

gbdk-lib Reference Manual

Generated by Doxygen 1.0.0

Sat May 6 19:57:06 2000

Contents

1	Gameboy Development Kit Library (gbdk-lib) documentation.	1
2	gbdk-lib Compound Index	3
2.1	gbdk-lib Compound List	3
3	gbdk-lib File Index	5
3.1	gbdk-lib File List	5
4	gbdk-lib Class Documentation	7
4.1	_fixed Union Reference	7
5	gbdk-lib File Documentation	9
5.1	cgb.h File Reference	9
5.2	console.h File Reference	11
5.3	ctype.h File Reference	12
5.4	drawing.h File Reference	13
5.5	gb.h File Reference	16
5.6	gbdk-lib.h File Reference	29
5.7	hardware.h File Reference	30
5.8	malloc.h File Reference	31
5.9	rand.h File Reference	33
5.10	sample.h File Reference	34
5.11	sgb.h File Reference	35
5.12	stdio.h File Reference	36
5.13	string.h File Reference	38
5.14	time.h File Reference	40
5.15	asm/gbz80/types.h File Reference	41
5.16	asm/types.h File Reference	42

Chapter 1

Gameboy Development Kit Library (gbdk-lib) documentation.

The following pages document a good size chunk of the libraries that go along with gbdk. They were automatically generated from the header files using `doxygen libc.dox`

Thanks to quang for many of the comments to the gb functions. Some of the comments are ripped directly from the Linux Programmers manual, and some directly from the pan/k00Pa document; due thanks must be given.

Links:

[quangs homepage.](#)

[Jeff Frohwein's GB development page.](#) Go there for the GeeBee GB faq, and the essential pan/k00Pa document.

[The gbdk homepage.](#)

Chapter 2

gbdk-lib Compound Index

2.1 gbdk-lib Compound List

Here are the classes, structs, unions and interfaces with brief descriptions:

_fixed (Useful definition for fixed point values) 7

Chapter 3

gbdk-lib File Index

3.1 gbdk-lib File List

Here is a list of all documented files with brief descriptions:

cgb.h (Support for Color GameBoy)	9
console.h (Console functions that work like Turbo C's)	11
ctype.h (Character type functions)	12
drawing.h (All Points Addressable (APA) mode drawing library)	13
gb.h (Gameboy specific functions)	16
gbdk-lib.h (Settings for the greater library system)	29
hardware.h (Defines that let the GB's hardware registers be accessed from C)	30
malloc.h (Header for a simple implementation of malloc())	31
rand.h (Random generator using the linear congruential method)	33
sample.h (Playback raw sound sample with length len from start at 8192Hz rate)	34
sgb.h (Super Gameboy definitions)	35
stdio.h (Basic file/console input output functions)	36
string.h (Generic string functions)	38
time.h (Sort of ANSI compliant time functions)	40
asm/gbz80/types.h (Types definitions for the gb)	41
asm/types.h (Shared types definitions)	42

Chapter 4

gbdk-lib Class Documentation

4.1 _fixed Union Reference

Useful definition for fixed point values.

```
#include <asm/types.h>
```

Public Members

- struct {
 UBYTE l
 UBYTE h
} b
- UWORD w
- struct {
 UBYTE l
 UBYTE h
} b

Detailed Description

Useful definition for fixed point values.

The documentation for this union was generated from the following files:

- **asm/types.h**
- asm/types.h~

Chapter 5

gbdk-lib File Documentation

5.1 cgb.h File Reference

Support for Color GameBoy.

Defines

- **#define RGB** (r, g, b)
Macro to create a palette entry out of the color components.
- **#define RGB_RED**
Common colors based on the EGA default palette.

Functions

- void **set_bkg_palette** (UINT8 first_palette, UINT8 nb_palettes, UINT16 *rgb_data)
Set bkg palette(s).
- void **set_sprite_palette** (UINT8 first_palette, UINT8 nb_palettes, UINT16 *rgb_data)
Set sprite palette(s).
- void **set_bkg_palette_entry** (UINT8 palette, UINT8 entry, UINT16 rgb_data)
Set a bkg palette entry.
- void **set_sprite_palette_entry** (UINT8 palette, UINT8 entry, UINT16 rgb_data)
Set a sprite palette entry.
- void **cpu_slow** (void)
Set CPU speed to slow operation.
- void **cpu_fast** (void)
Set CPU speed to fast operation.
- void **cgb_compatibility** (void)

Set defaults compatible with normal GameBoy.

Detailed Description

Support for Color GameBoy.

Function Documentation

5.1.1 void `cpu_slow` (void)

Set CPU speed to slow operation.

Make sure interrupts are disabled before call.

See also:

`cpu_fast`() (p. 10)

5.1.2 void `cpu_fast` (void)

Set CPU speed to fast operation.

Make sure interrupts are disabled before call.

See also:

`cpu_slow`() (p. 10)

5.2 console.h File Reference

Console functions that work like Turbo C's.

Functions

- void **gotoxy** (UINT8 x, UINT8 y)
Move the cursor to an absolute position.
- UINT8 **posx** (void)
Get the current X position of the cursor.
- UINT8 **posy** (void)
Get the current Y position of the cursor.
- void **setchar** (char c)
Writes out a single character at the current cursor position.

Detailed Description

Console functions that work like Turbo C's.

Note that the font is 8x8, making the screen 20x18 characters.

Function Documentation

5.2.1 void setchar (char c)

Writes out a single character at the current cursor position.

Does not update the cursor or interpret the character.

5.3 ctype.h File Reference

Character type functions.

Functions

- **BOOLEAN** `isalpha` (char c)
- **BOOLEAN** `isupper` (char c)
- **BOOLEAN** `islower` (char c)
- **BOOLEAN** `isdigit` (char c)
- **BOOLEAN** `isspace` (char c)
- char `toupper` (char c)
- char `tolower` (char c)

Detailed Description

Character type functions.

5.4 drawing.h File Reference

All Points Addressable (APA) mode drawing library.

Defines

- **#define GRAPHICS_WIDTH**
Size of the screen in pixels.
- **#define SOLID**
Possible drawing modes.
- **#define WHITE**
Possible drawing colours.
- **#define M_NOFILL**
Possible fill styles for `box()` (p. 14) and `circle()` (p. 14).
- **#define SIGNED**
Possible values for `signed_value` in `gprintln()` (p. 14) and `gprintn()` (p. 13).

