise the console.

Parameters:

console	A pointer to the console data to initialze (if it's NULL, the default console will be used)
layer	background layer to use
type	the type of the background

size	the size of the background
mapBase	the map base
tileBase	the tile graphics base
mainDisplay	if true main engine is used, otherwise false
loadGraphics	if true the default font graphics will be loaded into the layer

Returns:

A pointer to the current console.

Examples:

capture/ScreenShot/source/main.cpp, Graphics/3D/Paletted_Cube/source/main.cpp, Graphics/Printing/custom_font/source/main.c, Graphics/Printing/print_both_screens/source/template.c, Graphics/Printing/rotscale_text/source/main.c, and input/Touch_Pad/touch_test/source/main.c.

PrintConsole* consoleSelect (PrintConsole * *console*)

Make the specified console the render target.

Parameters:

console	A pointer to the console struct (must have been initialized with consoleInit(PrintConsole* console)

Returns:

a pointer to the previous console

Examples:

Graphics/Printing/console_windows/source/main.c, and Graphics/Printing/print_both_screens/source/template.c.

```
void consoleSetFont ( PrintConsole *  console,
                      ConsoleFont *   font
                    )
```

Loads the font into the console.

Parameters:

console	pointer to the console to update, if NULL it will update the current console
font	the font to load

Examples:

Graphics/Printing/custom_font/source/main.c, and Graphics/Printing/rotscale_text/source/main.c.

```
void consoleSetWindow ( PrintConsole *  console,
                        int             x,
                        int             y,
                        int             width,
                        int             height
                      )
```

Sets the print window.

Parameters:

console	console to set, if NULL it will set the current console window

x	x location of the window
y	y location of the window
width	width of the window
height	height of the window

Examples:

dswifi/ap_search/source/template.c, and Graphics/Printing/console_windows/source/main.c.

decompress.h File Reference

wraps the bios decompress functionality into something a bit easier to deal with. More...

```
#include <nds/ndstypes.h>
#include <nds/bios.h>
```

Enumerations

enum DecompressType {
LZ77,
LZ77Vram,
HUFF,
RLE,
RLEVram
}
the types of decompression available.

More...

Functions

void decompress (const void *data, void *dst, DecompressType type)
decompresses data using the suported type

void decompressStream (const void *data, void *dst, DecompressType type, getByteCallback readCB, getHeaderCallback getHeaderCB)
decompresses data using the suported type (only LZ77Vram, HUFF, and RLEVram support streaming)

Detailed Description

wraps the bios decompress functionality into something a bit easier to deal with.

Enumeration Type Documentation

enum DecompressType
the types of decompression available.

Enumerator:

LZ77	LZ77 decompression.
LZ77Vram	vram safe LZ77 decompression.
HUFF	vram safe huff decompression.

RLE	run length encoded decompression.
RLEVram	vram safe run length encoded decompression.

Function Documentation

```
void decompress ( const void *     data,
                  void *           dst,
                  DecompressType   type
                )
```

decompresses data using the suported type

Parameters:

dst	the destination to decompress to
data	the data to decompress
type	the type of data to decompress

Examples:

Graphics/Backgrounds/16bit_color_bmp/source/template.cpp.

```
void decompressStream ( const void *        data,
                        void *              dst,
                        DecompressType      type,
                        getByteCallback     readCB,
                        getHeaderCallback   getHeaderCB
                      )
```

decompresses data using the suported type (only LZ77Vram, HUFF, and RLEVram support streaming)

Parameters:

dst	the destination to decompress to.
data	the data to decompress.
type	the type of data to decompress.
readCB	a callback to read the next byte of data.
getHeaderCB	a callback to read the 32 byte header.

---

image.h File Reference

An image abstraction for working with image data. More...

```
#include <nds/arm9/video.h>
#include <nds/arm9/pcx.h>
```

Data Structures

struct	RGB_24 holds a red green blue triplet More...
struct	sImage A generic image structure. More...