Functions

- **void gprint** (char *str)
Print the string 'str' with no interpretation.
- **void gprintln** (INT16 number, INT8 radix, INT8 signed_value)
Print the long number 'number' in radix 'radix'.
- **void gprintn** (INT8 number, INT8 radix, INT8 signed_value)
Print the number 'number' as in 'gprintln'.
- **INT8 gprintf** (char *fmt,...)
Print the formatted string 'fmt' with arguments '...'.
- **void plot** (UINT8 x, UINT8 y, UINT8 colour, UINT8 mode)
Old style plot - try `plot_point()` (p. 13).
- **void plot_point** (UINT8 x, UINT8 y)
Plot a point in the current drawing mode and colour at (x,y).
- **void switch_data** (UINT8 x, UINT8 y, unsigned char *src, unsigned char *dst)
I (MLH) have no idea what switch_data does...
- **void draw_image** (unsigned char *data)
Ditto.
- **void line** (UINT8 x1, UINT8 y1, UINT8 x2, UINT8 y2)
Draw a line in the current drawing mode and colour from (x1,y1) to (x2,y2).

- void **box** (UINT8 x1, UINT8 y1, UINT8 x2, UINT8 y2, UINT8 style)

Draw a box (rectangle) with corners (x1,y1) and (x2,y2) using fill mode 'style' (one of NOFILL or FILL).
- void **circle** (UINT8 x, UINT8 y, UINT8 radius, UINT8 style)

Draw a circle with centre at (x,y) and radius 'radius'.
- UINT8 **getpix** (UINT8 x, UINT8 y)

Returns the current colour of the pixel at (x,y).
- void **wrtchr** (char chr)

Prints the character 'chr' in the default font at the current position.
- void **gotogxy** (UINT8 x, UINT8 y)

Sets the current text position to (x,y).
- void **color** (UINT8 forecolor, UINT8 backcolor, UINT8 mode)

Set the current foreground colour (for pixels), background colour, and draw mode.

Detailed Description

All Points Addressable (APA) mode drawing library.

Drawing routines originally by Pascal Felber
 Legendary overhaul by Jon Fuge <jonny@q-continuum.demon.co.uk>
 Commenting by Michael Hope

Note that the standard text **printf()** (p. 36) and **putchar()** (p. 36) cannot be used in APA mode - use **gprintf()** (p. 13) and **wrtchr()** (p. 14) instead.

Function Documentation

5.4.1 void gprintln (INT16 *number*, INT8 *radix*, INT8 *signed_value*)

Print the long number 'number' in radix 'radix'.

signed_value should be set to SIGNED or UNSIGNED depending on whether the number is signed or not

5.4.2 void circle (UINT8 *x*, UINT8 *y*, UINT8 *radius*, UINT8 *style*)

Draw a circle with centre at (x,y) and radius 'radius'.

'style' sets the fill mode

5.4.3 void gotogxy (UINT8 *x*, UINT8 *y*)

Sets the current text position to (x,y).

Note that x and y have units of cells (8 pixels)

5.5 gb.h File Reference

Gameboy specific functions.

Defines

- **#define J_START**
Joypad bits.
- **#define M_DRAWING**
Screen modes.
- **#define M_NO_SCROLL**
Set this in addition to the others to disable scrolling If scrolling is disabled, the cursor returns to (0,0).
- **#define M_NO_INTERP**
Set this to disable \n interpretation.
- **#define S_PALETTE**
If this is set, sprite colours come from OBJ1PAL.
- **#define S_FLIPX**
If set the sprite will be flipped horizontally.
- **#define S_FLIPY**
If set the sprite will be flipped vertically.
- **#define S_PRIORITY**
If this bit is clear, then the sprite will be displayed ontop of the background and window.
- **#define VBL_IFLAG**
Vertical blank interrupt.
- **#define LCD_IFLAG**
Interrupt when triggered by the STAT register.
- **#define TIM_IFLAG**
Interrupt when the timer TIMA overflows.
- **#define SIO_IFLAG**
Occurs when the serial transfer has completed.
- **#define JOY_IFLAG**
Occurs on a transition of the keypad.
- **#define SCREENWIDTH**
Width of the visible screen in pixels.
- **#define SCREENHEIGHT**

Height of the visible screen in pixels.

- **#define DMG_TYPE**
Original GB or Super GB.
- **#define MGB_TYPE**
Pocket GB or Super GB 2.
- **#define CGB_TYPE**
Color GB.
- **#define IO_IDLE**
IO is completed.
- **#define IO_SENDING**
Sending data.
- **#define IO_RECEIVING**
Receiving data.
- **#define IO_ERROR**
Error.
- **#define SWITCH_ROM_MBC1** (b)
Switches the upper 16k bank of the 32k rom to bank rombank using the MBC1 controller.
- **#define SWITCH_RAM_MBC1** (b)
- **#define SWITCH_ROM_MBC5** (b)
MBC5.
- **#define SWITCH_RAM_MBC5** (b)
- **#define DISPLAY_ON**
Turns the display back on.
- **#define DISPLAY_OFF**
Turns the display off immediatly.
- **#define SHOW_BKG**
Turns on the background layer.
- **#define HIDE_BKG**
Turns off the background layer.
- **#define SHOW_WIN**
Turns on the window layer Sets bit 5 of the LCDC register to 1.
- **#define HIDE_WIN**
Turns off the window layer.
- **#define SHOW_SPRITES**
Turns on the sprites layer.

- **#define HIDE_SPRITES**
Turns off the sprites layer.
- **#define SPRITES_8x16**
Sets sprite size to 8x16 pixels, two tiles one above the other.
- **#define SPRITES_8x8**
Sets sprite size to 8x8 pixels, one tile.

Typedefs

- **typedef void (* int_handler)(void)**
Interrupt handlers.

Functions

- **void remove_VBL (int_handler h)**
The remove functions will remove any interrupt handler.
- **void remove_LCD (int_handler h)**
- **void remove_TIM (int_handler h)**
- **void remove_SIO (int_handler h)**
- **void remove_JOY (int_handler h)**
- **void add_VBL (int_handler h)**
Adds a V-blank interrupt handler.
- **void add_LCD (int_handler h)**
Adds a LCD interrupt handler.
- **void add_TIM (int_handler h)**
Adds a timer interrupt handler.
- **void mode (UINT8 m)**
Set the current mode - one of M_ defined above.*
- **UINT8 get_mode (void)**
Returns the current mode.
- **void send_byte (void)**
Send byte in _io_out to the serial port.
- **void receive_byte (void)**
Receive byte from the serial port in _io_in.
- **void delay (UINT16 d)**
Delays the given number of milliseconds.

- **UINT8 joypad** (void)
Reads and returns the current state of the joypad.
- **UINT8 waitpad** (UINT8 mask)
Waits until all the keys given in mask are pressed.
- **void waitpadup** (void)
Waits for the pad and all buttons to be released.
- **void enable_interrupts** (void)
Enables unmasked interrupts.
- **void disable_interrupts** (void)
Disables interrupts.
- **void set_interrupts** (UINT8 flags)
Clears any pending interrupts and sets the interrupt mask register IO to flags.
- **void reset** (void)
Performs a warm reset by reloading the CPU value then jumping to the start of crt0 (0x0150).
- **void wait_vbl_done** (void)
Waits for the vertical blank interrupt (VBL) to finish.
- **void display_off** (void)
Turns the display off.
- **void hiramcpy** (UINT8 dst, const void *src, UINT8 n)
Copies data from somewhere in the lower address space to part of hi-ram.
- **void set_bkg_data** (UINT8 first_tile, UINT8 nb_tiles, unsigned char *data)
Sets the tile patterns in the Background Tile Pattern table.
- **void set_bkg_tiles** (UINT8 x, UINT8 y, UINT8 w, UINT8 h, unsigned char *tiles)
Sets the tiles in the background tile table.
- **void get_bkg_tiles** (UINT8 x, UINT8 y, UINT8 w, UINT8 h, unsigned char *tiles)
- **void move_bkg** (UINT8 x, UINT8 y)
Moves the background layer to the position specified in x and y in pixels.
- **void scroll_bkg** (INT8 x, INT8 y)
Moves the background relative to it's current position.
- **void set_win_data** (UINT8 first_tile, UINT8 nb_tiles, unsigned char *data)
Sets the window tile data.
- **void set_win_tiles** (UINT8 x, UINT8 y, UINT8 w, UINT8 h, unsigned char *tiles)
Sets the tiles in the win tile table.
- **void get_win_tiles** (UINT8 x, UINT8 y, UINT8 w, UINT8 h, unsigned char *tiles)
- **void move_win** (UINT8 x, UINT8 y)

Moves the window layer to the position specified in x and y in pixels.

- void **scroll_win** (INT8 x, INT8 y)
Move the window relative to its current position.
- void **set_sprite_data** (UINT8 first_tile, UINT8 nb_tiles, unsigned char *data)
Sets the tile patterns in the Sprite Tile Pattern table.
- void **get_sprite_data** (UINT8 first_tile, UINT8 nb_tiles, unsigned char *data)
- void **set_sprite_tile** (UINT8 nb, UINT8 tile)
Sets sprite n to display tile number t, from the sprite tile data.
- UINT8 **get_sprite_tile** (UINT8 nb)
- void **set_sprite_prop** (UINT8 nb, UINT8 prop)
Sets the property of sprite n to those defined in p.
- UINT8 **get_sprite_prop** (UINT8 nb)
- void **move_sprite** (UINT8 nb, UINT8 x, UINT8 y)
Moves the given sprite to the given position on the screen.
- void **scroll_sprite** (INT8 nb, INT8 x, INT8 y)
Moves the given sprite relative to its current position.
- void **set_data** (unsigned char *vram_addr, unsigned char *data, UINT16 len)
- void **get_data** (unsigned char *data, unsigned char *vram_addr, UINT16 len)
- void **set_tiles** (UINT8 x, UINT8 y, UINT8 w, UINT8 h, unsigned char *vram_addr, unsigned char *tiles)
- void **get_tiles** (UINT8 x, UINT8 y, UINT8 w, UINT8 h, unsigned char *tiles, unsigned char *vram_addr)

Variables

- UINT8 **_cpu**
GB type (GB, PGB, CGB).
- UINT16 **sys_time**
Time in VBL periods (60Hz).
- UINT8 **_io_status**
An OR of IO_.*
- UINT8 **_io_in**
Byte just read.
- UINT8 **_io_out**
*Write the byte to send here before calling **send_byte()** (p. 18).*

Detailed Description

Gameboy specific functions.

Define Documentation

5.5.1 `#define J_START ()`

Joypad bits.

A logical OR of these is used in the `wait_pad` and `joypad` functions. For example, to see if the B button is pressed try

```
UINT8 keys; keys = joypad() (p. 24); if (keys & J_B) { ... }
```

See also:

`joypad()` (p. 24)

5.5.2 `#define M_DRAWING ()`

Screen modes.

Normally used by internal functions only.

5.5.3 `#define S_PALETTE ()`

If this is set, sprite colours come from OBJ1PAL.

Else they come from OBJ0PAL.

5.5.4 `#define VBL_IFLAG ()`

Vertical blank interrupt.

Occurs at the start of the vertical blank. During this period the video ram may be freely accessed.

5.5.5 `#define LCD_IFLAG ()`

Interrupt when triggered by the STAT register.

See the Pan doc.

5.5.6 `#define SWITCH_ROM_MBC1 (b)`

Switches the upper 16k bank of the 32k rom to bank rombank using the MBC1 controller.

By default the upper 16k bank is 1. Make sure the rom you compile has more than just bank 0 and bank 1, a 32k rom. This is done by feeding `lcc.exe` the following switches:

`-Wl-yt#` where `#` is the type of cartridge. 1 for ROM+MBC1.

-Wl-yo# where # is the number of rom banks. 2,4,8,16,32.

5.5.7 **#define DISPLAY_ON ()**

Turns the display back on.

See also:

display_off() (p. 25), **DISPLAY_OFF()** (p. 22)

5.5.8 **#define DISPLAY_OFF ()**

Turns the display off immediatly.

See also:

display_off() (p. 25), **DISPLAY_ON()** (p. 22)

5.5.9 **#define SHOW_BKG ()**

Turns on the background layer.

Sets bit 0 of the LCDC register to 1.

5.5.10 **#define HIDE_BKG ()**

Turns off the background layer.

Sets bit 0 of the LCDC register to 0.

5.5.11 **#define HIDE_WIN ()**

Turns off the window layer.

Clears bit 5 of the LCDC register to 0.

5.5.12 **#define SHOW_SPRITES ()**

Turns on the sprites layer.

Sets bit 1 of the LCDC register to 1.

5.5.13 **#define HIDE_SPRITES ()**

Turns off the sprites layer.