Typedefs

typedef struct sImage	sImage A generic image structure.

Functions

struct RGB_24	__attribute__ ((packed)) RGB_24 holds a red green blue triplet
void	image24to16 (sImage *img) Converts a 24 bit image to 16 bit.
void	image8to16 (sImage *img) Converts an 8 bit image to 16 bit setting the alpha bit.
void	image8to16trans (sImage *img, u8 transparentColor) Converts an 8 bit image to 16 bit with alpha bit cleared for the supplied palette index.
void	imageDestroy (sImage *img) frees the image data. Only call if the image data was returned from an image loader
void	imageTileData (sImage *img) Tiles 8 bit image data into a sequence of 8x8 tiles.

Variables

unsigned char	b 8 bits for the blue value.
unsigned char	g 8 bits for the green value.
unsigned char	r 8 bits for the red value.

Detailed Description

An image abstraction for working with image data.

Image data pointers must be allocated using malloc as the conversion rutiens will free the pointers and allocate new data. As such any loader implemented utilizing this structure must use malloc() to allocate the image pointer data.

Function Documentation

struct RGB_24 __attribute__ ((packed))
holds a red green blue triplet

holds a red green blue triplet

Theme/Color values:

- 0 = Gray
- 1 = Brown
- 2 = Red
- 3 = Pink
- 4 = Orange
- 5 = Yellow
- 6 = Yellow/Green-ish
- 7 = Green
- 8 = Dark Green
- 9 = Green/Blue-ish
- 10 = Light Blue
- 11 = Blue
- 12 = Dark Blue
- 13 = Dark Purple
- 14 = Purple
- 15 = Purple/Red-ish

Language values:

- 0 = Japanese
- 1 = English
- 2 = French
- 3 = German
- 4 = Italian
- 5 = Spanish
- 6 = Chinese(?)
- 7 = Unknown/Reserved

< User's language.

< GBA screen selection (lower screen if set, otherwise upper screen).

< Brightness level at power on, dslite.

< The DS should boot from the DS cart or GBA cart automatically if one is inserted.

< User Settings Lost (0=Normal, 1=Prompt/Settings Lost)

void image24to16 (sImage * *img*)
Converts a 24 bit image to 16 bit.

Parameters:

img a pointer to image to manipulate

void image8to16 (sImage * *img*)

Converts an 8 bit image to 16 bit setting the alpha bit.

Parameters:

img a pointer to image to manipulate

Examples:

Graphics/3D/nehe/lesson06/source/nehe6.cpp, Graphics/3D/nehe/lesson07/source/nehe7.cpp, Graphics/3D/nehe/lesson08/source/nehe8.cpp, Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/3D/nehe/lesson10b/source/nehe10b.cpp, Graphics/3D/nehe/lesson11/source/nehe11.cpp, Graphics/3D/Ortho/source/main.cpp, and input/Touch_Pad/touch_look/source/main.cpp.

void image8to16trans (sImage * *img*,
u8 *transparentColor*
)

Converts an 8 bit image to 16 bit with alpha bit cleared for the supplied palette index.

Parameters:

img	a pointer to image to manipulate
transparentColor	Color indexes equal to this value will have the alpha bit clear

Examples:

Graphics/3D/nehe/lesson09/source/nehe9.cpp.

void imageDestroy (sImage * *img*)

frees the image data. Only call if the image data was returned from an image loader

Parameters:

img a pointer to image to manipulate (the image data will be free())

Examples:

Graphics/3D/nehe/lesson06/source/nehe6.cpp, Graphics/3D/nehe/lesson07/source/nehe7.cpp, Graphics/3D/nehe/lesson08/source/nehe8.cpp, Graphics/3D/nehe/lesson09/source/nehe9.cpp, Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/3D/Ortho/source/main.cpp, and input/Touch_Pad/touch_look/source/main.cpp.

void imageTileData (sImage * *img*)

Tiles 8 bit image data into a sequence of 8x8 tiles.