Clears bit 1 of the LCDC register to 0.

5.5.14 `#define SPRITES_8x16 ()`

Sets sprite size to 8x16 pixels, two tiles one above the other.

Sets bit 2 of the LCDC register to 1.

5.5.15 `#define SPRITES_8x8 ()`

Sets sprite size to 8x8 pixels, one tile.

Clears bit 2 of the LCDC register to 0.

Function Documentation

5.5.16 `void remove_VBL (int_handler h)`

The remove functions will remove any interrupt handler.

A handler of NULL will cause bad things to happen.

5.5.17 `void add_VBL (int_handler h)`

Adds a V-blank interrupt handler.

The handler 'h' will be called whenever a V-blank interrupt occurs. Up to 4 handlers may be added, with the last added being called last. If the remove_VBL function is to be called, only three may be added.

See also:

`remove_VBL()` (p. 23)

5.5.18 `void add_LCD (int_handler h)`

Adds a LCD interrupt handler.

Called when the LCD interrupt occurs, which is normally when `LY_REG == LYC_REG`.

From pan/k0Pa: There are various reasons for this interrupt to occur as described by the STAT register (\$FF40). One very popular reason is to indicate to the user when the video hardware is about to redraw a given LCD line. This can be useful for dynamically controlling the SCX/ SCY registers (\$FF43/\$FF42) to perform special video effects.

See also:

`add_VBL()` (p. 23)

5.5.19 `void add_TIM (int_handler h)`

Adds a timer interrupt handler.

From pan/k0Pa: This interrupt occurs when the TIMA register (\$FF05) changes from \$FF to \$00.

See also:

add_VBL() (p. 23)

5.5.20 void delay (UINT16 *d*)

Delays the given number of milliseconds.

Uses no timers or interrupts, and can be called with interrupts disabled (why nobody knows :)

5.5.21 UINT8 joypad (void)

Reads and returns the current state of the joypad.

Follows Nintendo's guidelines for reading the pad. Return value is an OR of J_*

See also:

J_START() (p. 21)

5.5.22 UINT8 waitpad (UINT8 *mask*)

Waits until all the keys given in mask are pressed.

Normally only used for checking one key, but it will support many, even J_LEFT at the same time as J_RIGHT :)

See also:

joypad() (p. 24), **J_START()** (p. 21)

5.5.23 void enable_interrupts (void)

Enables unmasked interrupts.

See also:

disable_interrupts() (p. 24)

5.5.24 void disable_interrupts (void)

Disables interrupts.

This function may be called as many times as you like; however the first call to enable_interrupts will re-enable them.

See also:

enable_interrupts() (p. 24)

5.5.25 void set_interrupts (UINT8 *flags*)

Clears any pending interrupts and sets the interrupt mask register IO to flags.

See also:

VBL_IFLAG() (p. 21)

Parameters:

flags A logical OR of *_IFLAGS

5.5.26 A logical OR of *_-

IFLAGS

vbl_-

done

wait_vbl_done (void)

Waits for the vertical blank interrupt (VBL) to finish.

This can be used to sync animation with the screen re-draw. If VBL interrupt is disabled, this function will never return. If the screen is off this function returns immediatly.

**5.5.27 PP Switches the upper k bank of the k rom to bank rombank
however the first call to enable_interrupts will re enable them
PP fBSee this function will never return If the screen is off this
function returns immediatly SS void display_off (void)**

Turns the display off.

Waits until the VBL interrupt before turning the display off.

See also:

DISPLAY_ON() (p. 22)

5.5.28 void hiramcpy (UINT8 *dst*, const void * *src*, UINT8 *n*)

Copies data from somewhere in the lower address space to part of hi-ram.

Parameters:

dst Offset in high ram (0xFF00 and above) to copy to.

src Area to copy from

n Number of bytes to copy.

5.5.29 void set_bkg_data (UINT8 *first_tile*, UINT8 *nb_tiles*, unsigned char * *data*)

Sets the tile patterns in the Background Tile Pattern table.

Starting with the tile pattern *x* and carrying on for *n* number of tile patterns. Taking the values starting from the pointer data. Note that patterns 128-255 overlap with patterns 128-255 of the sprite Tile Pattern table.

GBC: Depending on the VBK_REG this determines which bank of Background tile patterns are written to. VBK_REG=0 indicates the first bank, and VBK_REG=1 indicates the second.

Parameters:

first_tile Range 0 - 255

nb_tiles Range 0 - 255

5.5.30 void set_bkg_tiles (UINT8 *x*, UINT8 *y*, UINT8 *w*, UINT8 *h*, unsigned char * *tiles*)

Sets the tiles in the background tile table.

Starting at position *x*,*y* in tiles and writing across for *w* tiles and down for *h* tiles. Taking the values starting from the pointer data.

For the GBC, also see the pan/k00Pa section on VBK_REG.

Parameters:

x Range 0 - 31

y Range 0 - 31

w Range 0 - 31

h Range 0 - 31

data Pointer to an unsigned char. Usually the first element in an array.

5.5.31 void move_bkg (UINT8 *x*, UINT8 *y*)

Moves the background layer to the position specified in *x* and *y* in pixels.

Where 0,0 is the top left corner of the GB screen. You'll notice the screen wraps around in all 4 directions, and is always under the window layer.

5.5.32 void scroll_bkg (INT8 *x*, INT8 *y*)

Moves the background relative to it's current position.

See also:

`move_bkg()` (p. 26)

5.5.33 void set_win_data (UINT8 *first_tile*, UINT8 *nb_tiles*, unsigned char * *data*)

Sets the window tile data.

This is the same as `set_bkg_data`, as both the window layer and background layer share the same Tile Patterns.

See also:

`set_bkg_data()` (p. 25)

5.5.34 void set_win_tiles (UINT8 *x*, UINT8 *y*, UINT8 *w*, UINT8 *h*, unsigned char * *tiles*)

Sets the tiles in the win tile table.

Starting at position x,y in tiles and writing across for w tiles and down for h tiles. Taking the values starting from the pointer data. Note that patterns 128-255 overlap with patterns 128-255 of the sprite Tile Pattern table.

GBC only. Depending on the VBK_REG this determines if you're setting the tile numbers VBK_REG=0; or the attributes for those tiles VBK_REG=1;. The bits in the attributes are defined as: Bit 7 - Priority flag. When this is set, it puts the tile above the sprites with colour 0 being transparent. 0: below sprites, 1: above sprites Note SHOW_BKG needs to be set for these priorities to take place. Bit 6 - Vertical flip. Dictates which way up the tile is drawn vertically. 0: normal, 1: upside down. Bit 5 - Horizontal flip. Dictates which way up the tile is drawn horizontally. 0: normal, 1:back to front. Bit 4 - Not used. Bit 3 - Character Bank specification. Dictates from which bank of Background Tile Patterns the tile is taken. 0: Bank 0, 1: Bank 1 Bit 2 - See bit 0. Bit 1 - See bit 0. Bit 0 - Bits 0-2 indicate which of the 7 BKG colour palettes the tile is assigned.

Parameters:

x Range 0 - 31

y Range 0 - 31

w Range 0 - 31

h Range 0 - 31

5.5.35 void move_win (UINT8 *x*, UINT8 *y*)

Moves the window layer to the position specified in x and y in pixels.

Where 7,0 is the top left corner of the GB screen. The window is locked to the bottom right corner, and is always over the background layer.

See also:

SHOW_WIN() (p. 17), HIDE_WIN() (p. 22)

5.5.36 void scroll_win (INT8 *x*, INT8 *y*)

Move the window relative to its current position.

See also:

move_win() (p. 27)

5.5.37 void set_sprite_data (UINT8 *first_tile*, UINT8 *nb_tiles*, unsigned char * *data*)

Sets the tile patterns in the Sprite Tile Pattern table.

Starting with the tile pattern x and carrying on for n number of tile patterns. Taking the values starting from the pointer data. Note that patterns 128-255 overlap with patterns 128-255 of the Background Tile Pattern table.

GBC only. Depending on the VBK_REG this determines which bank of Background tile patterns are written to. VBK_REG=0 indicates the first bank, and VBK_REG=1 indicates the second.

5.5.38 void set_sprite_tile (UINT8 *nb*, UINT8 *tile*)

Sets sprite *n* to display tile number *t*, from the sprite tile data.

If the GB is in 8x16 sprite mode then it will display the next tile, *t*+1, below the first tile.

Parameters:

nb Sprite number, range 0 - 39

5.5.39 void set_sprite_prop (UINT8 *nb*, UINT8 *prop*)

Sets the property of sprite *n* to those defined in *p*.

Where the bits in *p* represent: Bit 7 - Priority flag. When this is set the sprites appear behind the background and window layer. 0: in front, 1: behind. Bit 6 - GBC only. Vertical flip. Dictates which way up the sprite is drawn vertically. 0: normal, 1: upside down. Bit 5 - GBC only. Horizontal flip. Dictates which way up the sprite is drawn horizontally. 0: normal, 1: back to front. Bit 4 - DMG only. Assigns either one of the two b/w palettes to the sprite. 0: OBJ palette 0, 1: OBJ palette 1. Bit 3 - GBC only. Dictates from which bank of Sprite Tile Patterns the tile is taken. 0: Bank 0, 1: Bank 1. Bit 2 - See bit 0. Bit 1 - See bit 0. Bit 0 - GBC only. Bits 0-2 indicate which of the 7 OBJ colour palettes the sprite is assigned.

Parameters:

nb Sprite number, range 0 - 39

5.5.40 void move_sprite (UINT8 *nb*, UINT8 *x*, UINT8 *y*)

Moves the given sprite to the given position on the screen.

Dont forget that the top left visible pixel on the screen is at (8,16). To put sprite 0 at the top left, use `move_sprite(0, 8, 16);`

Variable Documentation

5.5.41 UINT8 _io_out

Write the byte to send here before calling `send_byte()` (p. 18).

See also:

`send_byte()` (p. 18)

5.6 gbdk-lib.h File Reference

Settings for the greater library system.

Detailed Description

Settings for the greater library system.

5.7 hardware.h File Reference

Defines that let the GB's hardware registers be accessed from C.

Detailed Description

Defines that let the GB's hardware registers be accessed from C.

See the Pan doc for what to set them to.

5.8 malloc.h File Reference

Header for a simple implementation of malloc().

Compounds

- struct **smalloc_hunk**

Defines

- #define **MALLOC_FREE**
The malloc hunk flags Note: Could have used a negative size a'la TI.
- #define **MALLOC_MAGIC**
Magic number of a header.

Typedefs

- typedef struct smalloc_hunk **mmalloc_hunk**

Functions

- void **malloc_gc** (void)
Garbage collect (join free hunks).
- void **debug** (char *routine, char *msg)
debug message logger.

Variables

- struct smalloc_hunk* **pmmalloc_hunk**
- **UBYTE malloc_heap_start**
Start of free memory, as defined by the linker.
- pmmalloc_hunk **malloc_first**
First hunk.

Detailed Description

Header for a simple implementation of malloc().

This library may currently be broken.

Define Documentation

5.8.1 `#define MALLOC_MAGIC ()`

Magic number of a header.

Gives us some chance of surviving if the list is corrupted

5.9 rand.h File Reference

Random generator using the linear congruential method.

Functions

- **void initrand (UINT16 seed)**
Initialise the random number generator.
- **INT8 rand (void)**
Returns a random value.
- **UINT16 randw (void)**
Returns a random word.
- **void initarand (UINT16 seed)**
Random generator using the linear lagged additive method Note that 'initarand() (p. 33)' calls 'initrand() (p. 33)' with the same seed value, and uses 'rand() (p. 33)' to initialize the random generator.
- **INT8 arand (void)**
Generates a random number using the linear lagged additive method.

Detailed Description

Random generator using the linear congruential method.

Author(s):

Luc Van den Borre

Function Documentation

5.9.1 void initrand (UINT16 seed)

Initialise the random number generator.

seed needs to be different each time, else the same sequence will be generated. A good source is the DVI register.

5.9.2 void initarand (UINT16 seed)

Random generator using the linear lagged additive method Note that 'initarand() (p. 33)' calls 'initrand() (p. 33)' with the same seed value, and uses 'rand() (p. 33)' to initialize the random generator.

Author(s):

Luc Van den Borre

5.10 sample.h File Reference

Playback raw sound sample with length len from start at 8192Hz rate.

Functions

- void **play_sample** (UINT8 *start, UINT16 len)

Play the given, appropriately formatted sample.

Detailed Description

Playback raw sound sample with length len from start at 8192Hz rate.

len defines the length of the sample in samples/32 or bytes/16. The format of the data is unsigned 4-bit samples, 2 samples per byte, upper 4-bits played before lower 4 bits.