Parameters:

img a pointer to image to manipulate

Examples:

Graphics/Sprites/fire_and_sprites/source/main.cpp.

pcx.h File Reference

A simple 256 color pcx file loader. More...

Functions

struct PCXHeader	__attribute__ ((packed)) PCXHeader
	holds a red green blue triplet
int	loadPCX (const unsigned char *pcx, sImage *image)
	Loads an image structure with data from PCX formatted data.

Variables

char	version
	version of the banner.

Detailed Description

A simple 256 color pcx file loader.

Function Documentation

```
int loadPCX  ( const unsigned char *  pcx,
               sImage *               image
             )
```

Loads an image structure with data from PCX formatted data.

Parameters:

pcx	a pointer to the pcx file loaded into memory
image	the image structure to fill in (the loader will allocate room for the palette and pixel data)

Examples:

Graphics/3D/nehe/lesson06/source/nehe6.cpp, Graphics/3D/nehe/lesson07/source/nehe7.cpp, Graphics/3D/nehe/lesson08/source/nehe8.cpp, Graphics/3D/nehe/lesson09/source/nehe9.cpp, Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/3D/nehe/lesson10b/source/nehe10b.cpp, Graphics/3D/nehe/lesson11/source/nehe11.cpp, Graphics/3D/Ortho/source/main.cpp, Graphics/Sprites/fire_and_sprites/source/main.cpp, and input/Touch_Pad/touch_look/source/main.cpp.

dynamicArray.h File Reference

A dynamically resizing array for general use. More...

```
#include <stdlib.h>
#include <string.h>
#include <nds/ndstypes.h>
```

Data Structures

struct	DynamicArray A resizable array. More...

Typedefs

typedef struct DynamicArray	DynamicArray A resizable array.

Functions

void	DynamicArrayDelete (DynamicArray *v) Frees memory allocated by the dynamic array.
void *	DynamicArrayGet (DynamicArray *v, unsigned int index) Gets the entry at the supplied index.
void *	DynamicArrayInit (DynamicArray *v, unsigned int initialSize) Initializes an array with the supplied initial size.
bool	DynamicArraySet (DynamicArray *v, unsigned int index, void *item) Sets the entry to the supplied value.

Detailed Description

A dynamically resizing array for general use.

Function Documentation

void DynamicArrayDelete (DynamicArray * *v*)

Frees memory allocated by the dynamic array.

Parameters:

v The array to delete

void* DynamicArrayGet (DynamicArray * *v*,
unsigned int *index*
)

Gets the entry at the supplied index.

Parameters:

v	The array to get from.
index	The index of the data to get.

Returns:

The data or NULL if v is NULL or the index is out of range.

```
void* DynamicArrayInit  ( DynamicArray *  v,
                          unsigned int    initialSize
                        )
```

Initializes an array with the supplied initial size.

Parameters:

v	the array to initialize
initialSize	the initial size to allocate

Returns:

a pointer to the data, or NULL on error.

```
bool DynamicArraySet  ( DynamicArray *  v,
                        unsigned int    index,
                        void *          item
                      )
```

Sets the entry to the supplied value.

Parameters:

v	The array to set
index	The index of the data to set (array will be resized to fit the index).
item	The data to set.

Returns:

false if v is NULL or there isn't enough memory, true otherwise

linkedlist.h File Reference

A simple doubly linked, unsorted list implementation. More...

```
#include <malloc.h>
```

Data Structures

struct	LinkedList A node for the linked list. More...

Typedefs

typedef struct LinkedList	LinkedList A node for the linked list.

Functions

LinkedList *	linkedlistAdd (LinkedList **front, void *data) Adds data to a linked list.
void	linkedlistRemove (LinkedList *node) Removes a node from a linked list.

Detailed Description

A simple doubly linked, unsorted list implementation.

Function Documentation

```
LinkedList* linkedlistAdd  ( LinkedList **  front,
                             void *         data
                           )
```

Adds data to a linked list.

This will only store the pointer to the data, so you have to make sure that the pointer stays valid.

Parameters:

front	A pointer to a pointer to the front of the linked list (or a pointer to NULL if you don't have a linked list yet).
data	A pointer to the data you want to store.

Returns:

A pointer to the new node, which is also the new front, or NULL if there is not enough memory.

```
void linkedlistRemove  ( LinkedList *  node )
```

Removes a node from a linked list.

The data pointer of the node will be lost after this, so make sure you don't need it anymore.

Parameters:

node	The node you want to remove.

Custom Peripherals

rumble.h File Reference

nds rumble option pak support. More...

Functions

bool	isRumbleInserted (void)
	Check for rumble option pak.
void	setRumble (bool position)
	Fires the rumble actuator.

Variables

typedef	__attribute__
	the NDS file header format See gbatek for more info.

Detailed Description

nds rumble option pak support.

Function Documentation

bool isRumbleInserted (void)

Check for rumble option pak.

Returns:

true if the cart in the GBA slot is a Rumble option pak.

void setRumble (bool *position*)

Fires the rumble actuator.

Parameters:

position	Alternates position of the actuator in the pak

Variable Documentation

__attribute__

the NDS file header format See gbatek for more info.

the NDS banner format. See gbatek for more information.

< User's language.

< GBA screen selection (lower screen if set, otherwise upper screen).

< Brightness level at power on, dslite.

< The DS should boot from the DS cart or GBA cart automatically if one is inserted.

< User Settings Lost (0=Normal, 1=Prompt/Settings Lost)

---

ndsmotion.h File Reference

interface code for the ds motion card, ds motion pak, MK6. More...

Functions

int	motion_acceleration_x (void) gets acceleration value to mili G (where g is 9.8 m/s*s)
int	motion_acceleration_y (void) gets acceleration value to mili G (where g is 9.8 m/s*s)
int	motion_acceleration_z (void) gets acceleration value to mili G (where g is 9.8 m/s*s)
void	motion_deinit (void) Deinitializes the DS Motion Sensor.
void	motion_enable_ain_1 (void) This enables the analog input number 1. Required before reading analog input number 1.
void	motion_enable_ain_2 (void) This enables the analog input number 2. Required before reading analog input number 2.
MotionCalibration *	motion_get_calibration (void) This returns the current calibration settings for saving.
int	motion_init (void) Initializes the DS Motion Sensor. Run this before using any of the DS Motion Sensor functions save the return value and pass it to the other motion_ functions.
int	motion_read_ain_1 (void) This reads the analog input number 1. analog input number 1 needs to be enabled before reading.
int	motion_read_ain_2 (void) This reads the analog input number 2. analog input number 2 needs to be enabled before reading.
signed int	motion_read_gyro (void) read the Z rotational speed
signed int	motion_read_x (void) read the X acceleration
signed int	motion_read_y (void) read the Y acceleration
signed int	motion_read_z (void) read the Z acceleration
int	motion_rotation (void) converts raw rotation value to degrees per second
void	motion_set_calibration (MotionCalibration *cal)

This sets the calibration settings. Intended to restore saved calibration settings.

void motion_set_offs_gyro (void)

this should be called when the axis is under no rotation Default is 1680

void motion_set_offs_x (void)

this should be called when the axis is under no acceleration. Default is 2048

void motion_set_offs_y (void)

this should be called when the axis is under no acceleration Default is 2048

void motion_set_offs_z (void)

this should be called when the axis is under no acceleration Default is 2048

void motion_set_sens_gyro (int sens)

this should be passed the raw reading at 1g for accurate acceleration calculations. Default is 825

void motion_set_sens_x (int sens)

this should be passed the raw reading at 1g for accurate acceleration calculations. Default is 819

void motion_set_sens_y (int sens)

this should be passed the raw reading at 1g for accurate acceleration calculations. Default is 819

void motion_set_sens_z (int sens)

this should be passed the raw reading at 1g for accurate acceleration calculations. Default is 819

Detailed Description

interface code for the ds motion card, ds motion pak, MK6.