Adaption for GBDK by Lars Malmborg. Original code by Jeff Frohwein.

5.11 sgb.h File Reference

Super Gameboy definitions.

Functions

- **UINT8 sgb_check** (void)

Return a non-null value if running on Super GameBoy.

Detailed Description

Super Gameboy definitions.

5.12 stdio.h File Reference

Basic file/console input output functions.

Functions

- void **putchar** (char c)
Put the character 'c' to stdout.
- void **printf** (const char *format, ...)
Print the string and arguments given by format to stdout.
- void **sprintf** (char *str, const char *format, ...)
Print the string and arguments given by format to a buffer.
- void **puts** (const char *s)
puts() (p. 36) *writes the string s and a trailing newline to std out.*
- char* **gets** (char *s)
gets() (p. 36) *reads a line from stdin into the buffer pointed to by s until either a terminating newline or EOF, which it replaces with '\0'.*

Detailed Description

Basic file/console input output functions.

Function Documentation

5.12.1 void printf (const char * *format*, ...)

Print the string and arguments given by format to stdout.

Currently supported: %c (character), %u (unsigned int), %d (signed int), %x (unsigned int as hex), and %s (string). Does not return the number of characters printed.

5.12.2 void sprintf (char * *str*, const char * *format*, ...)

Print the string and arguments given by format to a buffer.

Currently supported: %c (character), %u (unsigned int), %d (signed int), %x (unsigned int as hex), and %s (string). Does not return the number of characters printed.

Parameters:

str The buffer to print into.

format The format string as per printf.

5.12.3 char * gets (char * s)

gets() (p. 36) reads a line from stdin into the buffer pointed to by s until either a terminating newline or EOF, which it replaces with '\0'.

No check for buffer overrun is performed.

5.13 string.h File Reference

Generic string functions.

Functions

- **char* strcpy** (char *dest, const char *src)
Copies the string pointed to be src (including the terminating '\0' character) to the array pointed to by dest.
- **int strcmp** (const char *s1, const char *s2)
Compares the two strings s1 and s2.
- **void* memcpy** (void *dest, const void *src, size_t len)
Copies n bytes from memory area src to memory area dest.
- **char* reverse** (char *s)
- **char* strcat** (char *s1, const char *s2)
- **int strlen** (const char *s)
- **char* strncat** (char *s1, const char *s2, int n)
- **int strncmp** (const char *s1, const char *s2, int n)
- **char* strncpy** (char *s1, const char *s2, int n)

Detailed Description

Generic string functions.

Function Documentation

5.13.1 char * strcpy (char * *dest*, const char * *src*)

Copies the string pointed to be src (including the terminating '\0' character) to the array pointed to by dest.

The strings may not overlap, and the destination string dest must be large enough to receive the copy.

Parameters:

dest Array to copy into.

src Array to copy from.

Returns:

A pointer to dest.

5.13.2 int strcmp (const char * *s1*, const char * *s2*)

Compares the two strings *s1* and *s2*.

It returns an integer less than, equal to, or greater than zero if *s1* is found, respectively, to be less than, to match, or be greater than *s2*.

5.13.3 void * memcpy (void * *dest*, const void * *src*, size_t *len*)

Copies *n* bytes from memory area *src* to memory area *dest*.

The memory areas may not overlap.

Parameters:

dest Array to copy into.

src Array to copy from.

len The length in bytes of *src*.

Returns:

A pointer to *dest*.

5.14 time.h File Reference

Sort of ANSI compliant time functions.

Typedefs

- typedef **UINT16** **time_t**

Functions

- **clock_t** **clock** (void)
*The **clock()** (p. 40) function returns an approximation of processor time used by the program.*
 - **time_t** **time** (time_t *t)
-

Detailed Description

Sort of ANSI compliant time functions.

Function Documentation

5.14.1 clock_t clock (void)

The **clock()** (p. 40) function returns an approximation of processor time used by the program.

The value returned is the CPU time used so far as a **clock_t**; to get the number of seconds used, divide by **CLOCKS_PER_SEC**.

5.15 asm/gbz80/types.h File Reference

Types definitions for the gb.

Typedefs

- typedef char **INT8**
Signed eight bit.
- typedef unsigned char **UINT8**
Unsigned eight bit.
- typedef int **INT16**
Signed sixteen bit.
- typedef unsigned int **UINT16**
Unsigned sixteen bit.
- typedef long **INT32**
Signed 32 bit.
- typedef unsigned long **UINT32**
Unsigned 32 bit.
- typedef int **size_t**
- typedef **UINT16** **clock_t**
Returned from clock.

Detailed Description

Types definitions for the gb.

Typedef Documentation

5.15.1 typedef UINT16 clock_t

Returned from clock.

See also:

clock() (p. 40)

5.16 asm/types.h File Reference

Shared types definitions.

Compounds

- union **_fixed**

Typedefs

- typedef **INT8 BOOLEAN**
TRUE or FALSE.
- typedef **INT8 BYTE**
Signed 8 bit.
- typedef **UINT8 UBYTE**
Unsigned 8 bit.
- typedef **INT16 WORD**
Signed 16 bit.
- typedef **UINT16 UWORD**
Unsigned 16 bit.
- typedef **INT32 LWORD**
Signed 32 bit.
- typedef **UINT32 ULWORD**
Unsigned 32 bit.
- typedef **INT32 DWORD**
Signed 32 bit.
- typedef **UINT32 UDWORD**
Unsigned 32 bit.
- typedef union **_fixed fixed**
Useful definition for fixed point values.

Detailed Description

Shared types definitions.