Function Documentation

int motion_init (void)

Initializes the DS Motion Sensor. Run this before using any of the DS Motion Sensor functions save the return value and pass it to the other motion_ functions.

Returns:

Examples:

ds_motion/source/main.c.

signed int motion_read_gyro (void)

read the Z rotational speed

Returns:

Examples:
ds_motion/source/main.c.

signed int motion_read_x (void)
read the X acceleration

Returns:

Examples:
ds_motion/source/main.c.

signed int motion_read_y (void)
read the Y acceleration

Returns:

Examples:
ds_motion/source/main.c.

signed int motion_read_z (void)
read the Z acceleration

Returns:

Examples:
ds_motion/source/main.c.

void motion_set_calibration (MotionCalibration * *cal*)
This sets the calibration settings. Intended to restore saved calibration settings.

Parameters:
cal the calibration settings

void motion_set_sens_gyro (int *sens*)
this should be passed the raw reading at 1g for accurate acceleration calculations. Default is 825

Parameters:
sens the raw reading at 1g for accurate acceleration calculations

void motion_set_sens_x (int *sens*)
this should be passed the raw reading at 1g for accurate acceleration calculations. Default is 819

Parameters:
sens the raw reading at 1g for accurate acceleration calculations

void motion_set_sens_y (int *sens*)
this should be passed the raw reading at 1g for accurate acceleration calculations. Default is 819

Parameters:
sens the raw reading at 1g for accurate acceleration calculations

void motion_set_sens_z (int *sens*)
this should be passed the raw reading at 1g for accurate acceleration calculations. Default

is 819

Parameters:

sens the raw reading at 1g for accurate acceleration calculations

Examples:

ds_motion/source/main.c.

piano.h File Reference

NDS Easy Piano option pack support. More...

Functions

bool	pianoIsInserted ()
	Check for piano option pack.
u16	pianoKeysDown ()
	Obtains the current piano keys pressed state.
u16	pianoKeysHeld ()
	Obtains the current piano keys held state.
u16	pianoKeysUp ()
	Obtains the current piano keys released state.
void	pianoScanKeys ()
	Obtain the current piano state. Call this function once per main loop to use the piano functions.

Detailed Description

NDS Easy Piano option pack support.

Function Documentation

bool pianoIsInserted ( )

Check for piano option pack.

Returns:

true if the cart in the GBA slot is the piano option pack.

Debugging

console.h File Reference

nds stdio support. More...

```
#include <nds/ndstypes.h>
#include <nds/arm9/background.h>
```

Data Structures

struct ConsoleFont
: a font struct for the console. More...

struct PrintConsole
: console structure used to store the state of a console render context. More...

Typedefs

typedef struct ConsoleFont ConsoleFont
: a font struct for the console.

typedef struct PrintConsole PrintConsole
: console structure used to store the state of a console render context.

Enumerations

enum DebugDevice {
  DebugDevice_NULL = 0x0,
  DebugDevice_NOCASH = 0x1,
  DebugDevice_CONSOLE = 0x02
}
Console debug devices supported by libnds.

More...

Functions

void consoleClear (void)
: Clears the screan by using iprintf("\x1b[2J");.

void consoleDebugInit (DebugDevice device)
: Initializes debug console output on stderr to the specified device.