Index

- `_cpu`
 - `gb.h`, 20
- `_fixed`, 7
 - `w`, 7
- `_fixed`
 - `b`, 7
- `_io.in`
 - `gb.h`, 20
- `_io.out`
 - `gb.h`, 28
- `_io.status`
 - `gb.h`, 20
- `add_LCD`
 - `gb.h`, 23
- `add_TIM`
 - `gb.h`, 23
- `add_VBL`
 - `gb.h`, 23
- `arand`
 - `rand.h`, 33
- `asm/gbz80/types.h`, 41
 - `clock_t`, 41
 - `INT16`, 41
 - `INT32`, 41
 - `INT8`, 41
 - `size_t`, 41
 - `UINT16`, 41
 - `UINT32`, 41
 - `UINT8`, 41
- `asm/types.h`, 42
 - `BOOLEAN`, 42
 - `BYTE`, 42
 - `DWORD`, 42
 - `fixed`, 42
 - `LWORD`, 42
 - `UBYTE`, 42
 - `UDWORD`, 42
 - `ULWORD`, 42
 - `UWORD`, 42
 - `WORD`, 42
- `b`
 - `_fixed`, 7
- `BOOLEAN`
 - `asm/types.h`, 42
- `box`
 - `drawing.h`, 14
- `BYTE`
 - `asm/types.h`, 42
- `cgb.h`, 9
 - `cgb_compatibility`, 9
 - `cpu_fast`, 10
 - `cpu_slow`, 10
 - `RGB`, 9
 - `RGB_RED`, 9
 - `set_bkg_palette`, 9
 - `set_bkg_palette_entry`, 9
 - `set_sprite_palette`, 9
 - `set_sprite_palette_entry`, 9
- `cgb_compatibility`
 - `cgb.h`, 9
- `CGB_TYPE`
 - `gb.h`, 17
- `circle`
 - `drawing.h`, 14
- `clock`
 - `time.h`, 40
- `clock_t`
 - `asm/gbz80/types.h`, 41
- `color`
 - `drawing.h`, 14
- `console.h`, 11
 - `gotoxy`, 11
 - `posx`, 11
 - `posy`, 11
 - `setchar`, 11
- `cpu_fast`
 - `cgb.h`, 10
- `cpu_slow`
 - `cgb.h`, 10
- `ctype.h`, 12
 - `isalpha`, 12
 - `isdigit`, 12
 - `islower`, 12
 - `isspace`, 12
 - `isupper`, 12
 - `tolower`, 12
 - `toupper`, 12

debug
 malloc.h, 31
 delay
 gb.h, 24
 disable_interrupts
 gb.h, 24
 DISPLAY_OFF
 gb.h, 22
 display_off
 gb.h, 25
 DISPLAY_ON
 gb.h, 22
 DMG_TYPE
 gb.h, 17
 draw_image
 drawing.h, 13
 drawing.h, 13
 box, 14
 circle, 14
 color, 14
 draw_image, 13
 getpix, 14
 gotogxy, 14
 gprint, 13
 gprintf, 13
 gprintln, 14
 gprintn, 13
 GRAPHICS_WIDTH, 13
 line, 13
 M_NOFILL, 13
 plot, 13
 plot_point, 13
 SIGNED, 13
 SOLID, 13
 switch_data, 13
 WHITE, 13
 wrtchr, 14
 DWORD
 asm/types.h, 42
 enable_interrupts
 gb.h, 24
 fixed
 asm/types.h, 42
 gb.h, 16
 _cpu, 20
 _io_in, 20
 _io_out, 28
 _io_status, 20
 add_LCD, 23
 add_TIM, 23
 add_VBL, 23
 CGB_TYPE, 17
 delay, 24
 disable_interrupts, 24
 DISPLAY_OFF, 22
 display_off, 25
 DISPLAY_ON, 22
 DMG_TYPE, 17
 enable_interrupts, 24
 get_bkg_tiles, 19
 get_data, 20
 get_mode, 18
 get_sprite_data, 20
 get_sprite_prop, 20
 get_sprite_tile, 20
 get_tiles, 20
 get_win_tiles, 19
 HIDE_BKG, 22
 HIDE_SPRITES, 22
 HIDE_WIN, 22
 hramcpy, 25
 int_handler, 18
 IO_ERROR, 17
 IO_IDLE, 17
 IO_RECEIVING, 17
 IO_SENDING, 17
 J_START, 21
 JOY_IFLAG, 16
 joypad, 24
 LCD_IFLAG, 21
 M_DRAWING, 21
 M_NO_INTERP, 16
 M_NO_SCROLL, 16
 MGB_TYPE, 17
 mode, 18
 move_bkg, 26
 move_sprite, 28
 move_win, 27
 receive_byte, 18
 remove_JOY, 18
 remove_LCD, 18
 remove_SIO, 18
 remove_TIM, 18
 remove_VBL, 23
 reset, 19
 S_FLIPX, 16
 S_FLIPY, 16
 S_PALETTE, 21
 S_PRIORITY, 16
 SCREENHEIGHT, 16
 SCREENWIDTH, 16
 scroll_bkg, 26
 scroll_sprite, 20
 scroll_win, 27
 send_byte, 18

- set_bkg_data, 25
- set_bkg_tiles, 26
- set_data, 20
- set_interrupts, 24
- set_sprite_data, 27
- set_sprite_prop, 28
- set_sprite_tile, 27
- set_tiles, 20
- set_win_data, 26
- set_win_tiles, 26
- SHOW_BKG, 22
- SHOW_SPRITES, 22
- SHOW_WIN, 17
- SIO_IFLAG, 16
- SPRITES_8x16, 22
- SPRITES_8x8, 23
- SWITCH_RAM_MBC1, 17
- SWITCH_RAM_MBC5, 17
- SWITCH_ROM_MBC1, 21
- SWITCH_ROM_MBC5, 17
- sys_time, 20
- TIM_IFLAG, 16
- VBL_IFLAG, 21
- wait_vbl_done, 25
- waitpad, 24
- waitpadup, 19
- gbdk-lib.h, 29
- get_bkg_tiles
 - gb.h, 19
- get_data
 - gb.h, 20
- get_mode
 - gb.h, 18
- get_sprite_data
 - gb.h, 20
- get_sprite_prop
 - gb.h, 20
- get_sprite_tile
 - gb.h, 20
- get_tiles
 - gb.h, 20
- get_win_tiles
 - gb.h, 19
- getpix
 - drawing.h, 14
- gets
 - stdio.h, 36
- gotogxy
 - drawing.h, 14
- gotoxy
 - console.h, 11
- gprint
 - drawing.h, 13
- gprintf
 - drawing.h, 13
- gprintln
 - drawing.h, 14
- gprintn
 - drawing.h, 13
- GRAPHICS_WIDTH
 - drawing.h, 13
- hardware.h, 30
- HIDE_BKG
 - gb.h, 22
- HIDE_SPRITES
 - gb.h, 22
- HIDE_WIN
 - gb.h, 22
- hiramcpy
 - gb.h, 25
- initarand
 - rand.h, 33
- initrand
 - rand.h, 33
- INT16
 - asm/gbz80/types.h, 41
- INT32
 - asm/gbz80/types.h, 41
- INT8
 - asm/gbz80/types.h, 41
- int_handler
 - gb.h, 18
- IO_ERROR
 - gb.h, 17
- IO_IDLE
 - gb.h, 17
- IO_RECEIVING
 - gb.h, 17
- IO_SENDING
 - gb.h, 17
- isalpha
 - ctype.h, 12
- isdigit
 - ctype.h, 12
- islower
 - ctype.h, 12
- isspace
 - ctype.h, 12
- isupper
 - ctype.h, 12
- J_START
 - gb.h, 21
- JOY_IFLAG
 - gb.h, 16
- joypad

- gb.h, 24
- LCD_IFLAG
 - gb.h, 21
- line
 - drawing.h, 13
- LWORD
 - asm/types.h, 42
- M_DRAWING
 - gb.h, 21
- M_NO_INTERP
 - gb.h, 16
- M_NO_SCROLL
 - gb.h, 16
- M_NOFILL
 - drawing.h, 13
- malloc.h, 31
 - debug, 31
 - malloc_first, 31
 - MALLOC_FREE, 31
 - malloc_gc, 31
 - malloc_heap_start, 31
 - MALLOC_MAGIC, 32
 - mmalloc_hunk, 31
 - pmmalloc_hunk, 31
- malloc_first
 - malloc.h, 31
- MALLOC_FREE
 - malloc.h, 31
- malloc_gc
 - malloc.h, 31
- malloc_heap_start
 - malloc.h, 31
- MALLOC_MAGIC
 - malloc.h, 32
- memcpy
 - string.h, 39
- MGB_TYPE
 - gb.h, 17
- mmalloc_hunk
 - malloc.h, 31
- mode
 - gb.h, 18
- move_bkg
 - gb.h, 26
- move_sprite
 - gb.h, 28
- move_win
 - gb.h, 27
- play_sample
 - sample.h, 34
- plot
 - drawing.h, 13
- plot_point
 - drawing.h, 13
- pmmalloc_hunk
 - malloc.h, 31
- posx
 - console.h, 11
- posy
 - console.h, 11
- printf
 - stdio.h, 36
- putchar
 - stdio.h, 36
- puts
 - stdio.h, 36
- rand
 - rand.h, 33
- rand.h, 33
 - arand, 33
 - initarand, 33
 - initrand, 33
 - rand, 33
 - randw, 33
- randw
 - rand.h, 33
- receive_byte
 - gb.h, 18
- remove_JOY
 - gb.h, 18
- remove_LCD
 - gb.h, 18
- remove_SIO
 - gb.h, 18
- remove_TIM
 - gb.h, 18
- remove_VBL
 - gb.h, 23
- reset
 - gb.h, 19
- reverse
 - string.h, 38
- RGB
 - cgb.h, 9
- RGB_RED
 - cgb.h, 9
- S_FLIPX
 - gb.h, 16
- S_FLIPY
 - gb.h, 16
- S_PALETTE
 - gb.h, 21
- S_PRIORITY

- gb.h, 16
- sample.h, 34
 - play_sample, 34
- SCREENHEIGHT
 - gb.h, 16
- SCREENWIDTH
 - gb.h, 16
- scroll_bkg
 - gb.h, 26
- scroll_sprite
 - gb.h, 20
- scroll_win
 - gb.h, 27
- send_byte
 - gb.h, 18
- set_bkg_data
 - gb.h, 25
- set_bkg_palette
 - cgb.h, 9
- set_bkg_palette_entry
 - cgb.h, 9
- set_bkg_tiles
 - gb.h, 26
- set_data
 - gb.h, 20
- set_interrupts
 - gb.h, 24
- set_sprite_data
 - gb.h, 27
- set_sprite_palette
 - cgb.h, 9
- set_sprite_palette_entry
 - cgb.h, 9
- set_sprite_prop
 - gb.h, 28
- set_sprite_tile
 - gb.h, 27
- set_tiles
 - gb.h, 20
- set_win_data
 - gb.h, 26
- set_win_tiles
 - gb.h, 26
- setchar
 - console.h, 11
- sgb.h, 35
 - sgb_check, 35
- sgb_check
 - sgb.h, 35
- SHOW_BKG
 - gb.h, 22
- SHOW_SPRITES
 - gb.h, 22
- SHOW_WIN

- gb.h, 17
- SIGNED
 - drawing.h, 13
- SIO_IFLAG
 - gb.h, 16
- size_t
 - asm/gbz80/types.h, 41
- SOLID
 - drawing.h, 13
- sprintf
 - stdio.h, 36
- SPRITES_8x16
 - gb.h, 22
- SPRITES_8x8
 - gb.h, 23
- stdio.h, 36
 - gets, 36
 - printf, 36
 - putchar, 36
 - puts, 36
 - sprintf, 36
- strcat
 - string.h, 38
- strcmp
 - string.h, 38
- strcpy
 - string.h, 38
- string.h, 38
 - memcpy, 39
 - reverse, 38
 - strcat, 38
 - strcmp, 38
 - strcpy, 38
 - strlen, 38
 - strncat, 38
 - strncmp, 38
 - strncpy, 38
- strlen
 - string.h, 38
- strncat
 - string.h, 38
- strncmp
 - string.h, 38
- strncpy
 - string.h, 38
- switch_data
 - drawing.h, 13
- SWITCH_RAM_MBC1
 - gb.h, 17
- SWITCH_RAM_MBC5
 - gb.h, 17
- SWITCH_ROM_MBC1
 - gb.h, 21
- SWITCH_ROM_MBC5

- gb.h, 17
- sys_time
 - gb.h, 20
- TIM_IFLAG
 - gb.h, 16
- time
 - time.h, 40
- time.h, 40
 - clock, 40
 - time, 40
 - time_t, 40
- time_t
 - time.h, 40
- tolower
 - ctype.h, 12
- toupper
 - ctype.h, 12
- UBYTE
 - asm/types.h, 42
- UDWORD
 - asm/types.h, 42
- UINT16
 - asm/gbz80/types.h, 41
- UINT32
 - asm/gbz80/types.h, 41
- UINT8
 - asm/gbz80/types.h, 41
- ULWORD
 - asm/types.h, 42
- UWORD
 - asm/types.h, 42
- VBL_IFLAG
 - gb.h, 21
- w
 - _fixed, 7
- wait_vbl_done
 - gb.h, 25
- waitpad
 - gb.h, 24
- waitpadup
 - gb.h, 19
- WHITE
 - drawing.h, 13
- WORD
 - asm/types.h, 42
- wrtchr
 - drawing.h, 14