PrintConsole * consoleDemoInit (void)
: Initialize the console to a default state for prototyping. This function sets the console to use sub display, VRAM_C, and BG0 and enables MODE_0_2D on the sub display. It is intended for use in prototyping

	applications which need print ability and not actual game use. Print functionality can be utilized with just this call.
PrintConsole *	consoleGetDefault (void)
	Gets a pointer to the console with the default values this should only be used when using a single console or without changing the console that is returned, other wise use consoleInit()
PrintConsole *	consoleInit (PrintConsole *console, int layer, BgType type, BgSize size, int mapBase, int tileBase, bool mainDisplay, bool loadGraphics)
	Initialise the console.
PrintConsole *	consoleSelect (PrintConsole *console)
	Make the specified console the render target.
void	consoleSetFont (PrintConsole *console, ConsoleFont *font)
	Loads the font into the console.
void	consoleSetWindow (PrintConsole *console, int x, int y, int width, int height)
	Sets the print window.

Detailed Description

nds stdio support.

Provides stdio integration for printing to the DS screen as well as debug print functionality provided by stderr.
General usage is to initialize the console by: consoleDemoInit() or to customize the console usage by: consoleInit()
The default instance utilizes the sub display, approximatly 15KiB of vram C starting at tile base 0 and 2KiB of map at map base 30.
Debug printing is performed by initializing the debug console via consoleDebugInit() as follows:

```
consoleDebugInit(DebugDevice_NOCASH);
fprintf(stderr, "debug message in no$gba window %i", stuff);

OR

consoleDebugInit(DebugDevice_CONSOLE);
fprintf(stderr, "debug message on DS console screen");
```

The print console must be initialized to use DB_CONSOLE

Typedef Documentation

typedef struct PrintConsole PrintConsole
console structure used to store the state of a console render context.

Default values from consoleGetDefault();

```
PrintConsole defaultConsole =
{
Font:
        {
                (u16*)default_font_bin, //font gfx
                0, //font palette
                0, //font color count
                4, //bpp
                0, //first ascii character in the set
                128, //number of characters in the font set
                true, //convert to single color
        },
        0, //font background map
        0, //font background gfx
        31, //map base
        0, //char base
        0, //bg layer in use
        -1, //bg id
        0,0, //cursorX cursorY
        0,0, //prevcursorX prevcursorY
        32, //console width
        24, //console height
        0,  //window x
        0,  //window y
        32, //window width
        24, //window height
        3, //tab size
        0, //font character offset
        0, //selected palette
        0,  //print callback
        false, //console initialized
        true, //load graphics
};
```

Enumeration Type Documentation

enum DebugDevice

Console debug devices supported by libnds.

Enumerator:

DebugDevice_NULL	swallows prints to stderr
DebugDevice_NOCASH	Directs stderr debug statements to no$gba debug window.
DebugDevice_CONSOLE	Directs stderr debug statements to DS console window.

Function Documentation

void consoleDebugInit (DebugDevice *device*)

Initializes debug console output on stderr to the specified device.

Parameters:

device The debug device (or devices) to output debug print statements to

PrintConsole* consoleDemoInit (void)

Initialize the console to a default state for prototyping. This function sets the console to use sub display, VRAM_C, and BG0 and enables MODE_0_2D on the sub display. It is intended for use in prototyping applications which need print ability and not actual game use. Print functionality can be utilized with just this call.

Returns:

A pointer to the current PrintConsole.

Examples:

audio/maxmod/audio_modes/source/main.c, audio/maxmod/basic_sound/source/MaxModExample.c, audio/maxmod/reverb/source/main.c, audio/maxmod/streaming/source/main.c, audio/micrecord/source/micrecord.c, card/eeprom/source/main.cpp, ds_motion/source/main.c, dswifi/ap_search/source/template.c, dswifi/autoconnect/source/autoconnect.c, dswifi/httpget/source/httpget.c, filesystem/libfat/libfatdir/source/directory.c, filesystem/nitrofs/nitrodir/source/directory.c, Graphics/3D/BoxTest/source/main.cpp, Graphics/3D/nehe/lesson10/source/nehe10.cpp, Graphics/Backgrounds/16bit_color_bmp/source/template.cpp, Graphics/Backgrounds/256_color_bmp/source/main.cpp, Graphics/Backgrounds/all_in_one/source/main.cpp, Graphics/Backgrounds/Double_Buffer/source/main.cpp, Graphics/Backgrounds/rotation/source/main.cpp, Graphics/Printing/ansi_console/source/main.c, Graphics/Printing/console_windows/source/main.c, Graphics/Sprites/allocation_test/source/main.c, Graphics/Sprites/bitmap_sprites/source/main.cpp, hello_world/source/main.cpp, input/keyboard/keyboard_async/source/template.c, input/keyboard/keyboard_stdin/source/keymain.c, input/Touch_Pad/touch_area/source/template.c, time/RealTimeClock/source/main.c, and time/stopwatch/source/main.c.

```
PrintConsole* consoleGetDefault ( void )
```

Gets a pointer to the console with the default values this should only be used when using a single console or without changing the console that is returned, other wise use consoleInit()

Returns:

A pointer to the console with the default values

```
PrintConsole* consoleInit ( PrintConsole *  console,
                            int             layer,
                            BgType          type,
                            BgSize          size,
                            int             mapBase,
                            int             tileBase,
                            bool            mainDisplay,
                            bool            loadGraphics
                          )
```

Initialise the console.

Parameters:

console	A pointer to the console data to initialze (if it's NULL, the default

	console will be used)
layer	background layer to use
type	the type of the background
size	the size of the background
mapBase	the map base
tileBase	the tile graphics base
mainDisplay	if true main engine is used, otherwise false
loadGraphics	if true the default font graphics will be loaded into the layer

Returns:

A pointer to the current console.

Examples:

capture/ScreenShot/source/main.cpp, Graphics/3D/Paletted_Cube/source/main.cpp, Graphics/Printing/custom_font/source/main.c, Graphics/Printing/print_both_screens/source/template.c, Graphics/Printing/rotscale_text/source/main.c, and input/Touch_Pad/touch_test/source/main.c.

PrintConsole* consoleSelect (PrintConsole * *console*)

Make the specified console the render target.

Parameters:

console	A pointer to the console struct (must have been initialized with consoleInit(PrintConsole* console)

Returns:

a pointer to the previous console

Examples:

Graphics/Printing/console_windows/source/main.c, and Graphics/Printing/print_both_screens/source/template.c.

```
void consoleSetFont ( PrintConsole * console,
                      ConsoleFont *  font
                    )
```

Loads the font into the console.

Parameters:

console	pointer to the console to update, if NULL it will update the current console
font	the font to load

Examples:

Graphics/Printing/custom_font/source/main.c, and Graphics/Printing/rotscale_text/source/main.c.

```
void consoleSetWindow ( PrintConsole * console,
                        int            x,
                        int            y,
                        int            width,
                        int            height
                      )
```

Sets the print window.

Parameters:

console	console to set, if NULL it will set the current console window
x	x location of the window
y	y location of the window
width	width of the window
height	height of the window

Examples:

dswifi/ap_search/source/template.c, and Graphics/Printing/console_windows/source/main.c.

debug.h File Reference

Currently only used to send debug messages to NO$GBA debug window. More...

Functions

void nocashMessage (const char *message)
 Send a message to the no$gba debug window.

Detailed Description

Currently only used to send debug messages to NO$GBA debug window.

On the ARM 9 this functionality is best accessed via the console studio integration.

- Debug Messages via stdio

Function Documentation

void nocashMessage (const char * *message*)

Send a message to the no$gba debug window.

Parameters:

message	The message to send

sassert.h File Reference

Simple assertion with a message conplies to nop if NDEBUG is defined. More...

```
#include "_ansi.h"
```

Defines

#define	sassert(e, msg) ((e) ? (void)0 : __sassert(__FILE__, __LINE__, #e, msg))
	Causes a blue screen of death if e is not true with the msg "msg" displayed.